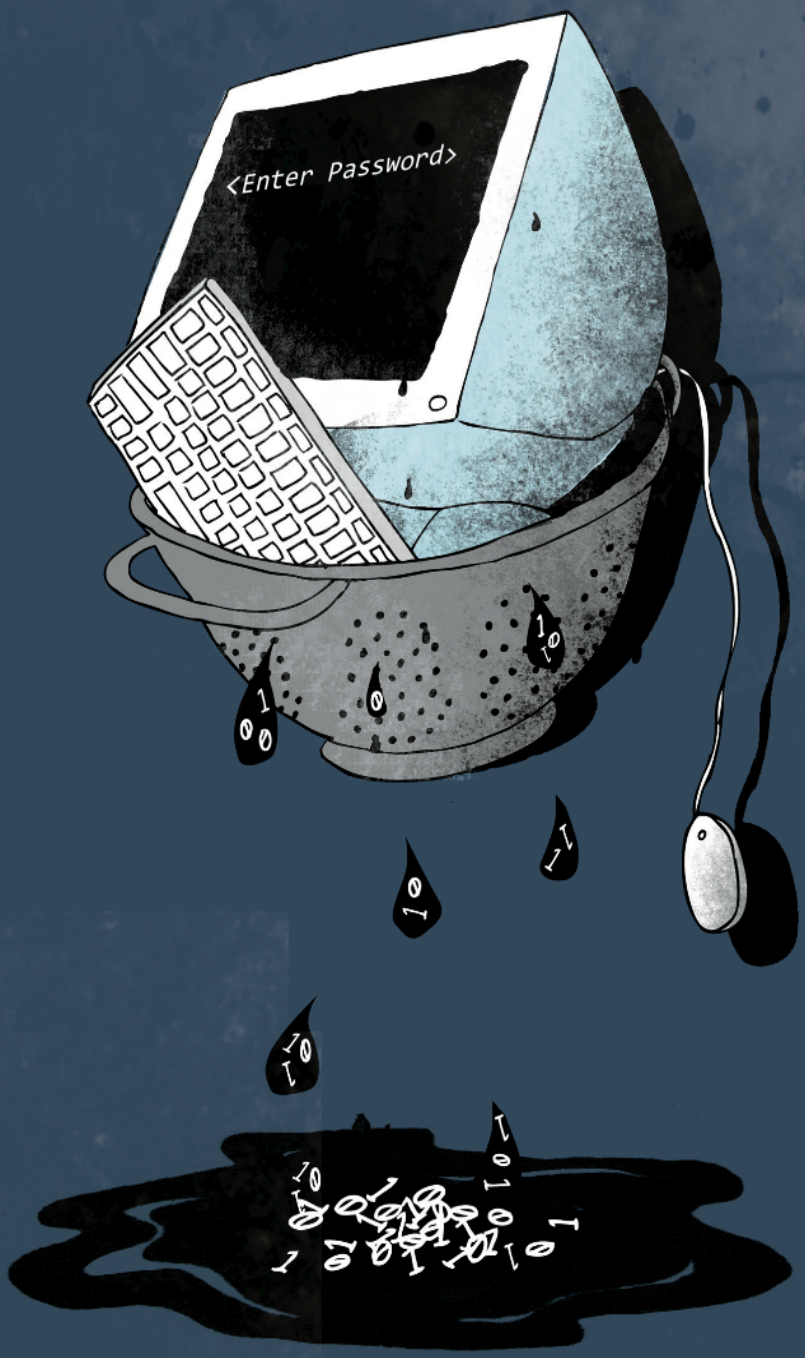
Quantitative Analysis of

Information Leakage

in Probabilistic and Nondeterministic Systems

Miguel E. Andrés



ISBN: 978-94-91211-74-4.
IPA dissertation series: 2011-09.

This thesis is typeset using LaTeX.
Translation of the Dutch summary: Peter van Rossum.
Cover designed by Marieke Meijer - www.mariekemeijer.com.
Thesis printed by Ipskamp Drukkers - www.ipskampdrukkers.nl.

The work in this thesis has been carried out at Radboud University and under the auspices of the research school IPA (Institute for Programming research and Algorithmics). The research funding was provided by the NWO Grant through the open project 612.000.526: Analysis of Anonymity. The author also wishes to acknowledge the French Institute for Research in Computer Science and Control (INRIA) for providing funding for several research visits to the École Polytechnique of Paris.

Quantitative Analysis of Information Leakage in Probabilistic and Nondeterministic Systems

A scientific essay in Science.

Doctoral thesis

to obtain the degree of doctor
from Radboud University Nijmegen
on the authority of the Rector Magnificus, Prof. dr. S.C.J.J. Kortmann,
according to the decision of the Council of Deans
to be defended in public on Friday, 1st July 2011
at 10:30 hours

by

Miguel E. Andrés

born in Río Cuarto, Córdoba, Argentina
on 02 July 1980.

Supervisor:

prof. dr. Bart P.F. Jacobs	Radboud University

Co-supervisors:

dr. Peter van Rossum	Radboud University
prof. dr. Catuscia Palamidessi	INRIA and Ecole Polytechnique

Doctoral Thesis Committee:

prof. dr. Joost-Pieter Katoen	RWTH Aachen University
dr. Pedro R. D'Argenio	Universidad Nacional de Córdoba
prof. dr. Frits W. Vaandrager	Radboud University
prof. dr. Holger Hermanns	Saarland University
dr. Mariëlle Stoelinga	University of Twente

Summary

As we dive into the digital era, there is growing concern about the amount of personal digital information that is being gathered about us. Websites often track people's browsing behavior, health care insurers gather medical data, and many smartphones and navigation systems store or transmit information that makes it possible to track the physical location of their users at any time. Hence, anonymity, and privacy in general, are increasingly at stake. Anonymity protocols counter this concern by offering anonymous communication over the Internet. To ensure the correctness of such protocols, which are often extremely complex, a rigorous framework is needed in which anonymity properties can be expressed, analyzed, and ultimately verified. Formal methods provide a set of mathematical techniques that allow us to rigorously specify and verify anonymity properties.

This thesis addresses the foundational aspects of formal methods for applications in security and in particular in anonymity. More concretely, we develop frameworks for the specification of anonymity properties and propose algorithms for their verification. Since in practice anonymity protocols always leak *some* information, we focus on quantitative properties which capture the *amount* of information leaked by a protocol.

We start our research on anonymity from its very foundations, namely conditional probabilities – these are the key ingredient of most quantitative anonymity properties. In Chapter 2 we present cpCTL, the first temporal logic making it possible to specify conditional probabilities. In addition, we present an algorithm to verify cpCTL formulas in a model-checking fashion. This logic, together with the model-checker, allows us to specify

and verify quantitative anonymity properties over complex systems where probabilistic and nondeterministic behavior may coexist.

We then turn our attention to more practical grounds: the constructions of algorithms to compute information leakage. More precisely, in Chapter 3 we present polynomial algorithms to compute the (information-theoretic) leakage of several kinds of fully probabilistic protocols (i.e. protocols without nondeterministic behavior). The techniques presented in this chapter are the first ones enabling the computation of (information-theoretic) leakage in interactive protocols.

In Chapter 4 we attack a well known problem in distributed anonymity protocols, namely full-information scheduling. To overcome this problem, we propose an alternative definition of schedulers together with several new definitions of anonymity (varying according to the attacker's power), and revise the famous definition of strong-anonymity from the literature. Furthermore, we provide a technique to verify that a distributed protocol satisfies some of the proposed definitions.

In Chapter 5 we provide (counterexample-based) techniques to debug complex systems, allowing for the detection of flaws in security protocols. Finally, in Chapter 6 we briefly discuss extensions to the frameworks and techniques proposed in Chapters 3 and 4.

Acknowledgements

This thesis would not have been possible without the continuous support of many people to whom I will always be grateful.

I am heartily thankful to my supervisor Bart Jacobs. He has closely followed the evolution of my PhD and made sure I always had all the resources a PhD student could possibly need.

I owe my deepest gratitude to my co-supervisor, Peter van Rossum. Four years have passed since he decided to take the risk to hire *me*, an Argentinian guy that he barely knew. I was really lucky to have Peter as my supervisor; he has always been very supportive, flexible, and extremely easygoing with me. I will never forget the football World Cup of 2006 (not that Argentina did very well); back then I was invited to spend one week in Nijmegen for an *official job interview*. But before I had the time to stress too much about formal talks and difficult questions, I found myself sharing a beer with Peter while watching Argentina vs the Netherlands (fortunately Argentina did not win — I still wonder what would have happened otherwise). This was just the first of many nice moments we shared together, including dinners, conversations, and trips. In addition to having fun, we have worked hard together — indeed we completed one of the most important proofs of this thesis at midnight after a long working day at Peter's house (and also after Mariëlle finally managed to get little Quinten to sleep ⌣̈).

I cannot allow myself to continue this letter without mentioning Catuscia Palamidessi. Much has changed in my life since I first met her in June 2007. Catuscia came then to visit our group in Nijmegen and we discovered that we had many research interests in common. Soon after, Catuscia invited me to visit her group in Paris and this turned out to be the beginning of a very fruitful collaboration. Since then we have spent countless days (and especially nights) working very hard together, writing many articles, and attending many conferences — including some in amazing places like Australia, Barbados, and Cyprus. Catuscia is not only an exceptionally bright and passionate scientist, but she is also one of the most thoughtful people I have ever met (placing the interests of her colleagues and PhD

students above her own), a wonderful person to work with (turning every work meeting into a relaxed intellectual discussion, enhanced with the finest *caffè italiano*), and, above all, an unconditional friend. For these reasons and many more (whose enumeration would require a second volume for this thesis), I am forever indebted to Catuscia.

This work has also greatly benefited from the insightful remarks and suggestions of the members of the reading committee Joost-Pieter Katoen, Pedro D'Argenio, and Frits Vaandrager, whom I wish to thank heartily. To Pedro I am also grateful for his sustained guidance and support in my life as a researcher. Many results in this thesis are a product of joint work, and apart from Peter and Catuscia, I am grateful to my co-authors Mário S. Alvim, Pedro R. D'Argenio, Geoffrey Smith and Ana Sokolova, all of whom shared their expertise with me. I am also thankful to Jasper Berendsen, Domingo Gómez, David Jansen, Mariëlle Stoelinga, Tingting Han, Sergio Giro, Jérémy Dubreil, and Konstantinos Chatizikokolakis for many fruitful discussions during my time as a PhD student. Also many thanks to Anne-Lise Laurain for her constant (emotional and technical) support during the writing of my thesis, Alexandra Silva for her insightful comments on the introduction of this work, and Marieke Meijer for devoting her artistic talent to the design of the cover of this thesis.

Special thanks to my paranymphs and dear friends Vicenç, Igor, Cristina, and Flavio. Together we have shared so much... uncountable lunches in the Refter, coffees in the blue coaches, and trips around the world among many more experiences. But, even more importantly, we have always been there to support each other in difficult times, and this is worth the world to me.

I wish to thank my colleagues in the DS group for providing such a friendly atmosphere which contributed greatly to the realization of my PhD. I explicitly want to thank Alejandro, Ana, Chris, Christian, Erik, Fabian, Gerhard, Ichiro, Jorik, Ken, Łukasz, Olha, Pieter, Pim, Roel, Thanh Son, and Wojciech with whom I have shared many coffees, nice conversations, table tennis, and much more. My journey in the DS group would not have been as easy if it was not for Maria and Desiree whose help on administrative issues saved me lots of pain; as for any kind of technical problem or

just IT advice, Ronny and Engelbert have been always very helpful.

I also wish to thank all the members of the Comète group for making me feel welcome in such a wonderful and fun group. In particular, I want to thank the Colombian crowd – Frank, Andrés, and Luis – for the nice nights out to "La Peña" and all the fun we had together, Jérémy for introducing me to the tennis group, Sophia for helping correct my English in this thesis, Kostas for many interesting conversations on the most diverse topics of computer science (and life in general), and Mário for finally acknowledging the superiority of the Argentinian soccer over the Brazilian one ◡̈.

Along the years, I always had a very happy and social life in Nijmegen. Many people, in addition to my paranymphs, have greatly contributed to this. A very big thanks to "Anita", for so many happy moments and her unconditional support during my life in Nijmegen. Also many thanks to my dear friend Renée, every time I hear "It is difficult to become friends with Dutch people, but once they are your friends they would NEVER let you down" I have to think of her. Special thanks to Elena and Clara for being there for me when I could not even speak English, Errez for sharing his wisdom in hunting matters ◡̈, Hilje and Daphne for being there for me when I just arrived to Nijmegen (this was very important to me), also thanks to Klasien, David, and Patricia for many nice nights out and to my dear neighbours – Marleen, Kim, and Marianne – for welcoming me in their house and being always so understanding with me. Besides, I would like to thank to the "Blonde Pater crowd" including Cristina, Christian, Daniela, Davide, Deniz, Francesco, Jordan, Mariam, Nina, Shankar, and Vicenç with all of whom I shared many nice cappuccinos, conversations, and nights out. Special thanks to the sweet Nina for being always willing help me, to Francesco for the wonderful guitar nights, to Mariam for her help in Dutch, and to Christian... well, simply for his "buena onda".

More than four years have passed since I left Argentina, and many things have certainly changed in my life. However, the support and affection of my lifetime friends have remained immutable. Many thanks to Duro, Seba, Tony, Viole, Gabi, and Martín for all they have contributed to my life. Special thanks to my "brothers" Tincho and Lapin, my life would not be the same without them.

Last but not least, all my affection to my family: dad Miguel, mum Bacho, sister Josefina, and brothers Ignacio and Augusto. Every single success in my life I owe mostly to them. Thanks to my brothers and sister for their constant support and everlasting smiles, which mean to me more than I can express in words. Thanks to my parents for their incessant and selfless sacrifice, thanks to which their children have had all anybody could possible require to be happy and successful. My parents are the greatest role models for me and to them I dedicate this thesis.

To my dear parents, Miguel and Bacho.

Por último, el "gracias" mas grande del mundo es para mis queridos padres - Miguel y Bacho - y hermanos - Ignacio, Josefina y Augusto. Porque cada logro conseguido en mi vida, ha sido (en gran parte) gracias a ellos. A mis hermanos, les agradezco su apoyo y sonrisas constantes, que significaron y siguen significando para mí mucho más de lo que las palabras puedan expresar. A mis padres, su incansable y desinteresado sacrificio, gracias al cual sus hijos han tenido y tienen todas las oportunidades del mundo para ser felices y exitosos. Ellos son, sin lugar a dudas, mi ejemplo de vida, y a ellos dedico esta tesis:

A mis queridos padres, Miguel y Bacho.

Miguel E. Andrés
Paris, May 2011.

Contents

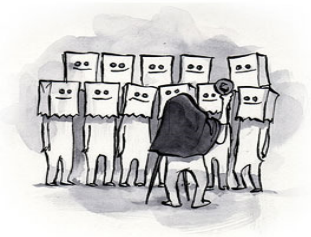
Chapter 1

Introduction

1.1 Anonymity

The world *Anonymity* derives from the Greek *ἀνωνυμία*, which means "without a name". In general, this term is used to express the fact that the identity of an individual is not publicly known.

Since the beginning of human society, anonymity has been an important issue. For instance, people have always felt the need to be able to express their opinions without being identified, because of the fear of social and economical retribution, harassment, or even threats to their lives.



1.1.1 The relevance of anonymity nowadays

With the advent of the Internet, the issue of anonymity has been magnified to extreme proportions. On the one hand, the Internet increases the opportunities of interacting online, communicating information, expressing opinion in public forums, etc. On the other hand, by using the Internet we are disclosing information about ourselves: every time we visit a website certain data about us may be recorded. In this way, organizations

like multinational corporations can build a permanent, commercially valuable record of our interests. Similarly, every email we send goes through multiple control points and it is most likely scanned by profiling software belonging to organizations like the National Security Agency of the USA. Such information can be used against us, ranging from slightly annoying practices like commercial spam, to more serious offences like stealing credit cards' information for criminal purposes.

Anonymity, however, is not limited to individual issues: it has considerable social and political implications. In countries controlled by repressive governments, the Internet is becoming increasingly more restricted, with the purpose of preventing their citizens from accessing uncensored information and from sending information to the outside world. The role of anonymizing technologies in this scenario is twofold: (1) they can help accessing sources of censored information via proxies (2) they can help individuals to freely express their ideas (for instance via online forums).

The practice of censoring the Internet is actually not limited to repressive governments. In fact, a recent research project conducted by the universities of Harvard, Cambridge, Oxford and Toronto, studied government censorship in 46 countries and concluded that 25 of them, including various western countries, filter to some extent communications concerning political or religious positions.

Anonymizing technologies, as most technologies, can also be used for malicious purposes. For instance, they can be used to help harassment, hate speech, financial scams, disclosure of private information, etc. Because of their nature, they are actually more controversial than other technologies: people are concerned that terrorists, pedophiles, or other criminals could take advantage of them.

Whatever is the use one can make of anonymity, and the personal view one may have on this topic, it is clearly important to be able to assess the degree of anonymity of a given system. This is one of the aims of this thesis.

1.1.2 Anonymizing technologies nowadays

The most common use of anonymizing technologies is to prevent observers from discovering the source of communications.

This is not an easy task, since in general users must include in the message information about themselves. In practice, for Internet communication, this information is the (unique) IP address of the computer in use, which specifies its location in the topology of the network. This IP number is usually logged along with the host name (logical name of the sender). Even when the user connects to the Internet with a temporary IP number assigned to him for a single session, this number is in general logged by the ISP (Internet Service Provider), which makes it possible, with the ISP's collaboration, to know who used a certain IP number at a certain time and thus to find out the identity of the user.

The currently available anonymity tools aim at preventing the observers of an online communication from learning the IP address of the participants. Most applications rely on proxies, i.e. intermediary computers to which messages are forwarded and which appear then as senders of the communication, thus hiding the original initiator of the communication. Setting up a proxy server nowadays is easy to implement and maintain. However, single-hop architectures in which all users enter and leave through the same proxy, create a single point of failure which can significantly threaten the security of the network. Multi-hop architectures have therefore been developed to increase the performance as well as the security of the system. In the so-called daisy-chaining anonymization for instance, traffic hops deliberately via a series of participating nodes (changed for every new communication) before reaching the intended receiver, which prevents any single entity from identifying the user. Anonymouse [Ans], FilterSneak [Fil] and Proxify [Pro] are well-known free web based proxies, while Anonymizer [Ane] is currently one of the leading commercial solutions.

1.1.3 Anonymizing technologies: a bit of history

Anonymous posting/reply services on the Internet were started around 1988 and were introduced primarily for use on specific newsgroups which discussed particularly volatile, sensitive and personal subjects. In 1992, anonymity services using remailers were originated by Cypherpunk. Global anonymity servers which served the entire Internet soon sprang up, combining the functions of anonymous posting as well as anonymous remailing in one service. The new global services also introduced the concept of pseudonymity which allowed anonymous mail to be replied.

The first popular anonymizing tool was the Penet remailer developed by Johan Helsingius of Finland in the early 1990s. The tool was originally intended to serve only Scandinavia but Helsingius eventually expanded to worldwide service due to a flood of international requests.

Based on this tool, in 1995, Mikael Berglund made a study on how anonymity was used. His study was based on scanning all publicly available newsgroups in a Swedish Usenet News server. He randomly selected a number of messages from users of the Penet remailer and classified them by topic. His results are shown in Table 1.1.

In 1993, Cottrell wrote the Mixmaster remailer and two years later he launched Anonymizer which became the first Web-based anonymity tool.

1.1.4 Anonymity and computer science

The role of computer science with respect to anonymity is twofold. On one the hand, the theory of communication helps in the design and implementation of anonymizing protocols. On the other hand, like for all software systems, there is the issue of correctness, i.e., of ensuring that the protocol achieves the expected anonymity guarantees.

While most of the work on anonymity in the literature belongs to the first challenge, this thesis addresses the second one. Ensuring the correctness of a protocol involves (1) the use of formalisms to precisely model the behaviour of the protocol, and (2) the use of formalisms to specify unambiguously the desired properties. Once the protocol and its desired prop-

Percentage	Type of message	
30,0 %	Discussion	
		Common topics: Sex, hobby, work, religion, politics, ethics, software.
23,1 %	Advertisements	
		Common topics: Sexual/romantic contact advertisements dominated, a few other advertisements also used anonymity, for example ads searching for friends with a particular interest. The authors of contact ads were mostly male.
16,5 %	Questions and answers	
		Common topics: Computer software issues, sex, medicine and drugs.
13,2 %	Texts	
		Common topics: Pornographic texts, about 50 % heterosexual and 50 % homosexual (purported to be written by both men and women), jokes, sometimes nasty.
9,9 %	Test messages	
		To try out if the anonymity server works.
3,7 %	Pictures	
		Mostly erotic/pornographic.
0,4 %	Computer software	
3,3 %	Unclassifiable	
		Written in a language the researcher could not read, such as several messages in Chinese. Note the repressive political regime in China, which may be a reason why there were several people who needed to use an anonymity server in discussing issues in that language.

Figure 1.1: Statistics on the Use of Anonymity – Penet

erties have been specified, it is possible to employ verification techniques to prove formally that the specified model satisfy such properties. These topics belong to the branch of computer science called *formal methods*.

1.2 Formal methods

Formal methods are a particular kind of mathematically-based techniques used in computer science and software engineering for the specification and verification of software and hardware systems. These techniques have their

foundations on the most diverse conceptual frameworks: logic calculi, automata theory, formal languages, program semantics, etc.

1.2.1 The need of formal verification

As explained in previous sections, internet technologies play an important role in our lives. However, Internet is not the only kind of technology we are in contact with: Every day we interact with embedded systems such as mobile phones, smart cards, GPS receivers, videogame consoles, digital cameras, DVD players, etc. Technology also plays an important role in critical-life systems, i.e., systems where the malfunction of any component may incur in life losses. Example of such systems can be found in the areas of medicine, aeronautics, nuclear energy generation, etc.

The malfunction of a technological device can have important negative consequences ranging from material to life loss. In the following we list some famous examples of disasters caused by software failure.

Material loss: In 2004, the Air Traffic Control Center of Los Angeles International Airport lost communication with Airplanes causing the immediate suspension of all operations. The failure in the radio system was due to a 32-bit countdown timer that decremented every millisecond. Due to a bug in the software, when the counter reached zero the system shut down unexpectedly. This communication outage disrupted about 600 flights (including 150 cancellations) impacting over 30.000 passengers and causing millionaire losses to airway companies involved.

In 1996, an Ariane 5 rocket launched by the European Space Agency exploded just forty seconds after lift-off. The rocket was on its first voyage, after a decade of development costing U$S 7 billion. The destroyed rocket and its cargo were valued at U$S 500 million. A board of inquiry investigated the causes of the explosion and in two weeks issued a report. It

turned out that the cause of the failure was a software error in the inertial reference system. Specifically a 64 bit floating point number related to the horizontal velocity of the rocket was converted to a 16 bit signed integer.

In the early nineties a bug (discovered by a professor of Lynchburg College, USA) in the floating-point division unit of the processor Intel Pentium II not only severely damaged Intel's reputation, but it also forced the replacement of faulty processors causing a loss of 475 million US dollars for the company.

Fatal loss: A software flaw in the control part of the radiation therapy machine Therac-25 caused the death of six cancer patients between 1985 and 1987 as they were exposed to an overdose of radiation.

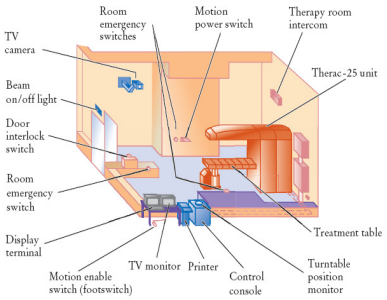


Figure 1.2: Therac-25 Facility.

In 1995 the American Airlines Flight 965 connecting Miami and Cali crashed just five minutes before its scheduled arrival. The accident led to a total of 159 deaths. Paris Kanellakis, a well known researcher (creator of the partition refinement algorithm, broadly used to verify bisimulation), was in the flight together with his family. Investigations concluded that the accident was originated by a sudden turn of the aircraft caused by the autopilot after an instruction of one of the pilots: the pilot input 'R' in the navigational computer referring to a location called 'Rozo' but the computer erroneously interpreted it as a location called 'Romeo' (due to the spelling similarity and physical proximity of the locations).

As the use and complexity of technological devices grow quickly, mechanisms to improve their correctness have become unavoidable. But, how can we be sure of the correctness of such technologies, with thousands (and sometimes, millions) of components interacting in complex ways? One possible answer is by using *formal verification*, a branch of formal methods.

1.2.2 Formal verification

Formal verification is considered a fundamental area of study in computer science. In the context of hardware and software systems, formal verification is the act of proving or disproving the correctness of the system with respect to a certain property, using formal methods. In order to achieve this, it is necessary to construct a mathematical model describing all possible behaviors of the system. In addition, the property must be formally specified avoiding, in this way, possible ambiguities.

Important formal verification techniques include theorem proving, simulation, testing, and model checking. In this thesis we focus on the use of this last technique.

Model checking Model checking is an automated verification technique that, given a finite model of the system and a formal property, systematically checks whether the property holds in the model or not. In addition, if the property is falsified, debugging information is provided in the form of a *counterexample*. This situation is represented in Figure 1.3.

Usual properties that can be verified are "Can the system reach a deadlock state?", or "Every sent message is received with probability at least 0.99?". Such automated verification is carried on by a so-called *model checker*, an algorithm that exhaustively searches the space state of the model looking for states violating the (correctness) property.

A major strength of model checking is the capability of generating counterexamples which provide diagnostic information in case the property is violated. Edmund M. Clarke, one of the pioneers of Model Checking said [Cla08]: *"It is impossible to overestimate the importance of the counterexample feature. The counterexamples are invaluable in debugging complex systems. Some people use model checking just for this feature"*. In case a state violating the property under consideration is encountered, the model checker provides a counterexample describing a possible execution that leads from the initial state of the system to a violating state.

Other important advantages of model checking are: it is highly automatic so it requires little interaction and knowledge of designers, it is

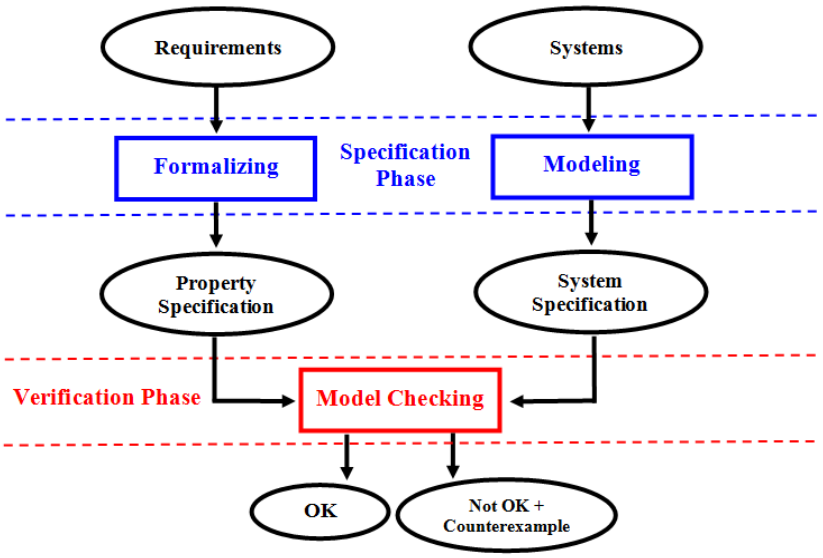


Figure 1.3: Schematic view of model-checking approach

rather fast, it can be applied to a large range of problems, it allows partial specifications.

The main disadvantage of model checking is that the space state of certain systems, for instance distributed systems, can be rather large, thus making the verifications inefficient and in some cases even unfeasible (because of memory limitations). This problem is known as the *state explosion problem*. Many techniques to alleviate it have been proposed since the invention of model checking. Among the most popular ones we mention the use Binary Decision Diagrams (BDDs), partial order reduction, abstraction, compositional reasoning, and symmetry reduction. State-of-the-art model checkers can easily handle up to 10^9 states with explicit state representation. For certain specific problems, more dedicated data structures (like BDDs) can be used thus making it possible to handle even up to 10^{476} states.

The popularity of model checking has grown considerably since its in-

vention at the beginning of the 80s. Nowadays, model checking techniques are employed by most or all leading hardware companies (e.g. INTEL, IBM and MOTOROLA - just to mention few of them). While model checking is applied less frequently by software developing companies, there have been several cases in which it has helped to detect previously unknown defects in real-world software. A prominent example is the result of research in Microsoft's SLAM project in which several formal techniques were used to automatically detect flaws in device drivers. In 2006, Microsoft released the Static Driver Verifier as part of Windows Vista, SDV uses the SLAM software-model-checking engine to detect cases in which device drivers linked to Vista violate one of a set of interface rules. Thus SDV helps uncover defects in device drivers, a primary source of software bugs in Microsoft applications. Investigations have shown that model checking procedures would have revealed the exposed defects in, e.g., Intels Pentium II processor and the Therac-25 therapy radiation machine.

Focus of this thesis This thesis addresses the foundational aspects of formal methods for applications in security and in particular in anonymity: We investigate various issues that have arisen in the area of anonymity, we develop frameworks for the specification of anonymity properties, and we propose algorithms for their verification.

1.3 Background

In this section we give a brief overview of the various approaches to the foundations of anonymity that have been explored in the literature. We will focus on anonymity properties, although the concepts and techniques developed for anonymity apply to a large extent also to neighbor topics like *information flow*, *secrecy*, *privacy*. The common denominator of these problems is the prevention of the leakage of information. More precisely, we are concerned with situations in which there are certain values (data, identities, actions, etc) that are intended to be secret, and we want to ensure that an adversary will not be able to infer the secret values from

the information which is publicly available. Some researchers use the term *information hiding* to refer to this class of problems [HO05].

The frameworks for reasoning about anonymity can be classified into two main categories: the *possibilistic* approaches, and the *probabilistic* (or *quantitative*) ones.

Possibilistic notions

The term "possibilistic" refers to the fact that we do not consider quantitative aspects. More precisely, anonymity is formulated in terms of the possibility or inferring some secrets, without worrying about "how likely" this is, or "how much" we narrow down the secret.

These approaches have been widely explored in the literature, using different conceptual frameworks. Examples include the proposals based on epistemic logic ([SS99, HO05]), on "function views" ([HS04]), and on process equivalences (see for instance [SS96, RS01]). In the following we will focus on the latter kind.

In general, possibilistic anonymity means that the observables do not identify a unique culprit. Often this property relies on *nondeterminism*: for each culprit, the system should be able to produce alternative executions with different observables, so that in turn for a given observable there are many agents that could be the culprit. More precisely, in its strongest version this property can be expressed as follows: if in one computation the identity of the culprit is i and the observable outcome is o, then for every other agent j there must be a computation where, with culprit j, the observable is still o.

This kind of approach can be applied also in case of systems that use randomization. The way this is done is by abstracting the probabilistic choices into nondeterministic ones. See for example the Dining Cryptographers example in [SS96], where the coin tossing is represented by a nondeterministic process.

In general the possibilistic approaches have the advantages of simplicity an efficiency. On the negative side, they lack precision, and in some cases the approximation can be rather loose. This is because every scenario that

has a non-null probability is interpreted as possible. For instance, consider the case in which a system reveals the culprit 90 percent of the times by outputting his identity, while in the remaining 10 percent of the times it outputs the name of some other agent. The system would not look very anonymous. Yet, the possibilistic definition of anonymity would be satisfied because all users would appear as possible culprits to the observer regardless of the output of the system. In general, in the possibilistic approach the strongest notion of anonymity we can express is *possible innocence*, which is satisfied when no agent appear to be the culprit *for sure*: there is always the possibility that he is innocent (no matter how unlikely it is).

In this thesis we consider only the probabilistic approaches. Their common feature is that they deal with probabilities in a concrete way and they are, therefore, much more precise. They have become very popular in recent times, and there has been a lot of work dedicated to understanding and formalizing the notion in a rigorous way. In the next section we give a brief overview of these efforts.

Probabilistic notions

These approaches take probabilities into account, and are based on the likelihood that an agent is the culprit, for a given observable. One notion of probabilistic anonymity which has been thoroughly investigated in the literature is *strong anonymity*.

Strong anonymity Intuitively the idea behind this notion is that the observables should not allow to infer any (quantitative) information about the identity of the culprit. The corresponding notion in the field of information flow is (quantitative) non-interference.

Once we try to formalize more precisely the above notion we discover however that there are various possibilities. Correspondingly, there have been various proposals. We recall here the three most prominent ones.

1. *Equality of the a posteriori probabilities for different culprits.* The idea is to consider a system strongly anonymous if, given an observ-

able o, the *a posteriori* probability that the identity of the culprit is i, $\mathbb{P}(i|o)$, is the same as the *a posteriori* probability of any other identity j. Formally:

$$\mathbb{P}(i|o) = \mathbb{P}(j|o) \quad \text{for all observables } o, \text{ and all identities } i \text{ and } j \tag{1.1}$$

This is the spirit of the definition of *strong anonymity* by Halpern and O'Neill [HO05], although their formalization involves more sophisticated epistemic notions.

2. *Equality of the a posteriori and a priori probabilities for the same culprit.* Here the idea is to consider a system strongly anonymous if, for any observable, the *a posteriori* probability that the culprit is a certain agent i is the same as its *a priori* probability. In other words, the observation does not increase or decrease the support for suspecting a certain agent. Formally:

$$\mathbb{P}(i|o) = \mathbb{P}(i) \quad \text{for all observables } o, \text{ and all identities } i \tag{1.2}$$

This is the definition of *anonymity* adopted by Chaum in [Cha88]. He also proved that the Dining Cryptographers satisfy this property if the coins are fair. Halpern and O'Neill consider a similar property in their epistemological setting, and they call it *conditional anonymity* [HO05].

3. *Equality of the likelihood of different culprits.* In this third definition a system is strongly anonymous if, for any observable o and agent i, the *likelihood* of i being the culprit, namely $\mathbb{P}(o|i)$, is the same as the likelihood of any other agent j. Formally:

$$\mathbb{P}(o|i) = \mathbb{P}(o|j) \quad \text{for all observables } o, \text{ and all identities } i \text{ and } j \tag{1.3}$$

This was proposed as definition of *strong anonymity* by Bhargava and Palamidessi [BP05].

In [BCPP08] it has been proved that definitions (1.2) and (1.3) are equivalent. Definition (1.3) has the advantage that it does extend in a natural way to the case in which the choice of the culprit is nondeterministic.

This could be useful when we do not know the a priori distribution of the culprit, or when we want to abstract from it (for instance because we are interested in the worst case).

Concerning Definition (1.1), it probably looks at first sight the most natural, but it actually turns out to be way too strong. In fact it is equivalent to (1.2) and (1.3), *plus* the following condition:

$$\mathbb{P}(i) = \mathbb{P}(j) \qquad \text{for all identities } i \text{ and } j \tag{1.4}$$

namely the condition that the a priori distribution be uniform.

It is interesting to notice that (1.1) can be split in two orthogonal properties: (1.3), which depends only in the protocol, and (1.4), which depends only in the distribution on the secrets.

Unfortunately all the strong anonymity properties discussed above are too strong, almost never achievable in practice. Hence researches have started exploring weaker notions. One of the most renowned properties of this kind (among the "simple" ones based on conditional probabilities) is that of *probable innocence*.

Probable innocence The notion of *probable innocence* was formulated by Rubin and Reiter in the context of their work on the Crowds protocol [RR98]. Intuitively the idea is that, after the observation, no agent is more likely to be the culprit than not to be. Formally:

$$\mathbb{P}(i|o) \leq \mathbb{P}(\neg i|o) \quad \text{for all observations o, and all identities } i$$

or equivalently

$$\mathbb{P}(i|o) \leq \frac{1}{2} \quad \text{for all observations o, and all identities } i$$

In [RR98] Rubin and Reiter proved that the Crowds protocol satisfies probable innocence under a certain assumption on the number of attackers relatively to the number of honest users.

All the notions discussed above are rather coarse, in the sense that they are cut-off notions and do not allow to represent small variations in the

degree of anonymity. In order to be able to compare different protocols in a more precise way, researcher have started exploring settings to measure the *degree of anonymity*. The most popular of these approaches are those based in information theory.

Information theory

The underlying idea is that anonymity systems are interpreted as *channels* in the information-theoretic sense. The input values are the possible identities of the culprit, which, associated to a probability distribution, form a random variable Id. The outputs are the observables, and the transition matrix consists of the conditional probabilities of the form $\mathbb{P}(o|i)$, representing the probability that the system produces an observable o when the culprit is i. A central notion here is the Shannon entropy, which represents the uncertainty of a random variable. For the culprit's possible identity, this is given by:

$$H(Id) = -\sum_i \mathbb{P}(i) \log \mathbb{P}(i) \quad \text{(uncertainty a priori)}$$

Note that Id and the matrix also determine a probability distribution on the observables, which can then be seen as another random variable Ob. The *conditional entropy* $H(Id|Ob)$, representing the uncertainty about the identity of the culprit *after* the observation, is given by

$$H(Id|Ob) = -\sum_o \mathbb{P}(o) \sum_i \mathbb{P}(i|o) \log \mathbb{P}(i|o) \quad \text{(uncertainty a posteriori)}$$

It can be shown that $0 \leq H(Id|Ob) \leq H(Id)$. We have $H(Id|Ob) = 0$ when there is no uncertainty left about Id after the value of Ob is known. Namely, when the value of Ob completely determines the value of Id. This is the case of maximum leakage. At the other extreme, we have $H(Id|Ob) = H(Id)$ when Ob gives no information about Id, i.e. when Ob and Id are independent.

The difference between $H(Id)$ and $H(Id|Ob)$ is called *mutual information* and it is denoted by $I(Id; Ob)$:

$$I(Id; Ob) = H(Id) - H(Id|Ob)$$

The maximum mutual information between Id and Ob over all possible input distributions $\mathbb{P}_{Id}(\cdot)$ is known as the channel's *capacity*:

$$C = \max_{\mathbb{P}_{Id}(\cdot)} I(Id; Ob)$$

In the case of anonymity, the mutual information represents the difference between the a priori and the a posteriori uncertainties about the identity of the culprit. It can therefore be considered as the leakage of information due to the system, i.e. the amount of anonymity which is lost because of the observables produced by the system. Similarly, the capacity represents the worst-case leakage under all possible distributions on the culprit's possible identities. It can be shown that the capacity is 0 if and only if the rows of the matrix are pairwise identical. This corresponds exactly to the version (1.3) of strong anonymity.

This view of the degree of anonymity has been advocated in various works, including [MNCM03, MNS03, ZB05, CPP08a]. In the context of information flow, the same view of leakage in information theoretic terms has been widely investigated as well. Without pretending to be exhaustive, we mention [McL90, Gra91, CHM01, CHM05a, Low02, Bor06].

In [Smi09] Smith has investigated the use of an alternative notion of entropy, namely Rényi's min entropy [Rén60], and has proposed to define leakage as the analogous of mutual information in the setting of Rényi's min entropy. The justification for proposing this variant is that it represents better certain attacks called *one-try attacks*. In general, as Köpf and Basin illustrate in their cornerstone paper [KB07], one can use the above information-theoretic approach with many different notions of entropy, each representing a different model of attacker, and a different way of measuring the success of an attack.

A different information-theoretic approach to leakage has been proposed in [CMS09]: in that paper, the authors define as information leakage the difference between the a priori accuracy of the guess of the attacker, and the a posteriori one, after the attacker has made his observation. The accuracy of the guess is defined as the Kullback-Leibler distance between the *belief* (which is a weight attributed by the attacker to each input hypothesis) and

the true distribution on the hypotheses. In [HSP10] a Rényi's min entropy variant of this approach has been explored as well.

We conclude this section by remarking that, in all the approaches discussed above, the notion of conditional probability plays a central role.

1.4 Contribution and plan of the thesis

We have seen in Section 1.3 that conditional probabilities are the key ingredients of all quantitative definitions of anonymity. It is therefore desirable to develop techniques to analyze and compute such probabilities.

Our first contribution is cpCTL, a temporal logic allowing us to specify properties concerned with conditional probabilities in systems combining probabilistic and nondeterministic behavior. This is presented in Chapter 2. cpCTL is essentially pCTL (probabilistic Computational Tree Logic) [HJ94] enriched with formulas of the kind $\mathbb{P}_{\leq a}[\phi|\psi]$, stating that the probability of ϕ given ψ is at most a. We do so by enriching pCTL with formulas of the form $\mathbb{P}_{\bowtie a}[\phi|\psi]$. We propose a model checker for cpCTL. Its design has been quite challenging, due to the fact that the standard model checking algorithms for pCTL in MDPs (Markov Decision Processes) do not extend to conditional probability formulas. More precisely, in contrast to pCTL, verifying a conditional probability cannot be reduced to a linear optimization problem. A related point is that, in contrast to pCTL, the optimal probabilities are not attained by history independent schedulers. We attack the problem by proposing the notion of *semi history independent schedulers*, and we show that these schedulers do attain optimality with respect to the conditional probabilities. Surprisingly, it turns out that we can further restrict to deterministic schedulers, and still attain optimality. Based on these results, we show that it is decidable whether a cpCTL formula is satisfied in a MDP, and we provide an algorithm for it. In addition, we define the notion of counterexample for the logic and sketch an algorithm for counterexample generation.

Unfortunately, the verification of conditional cpCTL formulae is not efficient in the presence of nondeterminism. Another issue, related to nonde-

terminism within the applications in the field of security, is the well known problem of almighty schedulers (see Chapter 4). Such schedulers have the (unrealistic) ability to peek on internal secrets of the component and make their scheduling policy dependent on these secrets, thus leaking the secrets to external observers. We address these problems in separate chapters.

In Chapter 3 we restrict the framework to purely probabilistic models where secrets and observables do not interact, and we consider the problem of computing the leakage and the maximal leakage in the information-theoretic approach. These are defined as mutual information and capacity, respectively. We address these notions with respect to both the Shannon entropy and the Rényi min entropy. We provide techniques to compute channel matrices in $O((o \times q)^3)$ time, where o is the number of observables, and q the number of states. (From the channel matrices, we can compute mutual information and capacity using standard techniques.) We also show that, contrarily to what was stated in literature, the standard information theoretical approaches to leakage do not extend to the case in which secrets and observable interact.

In Chapter 4 we consider the problem of the almighty schedulers. We define a restricted family of schedulers (*admissible schedulers*) which cannot base their decisions on secrets, thus providing more realistic notions of strong anonymity than arbitrary schedulers. We provide a framework to represent concurrent systems composed by purely probabilistic components. At the global level we still have nondeterminism, due to the various possible ways the component may interact with each other. Schedulers are then defined as devices that select at every point of the computation the component(s) moving next. Admissible schedulers make this choice independently from the values of the secrets. In addition, we provide a sufficient (but not necessary) technique based on automorphisms to prove strong anonymity for this family of schedulers.

The notion of counterexample has been approached indirectly in Chapters 2 and 3. In Chapter 5 we come back and fully focus on this topic. We propose a novel technique to generate counterexamples for model checking on Markov Chains. Our propose is to group together violating paths that are likely to provide similar debugging information thus alleviating the de-

bugging tasks. We do so by using strongly connected component analysis and show that it is possible to extend these techniques to Markov Decision Processes.

Chapter 6 is an overview chapter. There we briefly describe extensions to the frameworks presented in Chapters 3 and 4[1]. First, we consider the case in which secrets and observables interact, and show that it is still possible to define an information-theoretic notion of leakage, provided that we consider a more complex notion of channel, known in literature as *channel with memory and feedback*. Second, we extend the systems proposed in Chapter 4 by allowing nondeterminism also internally to the components. Correspondingly, we define a richer notion of admissible scheduler and we use it for defining notion of process equivalences relating to nondeterminism in a more flexible way than the standard ones in the literature. In particular, we use these equivalences for defining notions of anonymity robust with respect to implementation refinement.

In Figure 1.4 we describe the relation between the different chapters of the thesis. Chapter 5 is not explicitly depicted in the figure because it does not fit in any of the branches of cpCTL (efficiency - security foundations). However, the techniques developed in Chapter 5 have been applied to the works in both Chapters 2 and 3.

We conclude this thesis In Chapter 7, there we present a summary of our main contributions and discuss further directions.

1.5 Origins of the Chapters and Credits

In the following we list, for each chapter, the set of related articles together with their publication venue and corresponding co-authors.

- Chapter 2 is mainly based on the article [AvR08] by Peter van Rossum and myself. The article was presented in TACAS 2008. In addition, this chapter contains material of an extended version of [AvR08] that is being prepared for submission to a journal.

[1]For more information about the topics discussed in this chapter we refer the reader to [AAP10a, AAP11, AAP10b, AAPvR10].

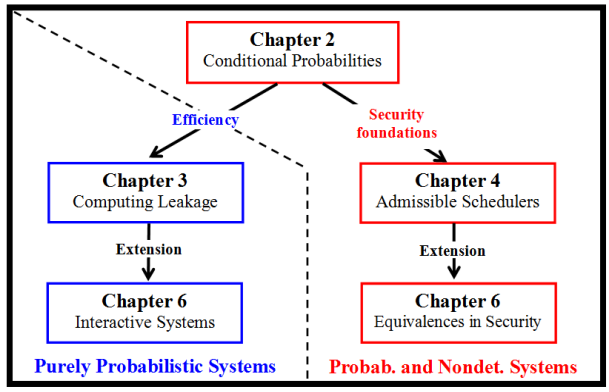


Figure 1.4: Chapters relation.

- Chapter 3 is based on the article [APvRS10a] by Catuscia Palamidessi, Peter van Rossum, Geoffrey Smith and myself. The article was presented in TACAS 2010.
- Chapter 4 is based on
  - The article [APvRS10b] by Catuscia Palamidessi, Peter van Rossum, Ana Sokolova and myself. This article was presented in QEST 2010.
  - The journal article [APvRS11] by the same authors.
- Chapter 5 is based on the article [ADvR08] by Pedro D'Argenio, Peter van Rossum, and myself. The article was presented in HVC 2008.
- Chapter 6 is based on
  - The article [AAP10b] by Mário S. Alvim, Catuscia Palamidessi, and myself. This work was presented in LICS 2010 as part of an invited talk by Catuscia Palamidessi.
  - The article [AAP10a] by Mário S. Alvim, Catuscia Palamidessi, and myself. This work presented in CONCUR 2010.
  - The journal article [AAP11] by the same authors of the previous works.

– The article [AAPvR10] by Mário S. Alvim, Catuscia Palamidessi, Peter van Rossum, and myself. This work was presented in IFIP-TCS 2010.

The chapters remain close to their published versions, thus there is inevitably some overlapping between them (in particular in their introductions where basic notions are explained).

A short note about authorship: I am the first author in all the articles and journal works included in this thesis with the exception of the ones presented in Chapter 6.

Chapter 2

Conditional Probabilities over Probabilistic and Nondeterministic Systems

In this chapter we introduce cpCTL*, a logic which extends the probabilistic temporal logic pCTL with conditional probabilities allowing to express statements of the form "the probability of ϕ given ψ is at most a". We interpret* cpCTL *over Markov Chains and Markov Decision Processes. While model checking* cpCTL *over Markov Chains can be done with existing techniques, those techniques do not carry over to Markov Decision Processes. We study the class of schedulers that suffice to find the maximum and minimum conditional probabilities, show that the problem is decidable for Markov Decision Processes and propose a model checking algorithm. Finally, we present the notion of counterexamples for* cpCTL *model checking and provide a method for counterexample generation.*

2.1 Introduction

Conditional probabilities are a fundamental concept in probability theory. In system validation these appear for instance in anonymity, risk assessment, and diagnosability. Typical examples here are: the probability that a certain message was sent by Alice, given that an intruder observes a certain traffic pattern; the probability that the dykes break, given that it rains heavily; the probability that component A has failed, given error message E.

In this chapter we introduce cpCTL (conditional probabilistic CTL), a logic which extends strictly the probabilistic temporal logic pCTL [HJ89] with new probabilistic operators of the form $\mathbb{P}_{\leq a}[\phi|\psi]$. Such formula means that the probability of ϕ given ψ is at most a. We interpret cpCTL formulas over Markov Chains (MCs) and Markov Decision Processes (MDPs). Model checking cpCTL over MCs can be done with model checking techniques for pCTL*, using the equality $\mathbb{P}[\phi|\psi] = \mathbb{P}[\phi \wedge \psi]/\mathbb{P}[\psi]$.

In the case of MDPs, cpCTL model checking is significantly more complex. Writing $\mathbb{P}_\eta[\phi|\psi]$ for the probability $\mathbb{P}[\phi|\psi]$ under scheduler η, model checking $\mathbb{P}_{\leq a}[\phi|\psi]$ reduces to computing $\mathbb{P}^+[\phi|\psi] = \mathsf{max}_\eta\, \mathbb{P}_\eta[\phi|\psi] = \mathsf{max}_\eta\, \mathbb{P}_\eta[\phi \wedge \psi]/\mathbb{P}_\eta[\psi]$. Thus, we have to maximize a non-linear function. (Note that in general $\mathbb{P}^+[\phi|\psi] \neq \mathbb{P}^+[\phi \wedge \psi]/\mathbb{P}^+[\psi]$.) Therefore, we cannot reuse the efficient techniques for pCTL model checking, since they heavily rely on linear optimization techniques [BdA95].

In particular we show that, differently from what happens in pCTL [BdA95], history independent schedulers are not sufficient for optimizing conditional reachability properties. This is because in cpCTL the optimizing schedulers are not determined by the local structure of the system. That is, the choices made by the scheduler in one branch may influence the optimal choices in other branches. We introduce the class of semi history-independent schedulers and show that these suffice to attain the optimal conditional probability. Moreover, deterministic schedulers still suffice to attain the optimal conditional probability. This is surprising since many non-linear optimization problems attain their optimal value in the interior of a convex polytope, which correspond to randomized schedulers in our

setting.

Based on these properties, we present an (exponential) algorithm for checking whether a given system satisfies a formula in the logic. Furthermore, we define the notion of counterexamples for cpCTL model checking and provide a method for counterexample generation.

To the best of our knowledge, our proposal is the first temporal logic dealing with the specification and verification of conditional probabilities.

Applications

Complex Systems. One application of the techniques presented in this chapter is in the area of complex system behavior. We can model the probability distribution of natural events as probabilistic choices, and the operator choices as non-deterministic choices. The computation of maximum and minimum conditional probabilities can then help to optimize run-time behavior. For instance, suppose that the desired behavior of the system is expressed as a pCTL formula ϕ and that during run-time we are making an observation about the system, expressed as a pCTL formula ψ. The techniques developed in this chapter allow us to compute the maximum probability of ϕ given ψ and to identify the actions (non-deterministic choices) that have to be taken to achieve this probability.

Anonymizing Protocols. Another application is in the area of anonymizing protocols. The purpose of these protocols is to hide the identity of the user performing a certain action. Such a user is usually called the *culprit*. Examples of these protocols are Onion Routing [CL05], Dining Cryptographers [Cha88], Crowds [RR98] and voting protocols [FOO92] (just to mention a few). Strong anonymity is commonly formulated [Cha88, BP05] in terms of conditional probability: A protocol is considered strongly anonymous if no information about the culprit's identity can be inferred from the behavior of the system. Formally, this is expressed by saying that culprit's identity and the observations, seen as random variables, are independent from each other. That is to say, for all users u and all observations of the adversary o:

$$\mathbf{P}[\text{culprit} = u \mid \text{observation} = o] = \mathbf{P}[\text{culprit} = u].$$

If considering a concurrent setting, it is customary to give the adversary full control over the network [DY83] and model its capabilities as nondeterministic choices in the system, while the user behavior and the random choices in the protocol are modeled as probabilistic choices. Since anonymity should be guaranteed for all possible attacks of the adversary, the above equality should hold for all schedulers. That is: the system is strongly anonymous if for all schedulers η, all users u and all adversarial observations o:

$$\mathbf{P}_\eta[\text{culprit} = u \mid \text{observation} = o] = \mathbf{P}_\eta[\text{culprit} = u]$$

Since the techniques in this chapter allow us to compute the maximal and minimal conditional probabilities over all schedulers, we can use them to prove strong anonymity in presence of nondeterminism.

Similarly, probable innocence means that a user is not more likely to be innocent than not to be (where "innocent" mans "not the culprit"). In cpCTL this can be expressed as $\mathbb{P}_{\leq 0.5}[\text{culprit} = u \,|\, \text{observations} = o]$.

Organization of the chapter In Section 2.2 we present the necessary background on MDPs. In Section 2.3 we introduce conditional probabilities over MDPs and in Section 2.4 we introduce cpCTL. Section 2.5 introduces the class of semi history-independent schedulers and Section 2.6 explains how to compute the maximum and minimum conditional probabilities. Finally, in Section 2.7 we investigate the notion of counterexamples.

2.2 Markov Decision Processes

Markov Decision Processes constitute a formalism that combines nondeterministic and probabilistic choices. They are a dominant model in corporate finance, supply chain optimization, and system verification and optimization. While there are many slightly different variants of this formalism (e.g., action-labeled MDPs [Bel57, FV97], probabilistic automata [SL95, SdV04]), we work with the state-labeled MDPs from [BdA95].

The set of all discrete probability distributions on a set S is denoted by $\mathsf{Distr}(S)$. The Dirac distribution on an element $s \in S$ is written as 1_s. We also fix a set $\mathcal{P}$ of propositions.

Definition 2.2.1. A *Markov Decision Process* (MDP) is a four-tuple $\Pi = (S, s_0, \tau, L)$ where: S is the finite state space of the system, $s_0 \in S$ is the initial state, $L \colon S \to \wp(\mathcal{P})$ is a labeling function that associates to each state $s \in S$ a subset of propositions, and $\tau \colon S \to \wp(\mathsf{Distr}(S))$ is a function that associates to each $s \in S$ a non-empty and finite subset of of successor distributions.

In case $|\tau(s)| = 1$ for all states s we say that Π is a *Markov Chain*. We define the *successor* relation $\varrho \subseteq S \times S$ by $\varrho \triangleq \{(s,t) \mid \exists \pi \in \tau(s) \,.\, \pi(t) > 0\}$ and for each state $s \in S$ we define the sets $\mathsf{Paths}(s_0) \triangleq \{s_0 s_1 s_2 \ldots \in S^\omega \mid s_0 = s \wedge \forall n \in \mathbb{N} \,.\, \varrho(s_n, s_{n+1})\}$, and $\mathsf{Paths}^\star(s) \triangleq \{s_0 s_1 \ldots s_n \in S^\star \mid s_0 = s \wedge \forall\, 0 \leq i < n \,.\, \varrho(s_n, s_{n+1})\}$ of paths and finite paths respectively beginning at s. Sometimes we will use $\mathsf{Paths}(\Pi)$ to denote $\mathsf{Paths}(s_0)$, i.e. the set of paths of Π. For $\omega \in \mathsf{Paths}(s)$, we write the n-th state of ω as ω_n. In addition, we write $\sigma_1 \sqsubseteq \sigma_2$ if σ_2 is an extension of σ_1, i.e. $\sigma_2 = \sigma_1 \sigma'$ for some σ'. We define the basic cylinder of a finite path σ as the set of (infinite) paths that extend it, i.e $\langle \sigma \rangle \triangleq \{\omega \in \mathsf{Paths}(s) \mid \sigma \sqsubseteq \omega\}$. For a set of paths R we write $\langle R \rangle$ for its set of cylinders, i.e. $\langle R \rangle \triangleq \bigcup_{\sigma \in R} \langle \sigma \rangle$. As usual, we let $\mathcal{B}_s \subseteq \wp(\mathsf{Paths}(s))$ be the Borel σ-algebra on the basic cylinders.

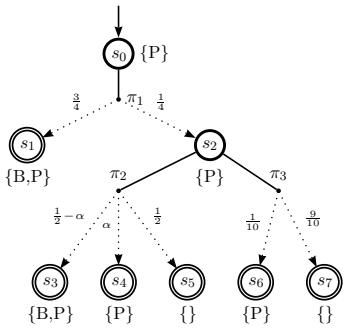


Figure 2.1: MDP

Example 2.2.2. Figure 2.1 shows a MDP. States with double lines represent absorbing states (i.e., states s with $\tau(s) = \{1_s\}$) and α is any constant in the interval $[0, \frac{1}{2}]$. This MDP features a single nondeterministic decision, to be made in state s_2.

Schedulers (also called strategies, adversaries, or policies) resolve the nondeterministic choices in a MDP [PZ93, Var85, BdA95].

Definition 2.2.3. Let $\Pi = (S, s_0, \tau, L)$ be a MDP and $s \in S$. An *s-scheduler* η for Π is a function from $\mathsf{Paths}^\star(s)$ to $\mathsf{Distr}(\wp(\mathsf{Distr}(S)))$ such that for all $\sigma \in \mathsf{Paths}^\star(s)$ we have $\eta(\sigma) \in \mathsf{Distr}(\tau(\mathsf{last}(\sigma)))$. We denote the set of all s-schedulers on Π by $\mathsf{Sch}_s(\Pi)$. When $s = s_0$ we omit it.

Note that our schedulers are randomized, i.e., in a finite path σ a scheduler chooses an element of $\tau(\mathsf{last}(\sigma))$ probabilistically. Under a scheduler η, the probability that the next state reached after the path σ is t, equals $\sum_{\pi \in \tau(\mathsf{last}(\sigma))} \eta(\sigma)(\pi) \cdot \pi(t)$. In this way, a scheduler induces a probability measure on $\mathcal{B}_s$ defined as follows:

Definition 2.2.4. Let Π be a MDP, $s \in S$, and η an s-scheduler on Π. The probability measure $\mathbb{P}_{s,\eta}$ is the unique measure on $\mathcal{B}_s$ such that for all $s_0 s_1 \dots s_n \in \mathsf{Paths}^\star(s)$

$$\mathbb{P}_{s,\eta}(\langle s_0 s_1 \dots s_n \rangle) \triangleq \prod_{i=0}^{n-1} \sum_{\pi \in \tau(s_i)} \eta(s_0 s_1 \dots s_i)(\pi) \cdot \pi(s_{i+1}).$$

Often we will write $\mathbb{P}_\eta(\Delta)$ instead of $\mathbb{P}_{s,\eta}(\Delta)$ when s is the initial state and $\Delta \in \mathcal{B}_s$. We now recall the notions of deterministic and history independent schedulers.

Definition 2.2.5. Let Π be a MDP, $s \in S$, and η an s-scheduler for Π. We say that η is *deterministic* if $\eta(\sigma)(\pi)$ is either 0 or 1 for all $\pi \in \tau(\mathsf{last}(\sigma))$ and all $\sigma \in \mathsf{Paths}^\star(s)$. We say that a scheduler is *history independent* (HI) if for all finite paths σ_1, σ_2 of Π with $\mathsf{last}(\sigma_1) = \mathsf{last}(\sigma_2)$ we have $\eta(\sigma_1) = \eta(\sigma_2)$.

Definition 2.2.6. Let Π be a MDP, $s \in S$, and $\Delta \in \mathcal{B}_s$. Then the *maximal and minimal probabilities of* Δ, $\mathbb{P}^+_s(\Delta), \mathbb{P}^-_s(\Delta)$, are defined as

$$\mathbb{P}^+_s(\Delta) \triangleq \sup_{\eta \in \mathsf{Sch}_s(\Pi)} \mathbb{P}_{s,\eta}(\Delta) \quad \text{and} \quad \mathbb{P}^-_s(\Delta) \triangleq \inf_{\eta \in \mathsf{Sch}_s(\Pi)} \mathbb{P}_{s,\eta}(\Delta).$$

A scheduler that attains $\mathbb{P}^+_s(\Delta)$ or $\mathbb{P}^-_s(\Delta)$ is called an *optimizing* scheduler.

We now define the notion of (finite) convex combination of schedulers.

Definition 2.2.7. Let Π be a MDP and let $s \in S$. An s-scheduler η is a *convex combination* of the s-schedulers $\eta_1, \ldots, \eta_n$ if there are $\alpha_1, \ldots, \alpha_n \in [0,1]$ with $\alpha_1 + \cdots + \alpha_n = 1$ such that for all $\Delta \in \mathcal{B}_s$, $\mathbb{P}_{s,\eta}(\Delta) = \alpha_1 \mathbb{P}_{s,\eta_1}(\Delta) + \cdots + \alpha_n \mathbb{P}_{s,\eta_n}(\Delta)$.

Note that taking the convex combination η of η_1 and η_2 as functions, i.e., $\eta(\sigma)(\pi) = \alpha\eta_1(\sigma)(\pi) + (1-\alpha)\eta_2(\sigma)(\pi)$, does not imply that η is a convex combination of η_1 and η_2 in the sense above.

2.3 Conditional Probabilities over MDPs

The conditional probability $P(A \mid B)$ is the probability of an event A, given the occurrence of another event B. Recall that given a probability space (Ω, F, P) and two events $A, B \in F$ with $P(B) > 0$, $P(A \mid B)$ is defined as $P(A \cap B)/P(B)$. If $P(B) = 0$, then $P(A \mid B)$ is undefined. In particular, given a MDP Π, a scheduler η, and a state s, consider the probabilistic space $(\mathsf{Paths}(s), \mathcal{B}_s, \mathbb{P}_{s,\eta})$. For two sets of paths $\Delta_1, \Delta_2 \in \mathcal{B}_s$ with $\mathbb{P}_{s,\eta}(\Delta_2) > 0$, the conditional probability of Δ_1 given Δ_2 is $\mathbb{P}_{s,\eta}(\Delta_1 \mid \Delta_2) = \mathbb{P}_{s,\eta}(\Delta_1 \cap \Delta_2)/\mathbb{P}_{s,\eta}(\Delta_2)$. If $\mathbb{P}_{s,\eta}(\Delta_2) = 0$, then $\mathbb{P}_{\eta,s}(\Delta_1 \mid \Delta_2)$ is undefined. We define the maximum and minimum conditional probabilities for all $\Delta_2 \in \mathcal{B}_s$ as follows:

Definition 2.3.1. Let Π be a MDP. The *maximal and minimal conditional probabilities* $\mathbb{P}^+_s(\Delta_1|\Delta_2)$, $\mathbb{P}^-_s(\Delta_1|\Delta_2)$ of sets of paths $\Delta_1, \Delta_2 \in \mathcal{B}_s$ are defined by

$$\mathbb{P}^+_s(\Delta_1|\Delta_2) \triangleq \begin{cases} \sup_{\eta \in \mathsf{Sch}^{>0}_{\Delta_2}} \mathbb{P}_{s,\eta}(\Delta_1|\Delta_2) & \text{if } \mathsf{Sch}^{>0}_{\Delta_2} \neq \emptyset, \\ 0 & \text{otherwise,} \end{cases}$$

$$\mathbb{P}^-_s(\Delta_1|\Delta_2) \triangleq \begin{cases} \inf_{\eta \in \mathsf{Sch}^{>0}_{\Delta_2}} \mathbb{P}_{s,\eta}(\Delta_1|\Delta_2) & \text{if } \mathsf{Sch}^{>0}_{\Delta_2} \neq \emptyset, \\ 1 & \text{otherwise,} \end{cases}$$

where $\mathsf{Sch}^{>0}_{\Delta_2} = \{\eta \in \mathsf{Sch}_s(\Pi) \mid \mathbb{P}_{s,\eta}(\Delta_2) > 0\}$.

The following lemma generalizes Lemma 6 of [BdA95] to conditional probabilities.

Lemma 2.3.2. Given $\Delta_1, \Delta_2 \in \mathcal{B}_s$, its maximal and minimal conditional probabilities are related by: $\mathbb{P}^+_s(\Delta_1|\Delta_2) = 1 - \mathbb{P}^-_s(\mathsf{Paths}(s) - \Delta_1|\Delta_2)$.

2.4 Conditional Probabilistic Temporal Logic

The logic cpCTL extends pCTL with formulas of the form $\mathbb{P}_{\bowtie a}[\phi|\psi]$ where $\bowtie \in \{<, \leq, >, \geq\}$. Intuitively, $\mathbb{P}_{\leq a}[\phi|\psi]$ holds if the probability of ϕ given ψ is at most a. Similarly for the other comparison operators.

Syntax: The cpCTL logic is defined as a set of state and path formulas, i.e., cpCTL $\triangleq \mathsf{Stat} \cup \mathsf{Path}$, where Stat and Path are defined inductively:

$$\begin{array}{rcl} \mathcal{P} & \subseteq & \mathsf{Stat}, \\ \phi, \psi \in \mathsf{Stat} & \Rightarrow & \phi \wedge \psi, \neg\phi \in \mathsf{Stat}, \\ \phi, \psi \in \mathsf{Path} & \Rightarrow & \mathbb{P}_{\bowtie a}[\phi], \mathbb{P}_{\bowtie a}[\phi|\psi] \in \mathsf{Stat}, \\ \phi, \psi \in \mathsf{Stat} & \Rightarrow & \phi\, \mathcal{U} \psi, \Diamond\phi, \Box\, \phi \in \mathsf{Path}\,. \end{array}$$

Here $\bowtie \in \{<, \leq, >, \geq\}$ and $a \in [0, 1]$.

Semantics: The satisfiability of state-formulas ($s \models \phi$ for a state s) and path-formulas ($\omega \models \psi$ for a path ω) is defined as an extension of the satisfiability for pCTL. Hence, the satisfiability of the logical, temporal, and pCTL operators is defined in the usual way. For the conditional probabilistic operators we define

$$\begin{array}{rcl} s \models \mathbb{P}_{\leq a}[\phi|\psi] & \Leftrightarrow & \mathbb{P}^+_s(\{\omega \in \mathsf{Paths}(s) \mid \omega \models \phi\}|\{\omega \in \mathsf{Paths}(s) \mid \omega \models \psi\}) \leq a, \\ s \models \mathbb{P}_{\geq a}[\phi|\psi] & \Leftrightarrow & \mathbb{P}^-_s(\{\omega \in \mathsf{Paths}(s) \mid \omega \models \phi\}|\{\omega \in \mathsf{Paths}(s) \mid \omega \models \psi\}) \geq a, \end{array}$$

and similarly for $s \models \mathbb{P}_{<a}[\phi|\psi]$ and $s \models \mathbb{P}_{>a}[\phi|\psi]$. We say that a model $\mathcal{M}$ satisfy ϕ, denoted by $\mathcal{M} \models \phi$ if $s_0 \models \phi$.

In the following we fix some notation that we will use in the rest of the chapter,

$$\mathbb{P}_s^+[\phi] \triangleq \mathbb{P}_s^+(\{\omega \in \mathsf{Paths}(s) \mid \omega \models \phi\}),$$
$$\mathbb{P}_s^+[\phi|\psi] \triangleq \mathbb{P}_s^+(\{\omega \in \mathsf{Paths}(s) \mid \omega \models \phi\}|\{\omega \in \mathsf{Paths}(s) \mid \omega \models \psi\}),$$
$$\mathbb{P}_{s,\eta}[\phi|\psi] \triangleq \mathbb{P}_{s,\eta}(\{\omega \in \mathsf{Paths}(s) \mid \omega \models \phi\}|\{\omega \in \mathsf{Paths}(s) \mid \omega \models \psi\}),$$

$\mathbb{P}_s^-[\phi|\psi]$ and $\mathbb{P}_s^-[\phi]$ are defined analogously.

Observation 2.4.1. Note that, in order to check if $s \models \mathbb{P}_{\bowtie a}[\phi|\psi]$, we only need to consider the cases where $\phi = \phi_1 \mathcal{U} \phi_2$ and where ψ is either $\psi_1 \mathcal{U} \psi_2$ or $\Box\psi_1$. This follows from $\Diamond\phi \leftrightarrow \mathbf{true}\mathcal{U}\phi$, $\Box\phi \leftrightarrow \neg\Diamond\neg\phi$ and the relations

$$\mathbb{P}_s^+[\neg\phi|\psi] = 1 - \mathbb{P}_s^-[\phi|\psi] \qquad \text{and} \qquad \mathbb{P}_s^-[\neg\phi|\psi] = 1 - \mathbb{P}_s^+[\phi|\psi]$$

derived from Lemma 2.3.2. Since there is no way to relate $\mathbb{P}^+[\phi|\psi]$ and $\mathbb{P}^+[\phi|\neg\psi]$, we have to provide algorithms to compute both $\mathbb{P}^+[\phi|\psi_1\mathcal{U}\psi_2]$ and $\mathbb{P}^+[\phi|\Box\psi_1]$. The same remark holds for the minimal conditional probabilities $\mathbb{P}^-[\phi|\psi_1\mathcal{U}\psi_2]$ and $\mathbb{P}^-[\phi|\Box\psi_1]$. In this chapter we will only focus on the former problem, i.e., computing maximum conditional probabilities, the latter case follows using similar techniques.

2.4.1 Expressiveness

We now show that cpCTL is strictly more expressive than pCTL. The notion of *expressiveness* of a temporal logic is based on the notion of *formula equivalence*. Two temporal logic formulas ϕ and ψ are *equivalent* with respect to a set $\mathcal{D}$ of models (denoted by $\phi \equiv_{\mathcal{D}} \psi$) if for any model $m \in \mathcal{D}$ we have $m \models \phi$ if and only if $m \models \psi$. A temporal logic $\mathcal{L}$ is said to be *at least as expressive as* a temporal logic $\mathcal{L}'$, over a set of models $\mathcal{D}$, if for any formula $\phi \in \mathcal{L}'$ there is a formula $\psi \in \mathcal{L}$ that is equivalent to ϕ over $\mathcal{D}$. Two temporal logics are *equally expressive* when each of them is at least as expressive as the other. Formally:

Definition 2.4.1. Two temporal logics $\mathcal{L}$ and $\mathcal{L}'$ are equally expressive with respect to $\mathcal{D}$ if

$$\forall\phi \in \mathcal{L}.\left(\exists\psi \in \mathcal{L}'.\phi \equiv_{\mathcal{D}} \psi\right) \wedge \forall\psi \in \mathcal{L}'.\left(\exists\phi \in \mathcal{L}.\phi \equiv_{\mathcal{D}} \psi\right).$$

Theorem 2.4.2. cpCTL is more expressive than pCTL with respect to MCs and MDPs.

Proof. Obviously cpCTL is at least as expressive as pCTL, hence we only need to show that the reverse does not hold. The result is rather intuitive since the semantics of the conditional operator for cpCTL logic is provided by a non-linear equation whereas there is no pCTL formula with non-linear semantics.

The following is a formal proof. We plan to show that there is no pCTL formula ψ equivalent to $\phi = \mathbb{P}_{\leq 0.5}[\Diamond A | \Diamond B]$, with A and B atomic propositions. The proof is by cases on the structure of the pCTL formula ψ. The most interesting case is when ψ is of the form $\mathbb{P}_{\leq b}[\psi]$, so we will only prove this case. In addition we restrict our attention to b's such that $0 < b < 1$ (the cases $b = 0$ and $b = 1$ are easy). In Figure 2.2 we depict the Markov Chains involved in the proof. We use $\neg\psi_1$ to mark the states with an assignment of truth values (to propositional variables) falsifying ψ_1.

Case $\psi = \mathbb{P}_{\leq b}[\Diamond\psi_1]$:
If ψ_1 is **true** or **false** the proof is obvious, so we assume otherwise. We first note that we either have $\neg\psi_1 \Rightarrow \neg(B \wedge \neg A)$ or $\neg\psi_1 \Rightarrow (B \wedge \neg A)$. In the former case, it is easy to see (using $\neg B \Rightarrow \psi_1$) that we have $m_2 \models \phi$ and $m_2 \not\models \psi$. In the second case we have $m_1 \not\models \phi$ and $m_1 \models \psi$.

Case $\psi = \mathbb{P}_{\leq b}[\psi_1 \mathcal{U} \psi_2]$:
We assume $\psi_1 \neq$ **true**, otherwise we fall into the previous case. We can easily see that we have $m_3 \models \psi$ but $m_3 \not\models \phi$.

Case $\psi = \mathbb{P}_{\leq b}[\Box\psi_1]$:
The case when $\psi_1 =$ **true** is easy, so we assume $\psi_1 \neq$ **true**. We can easily see that we have $m_3 \models \psi$ but $m_3 \not\models \phi$. □

Note that, since MCs are a special case of MDPs, the proof also holds for the latter class.

We note that, in spite of the fact that a cpCTL formula of the form $\mathbb{P}_{\leq a}[\phi|\psi]$ cannot be expressed as a pCTL formula, if dealing with fully

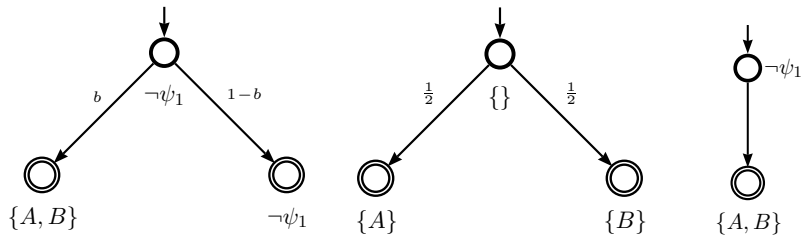


Figure 2.2: Markov Chains m_1, m_2, and m_3 respectively.

probabilistic systems (i.e. systems without nondeterministic choices) it is still possible to verify such conditional probabilities formulas as the quotient of two pCTL* formulas: $\mathbb{P}[\phi|\psi] = \frac{\mathbb{P}[\phi \wedge \psi]}{\mathbb{P}[\psi]}$. However, this observation does not carry over to systems where probabilistic choices are combined with nondeterministic ones (as it is the case of Markov Decision Processes). This is due to the fact that, in general, it is not the case that $\mathbb{P}^+[\phi|\psi] = \frac{\mathbb{P}^+[\phi \wedge \psi]}{\mathbb{P}^+[\psi]}$.

2.5 Semi History-Independent and Deterministic Schedulers

Recall that there exist optimizing (i.e. maximizing and minimizing) schedulers on pCTL that are HI and deterministic [BdA95]. We show that, for cpCTL, deterministic schedulers still suffice to reach the optimal conditional probabilities. Because we now have to solve a non-linear optimization problem, the proof differs from the pCTL case in an essential way. We also show that HI schedulers do not suffice to attain optimal conditional probability and introduce the family of semi history-independent schedulers that do attain it.

2.5.1 Semi History-Independent Schedulers

The following example shows that maximizing schedulers are not necessarily HI.

Example 2.5.1. Let Π be the MDP of Figure 2.3 and the conditional probability $\mathbb{P}_{s_0,\eta}[\Diamond B|\Diamond P]$. There are only three deterministic history independent schedulers, choosing π_1, π_2, or π_3 in s_0. For the first one, the conditional probability is undefined and for the second and third it is 0. The scheduler η that maximizes $\mathbb{P}_{s_0,\eta}[\Diamond B|\Diamond P]$ satisfies $\eta(s_0) = \pi_3$, $\eta(s_0 s_3) = \pi_5$, and $\eta(s_0 s_3 s_0) = \pi_1$. Since η chooses on s_0 first π_2 and later π_1, η is not history independent.

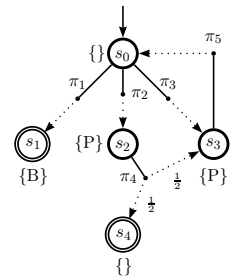


Figure 2.3: MDP

Fortunately, as we show in Theorem 2.5.3, there exists a *nearly* HI scheduler that attain optimal conditional probability. We say that such schedulers are nearly HI because they always take the same decision *before* the system reaches a certain condition φ and also always take the same decision *after* φ. This family of schedulers is called φ-semi history independent (φ-sHI for short) and the condition φ is called *stopping condition*. For a pCTL path formula ϕ the stopping condition is a boolean proposition either validating or contradicting ϕ. So, the (validating) stopping condition of $\Diamond\phi$ is ϕ whereas the (contradicting) stopping condition of $\Box\phi$ is $\neg\phi$. Formally:

$$\mathsf{StopC}(\phi) \triangleq \begin{cases} \neg\psi_1 \vee \psi_2 & \text{if } \phi = \psi_1 \mathcal{U} \psi_2, \\ \neg\psi & \text{if } \phi = \Box\psi. \end{cases}$$

Similarly, for a cpCTL formula $\mathbb{P}_{\bowtie a}[\phi|\psi]$, the stopping condition is a condition either validating or contradicting any of its pCTL formulas (ϕ, ψ), i.e., $\mathsf{StopC}(\mathbb{P}_{\bowtie a}[\phi|\psi]) = \mathsf{StopC}(\phi) \vee \mathsf{StopC}(\psi)$.

We now proceed with the formalization of semi history independent schedulers.

Definition 2.5.2 (Semi History-Independent Schedulers)**.** Let Π be a MDP, η a scheduler for Π, and $\phi \vee \psi \in \mathsf{Stat}$. We say that η is a $(\phi \vee \psi)$ *semi history-independent scheduler* ($(\phi \vee \psi)$-sHI scheduler for short) if for all

$\sigma_1, \sigma_2 \in \mathsf{Paths}^\star(s)$ such that $\mathsf{last}(\sigma_1) = \mathsf{last}(\sigma_2)$ we have

$$\begin{aligned} \sigma_1, \sigma_2 \not\models \Diamond(\phi \vee \psi) \Rightarrow \eta(\sigma_1) = \eta(\sigma_2), and \quad & \{\text{HI before stopping condition}\} \\ \sigma_1, \sigma_2 \models \Diamond\phi \Rightarrow \eta(\sigma_1) = \eta(\sigma_2), and \quad & \{\text{HI after stopping condition}\} \\ \sigma_1, \sigma_2 \models \Diamond\psi \Rightarrow \eta(\sigma_1) = \eta(\sigma_2). \quad & \{\text{HI after stopping condition}\} \end{aligned}$$

We denote the set of all φ-sHI schedulers of Π by $\mathsf{Sch}^\varphi(\Pi)$.

We now prove that semi history-independent schedulers suffice to attain the optimal conditional probabilities for cpCTL formula.

Theorem 2.5.3. Let Π be a MDP, $\phi, \psi \in \mathsf{Path}$, and $\varphi = \mathsf{StopC}(\phi) \vee \mathsf{StopC}(\psi)$. Assume that there exists a scheduler η such that $\mathbb{P}_\eta[\psi] > 0$. Then:

$$\mathbb{P}^+[\phi|\psi] = \sup_{\eta \in \mathsf{Sch}^\varphi(\Pi)} \mathbb{P}_\eta[\phi|\psi].$$

(If there exists no scheduler η such that $\mathbb{P}_\eta[\psi] > 0$, then the supremum is 0.)

The proof of this theorem is rather complex. The first step is to prove that there exists a scheduler η HI before the stopping condition and such that $\mathbb{P}_\eta[\phi|\psi]$ is 'close' (i.e. not further than a small value ϵ) to the optimal conditional probability $\mathbb{P}^+[\phi|\psi]$. For this purpose we introduce some definitions and prove this property first for long paths (Lemma 2.5.5) and then, step-by-step, in general (Lemma 2.5.6 and Corollary 2.5.1). After that, we create a scheduler that is also HI after the stopping condition and whose conditional probability is still close to the optimal one (Lemma 2.5.7). From the above results, the theorem readily follows.

We now introduce some definitions and notation that we will need for the proof.

Definition 2.5.4 (Cuts)**.** Given a MDP Π we say that a set $K \subseteq \mathsf{Paths}^\star(\Pi)$ is a *cut* of Π if K is a downward-closed set of finite paths such that every infinite path passes through it, i.e.

- $\forall\, \sigma_1 \in K \,.\, \forall\, \sigma_2 \in \mathsf{Paths}^\star(\Pi) \,.\, \sigma_1 \sqsubseteq \sigma_2 \implies \sigma_2 \in K$, and

- $\forall \omega \in \mathsf{Paths}(\Pi) \,.\, \exists \sigma \in K \,.\, \sigma \sqsubset \omega$.

where $\sigma_1 \sqsubseteq \sigma_2$ means that σ_2 is an "extension" of σ_1, i.e. $\sigma_2 = \sigma_1\sigma'$ for some path σ'. We denote the set of all cuts of Π by $K(\Pi)$.

For $R \subseteq \mathsf{Paths}^\star(s)$, we say that η is history independent in R if for all $\sigma_1, \sigma_2 \in R$ such that $\mathsf{last}(\sigma_1) = \mathsf{last}(\sigma_2)$ we have that $\eta(\sigma_1) = \eta(\sigma_2)$. We also define the sets Φ and Ψ as the set of finite paths validating ϕ and ψ respectively, i.e. $\Phi \triangleq \{\sigma \in \mathsf{Paths}^\star(\Pi) \mid \sigma \models \phi\}$ and $\Psi \triangleq \{\sigma \in \mathsf{Paths}^\star(\Pi) \mid \sigma \models \psi\}$. Finally, given a MDP Π, two path formulas ϕ, ψ, and $\hat{\epsilon} > 0$ we define the set

$$\mathcal{K} \triangleq \{(K, \eta) \in K(\Pi) \times \mathsf{Sch}(\Pi) \mid \Phi \cup \Psi \subseteq K \text{ and } \eta \text{ is HI in } K \setminus (\Phi \cup \Psi) \text{ and } \mathbb{P}^+[\phi|\psi] - \mathbb{P}_\eta[\phi|\psi] < \hat{\epsilon}\}$$

If a scheduler η is HI in $K \setminus (\Phi \cup \Psi)$ then we say that η is HI before the stopping condition.

Lemma 2.5.5 (non emptiness of $\mathcal{K}$)**.** There exists (K, η) such that $(K, \eta) \in \mathcal{K}$ and that its complement $K^c \triangleq \mathsf{Paths}^\star(\Pi) \setminus K$ is finite.

Proof. We show that, given formulas ϕ, ψ and $\hat{\epsilon} > 0$, there exists a cut K and a scheduler $\eta^\star$ such that K^c is finite, $\Phi \cup \Psi \subseteq K$, $\eta^\star$ is HI in $K \setminus (\Phi \cup \Psi)$, and $\mathbb{P}^+[\phi|\psi] - \mathbb{P}_{\eta^\star}[\phi|\psi] < \hat{\epsilon}$.

The proof is by case analysis on the structure of ϕ and ψ. We will consider the cases where ϕ and ψ are either "eventually operators" ($\Diamond$) or "globally operators" ($\Box$), the proof for the until case follows along the same lines.

• **Case ϕ is of the form $\Diamond\phi$ and ψ is of the form $\Diamond\psi$:**
Let us start by defining the the probability of reaching ϕ in at most N steps, as $\mathbb{P}_\eta[\leq N, \Diamond\phi] \triangleq \mathbb{P}_\eta[\langle\{\sigma \in \mathsf{Paths}^\star(\Pi) \mid \sigma \models \Diamond\phi \wedge |\sigma| \leq N\}\rangle]$. Note that for all pCTL reachability properties $\Diamond\phi$ and schedulers η we have

$$\lim_{N \to \infty} \mathbb{P}_\eta[\leq N, \Diamond\phi] = \mathbb{P}_\eta[\Diamond\phi].$$

We also note that this result also holds for pCTL* formulas of the form $\Diamond\phi \wedge \Diamond\psi$.

Let us now take a scheduler η and a number N such that

$$\mathbb{P}^{+}[\Diamond\phi|\Diamond\psi] - \mathbb{P}_{\eta}[\Diamond\phi|\Diamond\psi] < \epsilon \triangleq \hat{\epsilon}/3, \text{ and} \tag{2.1}$$

$$\mathbb{P}_{\eta}[\Diamond\phi \wedge \Diamond\psi] - \mathbb{P}_{\eta}[\leq N, \Diamond\phi \wedge \Diamond\psi] < \epsilon', \text{ and} \tag{2.2}$$

$$\mathbb{P}_{\eta}[\Diamond\psi] - \mathbb{P}_{\eta}[\leq N, \Diamond\psi] < \epsilon'. \tag{2.3}$$

where ϵ' is such that $\epsilon' < \mathsf{min}\left(2 \cdot \epsilon \cdot \mathbb{P}_{\eta}[\Diamond\psi], \frac{\epsilon \cdot \mathbb{P}_{\eta}[\Diamond\psi]^2}{\mathbb{P}_{\eta}[\Diamond\phi \wedge \Diamond\psi] + 2 \cdot \epsilon \cdot \mathbb{P}_{\eta}[\Diamond\psi]}\right)$. The reasons for this particular choice for the bound of ϵ' will become clear later on in the proof.

We define K as $\Phi \cup \Psi \cup \mathsf{Paths}^{\star}(\leq N, \Pi)$, where the latter set is defined as the set of paths with length larger than N, i.e. $\mathsf{Paths}^{\star}(\leq N, \Pi) \triangleq \{\sigma \in \mathsf{Paths}^{\star}(\Pi) \mid N \leq |\sigma|\}$. In addition, we define $\eta^{\star}$ as a scheduler HI in $\mathsf{Paths}^{\star}(\leq N, \Pi)$ behaving like η for paths of length less than or equal to N which additionally minimizes $\mathbb{P}[\Diamond\psi]$ after level N. In order to formally define such a scheduler we let S_N to be the set of states that can be reached in exactly N steps, i.e., $S_N \triangleq \{s \in S \mid \exists \sigma \in \mathsf{Paths}^{\star}(\Pi) : |\sigma| = N \wedge \mathsf{last}(\sigma) = s\}$. Now for each $s \in S$ we let ξ_s to be a HI s-scheduler such that $\mathbb{P}_{s,\xi_s}[\Diamond\psi] = \mathbb{P}^{-}_{s}[\Diamond\psi]$. Note that such a scheduler exists, i.e., it is always possible to find a HI scheduler minimizing a reachability pCTL formula [BdA95].

We now define $\eta^{\star}$ as

$$\eta^{\star}(\sigma) \triangleq \begin{cases} \xi_s(\sigma_{|\alpha|}\sigma_{|\alpha|+1}\cdots\sigma_{|\sigma|}) & \text{if } \alpha \sqsubseteq \sigma \text{ for some } \alpha \in \mathsf{Paths}^{\star}(=N, \Pi) \\ & \qquad \text{such that } \mathsf{last}(\alpha) = s, \\ \eta(\sigma) & \text{otherwise.} \end{cases}$$

where $\mathsf{Paths}^{\star}(=N, \Pi)$ denotes the set of paths of Π of length N. It is easy to see that $\eta^{\star}$ minimizes $\mathbb{P}[\Diamond\psi]$ after level N. As for the history independency of $\eta^{\star}$ in K there is still one more technical detail to consider: note there may still be paths $\alpha_1 s_1 \sigma_1 t$ and $\alpha_2 s_2 \sigma_2 t$ such that $\alpha_1 s_1, \alpha_2 s_2 \in \mathsf{Paths}^{\star}(=N, \Pi)$ and $\xi_{s_1}(s_1\sigma_1 t) \neq \xi_{s_2}(s_2\sigma_2 t)$. This is the case when there is more than one distribution in $\tau(t)$ minimizing $\mathbb{P}_t[\Diamond\psi]$, and ξ_{s_1} happens to choose a different

(minimizing) distribution than ξ_{s_2} for the state t. Thus, the selection of the family of schedulers $\{\xi_s\}_{s\in S_N}$ must be made in such a way that: for all $s_1, s_2 \in S_N$ we have $\mathbb{P}_{s_1,\xi_{s_1}}[\Diamond\psi] = \mathbb{P}^-_{s_1}[\Diamond\psi]$, $\mathbb{P}_{s_2,\xi_{s_2}}[\Diamond\psi] = \mathbb{P}^-_{s_2}[\Diamond\psi]$, and for all $\sigma_1 t \in \mathsf{Paths}^\star(s_1), \sigma_2 t \in \mathsf{Paths}^\star(s_2) : \xi_{s_1}(\sigma_1 t) = \xi_{s_2}(\sigma_2 t)$. It is easy to check that such family exists. We conclude that $\eta^\star$ is HI in $\mathsf{Paths}^\star(\leq N, \Pi)$ and thus HI in $K \setminus (\Phi \cup \Psi)$.

We note that $\mathbb{P}_{\eta^\star}[\Diamond\psi] > 0$, this follows from $0 < \mathbb{P}_\eta[\Diamond\psi]$, (2.1), (2.3), and the definition of $\eta^\star$.

Having defined $\eta^\star$ we proceed to prove that such scheduler satisfies $\mathbb{P}^+[\phi|\psi] - \mathbb{P}_\eta[\phi|\psi] < \hat{\epsilon}$. It is possible to show that:

$$\mathbb{P}_\eta[\leq N, \Diamond\psi] \leq \quad \mathbb{P}_{\eta^\star}[\Diamond\psi] \leq \quad \mathbb{P}_\eta[\Diamond\psi], \tag{2.4}$$

$$\mathbb{P}_\eta[\leq N, \Diamond\phi \wedge \Diamond\psi] \leq \quad \mathbb{P}_{\eta^\star}[\Diamond\phi \wedge \Diamond\psi] < \quad \mathbb{P}_\eta[\Diamond\phi \wedge \Diamond\psi] + \epsilon \cdot \mathbb{P}_\eta[\Diamond\psi]. \tag{2.5}$$

(2.4) and the first inequality of (2.5) follow straightforwardly from the definition of $\eta^\star$. For the second inequality of (2.5) suppose by contradiction that $\mathbb{P}_{\eta^\star}[\Diamond\phi \wedge \Diamond\psi] \geq \mathbb{P}_\eta[\Diamond\phi \wedge \Diamond\psi] + \epsilon \cdot \mathbb{P}_\eta[\Diamond\psi]$. Then

$$\frac{\mathbb{P}_{\eta^\star}[\Diamond\phi \wedge \Diamond\psi]}{\mathbb{P}_{\eta^\star}[\Diamond\psi]} \geq \frac{\mathbb{P}_\eta[\Diamond\phi \wedge \Diamond\psi] + \epsilon \cdot \mathbb{P}_\eta[\Diamond\psi]}{\mathbb{P}_\eta[\Diamond\psi]} = \mathbb{P}_\eta[\Diamond\phi|\Diamond\psi] + \epsilon$$

contradicting (2.1).

Now we have all the necessary ingredients to show that

$$|\mathbb{P}_\eta[\Diamond\phi|\Diamond\psi] - \mathbb{P}_{\eta^\star}[\Diamond\phi|\Diamond\psi]| < 2 \cdot \epsilon. \tag{2.6}$$

Note that

$$\frac{\mathbb{P}_\eta[\Diamond\phi \wedge \Diamond\psi] - \epsilon'}{\mathbb{P}_\eta[\Diamond\psi]} < \mathbb{P}_{\eta^\star}[\Diamond\phi|\Diamond\psi] \text{ and } \mathbb{P}_{\eta^\star}[\Diamond\phi|\Diamond\psi] < \frac{\mathbb{P}_\eta[\Diamond\phi \wedge \Diamond\psi] + \epsilon \cdot \mathbb{P}_\eta[\Diamond\psi]}{\mathbb{P}_\eta[\Diamond\psi] - \epsilon'}.$$

The first inequality holds because $\mathbb{P}_{\eta^\star}[\Diamond\psi] \leq \mathbb{P}_\eta[\Diamond\psi]$ and (combining (2.5) and (2.2)) $\mathbb{P}_{\eta^\star}[\Diamond\phi \wedge \Diamond\psi] > \mathbb{P}_\eta[\Diamond\phi \wedge \Diamond\psi] - \epsilon'$. The second inequality holds

because $\mathbb{P}_{\eta^\star}[\Diamond\phi \wedge \Diamond\psi] < \mathbb{P}_{\eta}[\Diamond\phi \wedge \Diamond\psi] + \epsilon \cdot \mathbb{P}_{\eta}[\Diamond\psi]$ and (combining (2.4) and (2.3)) $\mathbb{P}_{\eta^\star}[\Diamond\psi] > \mathbb{P}_{\eta}[\Diamond\psi] - \epsilon'$. It is easy to see that $\mathbb{P}_{\eta}[\Diamond\phi|\Diamond\psi]$ falls in the same interval, i.e., both $\mathbb{P}_{\eta}[\Diamond\phi|\Diamond\psi]$ and $\mathbb{P}_{\eta^\star}[\Diamond\phi|\Diamond\psi]$ are in the interval

$$\left(\frac{\mathbb{P}_{\eta}[\Diamond\phi \wedge \Diamond\psi] - \epsilon'}{\mathbb{P}_{\eta}[\Diamond\psi]} \quad , \quad \frac{\mathbb{P}_{\eta}[\Diamond\phi \wedge \Diamond\psi] + \epsilon \cdot \mathbb{P}_{\eta}[\Diamond\psi]}{\mathbb{P}_{\eta}[\Diamond\psi] - \epsilon'} \right).$$

Thus, we can prove (2.6) by proving

$$\frac{\mathbb{P}_{\eta}[\Diamond\phi \wedge \Diamond\psi]}{\mathbb{P}_{\eta}[\Diamond\psi]} - \frac{\mathbb{P}_{\eta}[\Diamond\phi \wedge \Diamond\psi] - \epsilon'}{\mathbb{P}_{\eta}[\Diamond\psi]} < 2 \cdot \epsilon, \text{ and}$$
$$\frac{\mathbb{P}_{\eta}[\Diamond\phi \wedge \Diamond\psi] + \epsilon \cdot \mathbb{P}_{\eta}[\Diamond\psi]}{\mathbb{P}_{\eta}[\Diamond\psi] - \epsilon'} - \frac{\mathbb{P}_{\eta}[\Diamond\phi \wedge \Diamond\psi]}{\mathbb{P}_{\eta}[\Diamond\psi]} < 2 \cdot \epsilon.$$

The first inequality holds if and only if $\epsilon' < 2 \cdot \epsilon \cdot \mathbb{P}_{\eta}[\Diamond\psi]$. As for the second inequality, we have

$$\begin{aligned}
& \tfrac{\mathbb{P}_{\eta}[\Diamond\phi\wedge\Diamond\psi]+\epsilon\cdot\mathbb{P}_{\eta}[\Diamond\psi]}{\mathbb{P}_{\eta}[\Diamond\psi]-\epsilon'} - \tfrac{\mathbb{P}_{\eta}[\Diamond\phi\wedge\Diamond\psi]}{\mathbb{P}_{\eta}[\Diamond\psi]} < 2\cdot\epsilon \\
\iff \quad & \mathbb{P}_{\eta}[\Diamond\psi]^2 \cdot \epsilon + \mathbb{P}_{\eta}[\Diamond\phi \wedge \Diamond\psi] \cdot \epsilon' < 2 \cdot \epsilon \cdot (\mathbb{P}_{\eta}[\Diamond\psi] - \epsilon') \cdot \mathbb{P}_{\eta}[\Diamond\psi] \\
\iff \quad & \mathbb{P}_{\eta}[\Diamond\psi]^2 \cdot \epsilon + \mathbb{P}_{\eta}[\Diamond\phi \wedge \Diamond\psi] \cdot \epsilon' < 2 \cdot \epsilon \cdot \mathbb{P}_{\eta}[\Diamond\psi]^2 - 2 \cdot \epsilon \cdot \epsilon' \cdot \mathbb{P}_{\eta}[\Diamond\psi] \\
\iff \quad & \epsilon' < \tfrac{\epsilon\cdot\mathbb{P}_{\eta}[\Diamond\psi]^2}{\mathbb{P}_{\eta}[\Diamond\phi\wedge\Diamond\psi]+2\cdot\epsilon\cdot\mathbb{P}_{\eta}[\Diamond\psi]}.
\end{aligned}$$

We conclude, by definition of ϵ', that both inequalities hold.

Now, putting (2.1) and (2.6) together, we have $\mathbb{P}^+[\Diamond\phi|\Diamond\psi] - \mathbb{P}_{\eta^\star}[\Diamond\phi|\Diamond\psi] < 3 \cdot \epsilon = \hat{\epsilon}$, which concludes the proof for this case.

• **Case ϕ is of the form $\Diamond\phi$ and ψ is of the form $\Box\psi$:**
We now construct a cut K and a scheduler $\eta^\star$ such that K^c is finite, $\Phi \cup \Psi \subseteq K$, $\eta^\star$ is HI in $K \setminus (\Phi \cup \Psi)$, and $\mathbb{P}_{\eta^\star}[\Box\neg\phi|\Box\psi] - \mathbb{P}^-[\Box\neg\phi|\Box\psi] < \hat{\epsilon}$. Note that such a cut and scheduler also satisfy $\mathbb{P}^+[\Diamond\phi|\Box\psi] - \mathbb{P}_{\eta^\star}[\Diamond\phi|\Box\psi] < \hat{\epsilon}$.

The proof goes similarly to the previous case. We start by defining the probability of paths of length N always satisfying ϕ as $\mathbb{P}_{\eta}[=N, \Box\phi] \triangleq$

$\mathbb{P}_\eta[\langle\{\sigma \in \mathsf{Paths}^\star(\Pi) \mid \sigma \models \Box\phi \wedge |\sigma| = N\}\rangle]$. Note that for all pCTL formula of the form $\Box\phi$ and schedulers η we have

$$\lim_{N\to\infty} \mathbb{P}_\eta[{=}N, \Box\phi] = \mathbb{P}_\eta[\Box\phi].$$

The same result holds for the pCTL* formula $\Box(\phi \wedge \psi)$. It is easy to check that for all N and ϕ we have $\mathbb{P}_\eta[{=}N, \Box\phi] \geq \mathbb{P}_\eta[\Box\phi]$.

Now we take a scheduler η and a number N such that:

$$\begin{aligned} 0 \leq \mathbb{P}_\eta[\Box\neg\phi|\Box\psi] - \mathbb{P}^-[\Box\neg\phi|\Box\psi] < \epsilon \triangleq \hat{\epsilon}/3, \text{ and} \\ 0 \leq \mathbb{P}_\eta[{=}N, \Box(\neg\phi \wedge \psi)] - \mathbb{P}_\eta[\Box(\neg\phi \wedge \psi)] < \epsilon', \text{ and} \\ 0 \leq \mathbb{P}_\eta[{=}N, \Box\psi] - \mathbb{P}_\eta[\Box\psi] < \epsilon'. \end{aligned}$$

where ϵ' is such that $\epsilon' < \mathsf{min}\left(\epsilon \cdot \mathbb{P}_\eta[\Box\psi], \frac{\epsilon \cdot \mathbb{P}_\eta[\Box\psi]^2}{\mathbb{P}_\eta[\Box(\neg\phi \wedge \psi)]}\right)$.

We define K as before, i.e., $K \triangleq \Phi \cup \Psi \cup \mathsf{Paths}^\star(\leq N, \Pi)$. In addition, we can construct (as we did in the previous case) a scheduler $\eta^\star$ behaving as η for paths of length at most N and maximizing (instead of minimizing as in the previous case) $\mathbb{P}[\Box\psi]$ afterwards. Again, it is easy to check that $\eta^\star$ is HI in $K \setminus (\Phi \cup \Psi)$.

Then we have

$$\begin{array}{rcccl} \mathbb{P}_\eta[\Box\psi] & \leq & \mathbb{P}_{\eta^\star}[\Box\psi] & \leq & \mathbb{P}_\eta[{=}N, \Box\psi], \\ \mathbb{P}_\eta[\Box(\neg\phi \wedge \psi)] - \epsilon \cdot \mathbb{P}_\eta[\Box\psi] & < & \mathbb{P}_{\eta^\star}[\Box(\neg\phi \wedge \psi)] & \leq & \mathbb{P}_\eta[{=}N, \Box(\neg\phi \wedge \psi)]. \end{array}$$

In addition, it is easy to check that

$$\begin{array}{rcccl} -\mathbb{P}_\eta[\Box\psi] \cdot \epsilon & < & \mathbb{P}_{\eta^\star}[\Box(\neg\phi \wedge \psi)] - \mathbb{P}_\eta[\Box(\neg\phi \wedge \psi)] & < & \epsilon' \\ 0 & \leq & \mathbb{P}_{\eta^\star}[\Box\psi] - \mathbb{P}_\eta[\Box\psi] & < & \epsilon'. \end{array}$$

Similarly to the previous case we now show that

$$|\mathbb{P}_\eta[\Box\neg\phi|\Box\psi] - \mathbb{P}_{\eta^\star}[\Box\neg\phi|\Box\psi]| < 2 \cdot \epsilon. \tag{2.7}$$

which together with $\mathbb{P}_\eta[\Box\neg\phi|\Box\psi] - \mathbb{P}^-[\Box\neg\phi|\Box\psi] < \epsilon$ concludes the proof.

In order to prove (2.7) we show that

$$-2 \cdot \epsilon < \mathbb{P}_{\eta^\star}[\Box\neg\phi|\Box\psi] - \mathbb{P}_{\eta}[\Box\neg\phi|\Box\psi] < \epsilon$$

or, equivalently

a) $\mathbb{P}_{\eta^\star}[\Box(\neg\phi \wedge \psi)] \cdot \mathbb{P}_{\eta}[\Box\psi] - \mathbb{P}_{\eta}[\Box(\neg\phi \wedge \psi)] \cdot \mathbb{P}_{\eta^\star}[\Box\psi] < \mathbb{P}_{\eta}[\Box\psi] \cdot \mathbb{P}_{\eta^\star}[\Box\psi] \cdot \epsilon$, and

b) $2 \cdot \mathbb{P}_{\eta}[\Box\psi] \cdot \mathbb{P}_{\eta^\star}[\Box\psi] \cdot \epsilon < \mathbb{P}_{\eta^\star}[\Box(\neg\phi \wedge \psi)] \cdot \mathbb{P}_{\eta}[\Box\psi] - \mathbb{P}_{\eta}[\Box(\neg\phi \wedge \psi)] \cdot \mathbb{P}_{\eta^\star}[\Box\psi]$.

It is possible to verify that a) is equivalent to $\epsilon' < \epsilon \cdot \mathbb{P}_{\eta}[\Box\psi]$ and that b) is equivalent to $\epsilon' < \frac{\epsilon \cdot \mathbb{P}_{\eta}[\Box\psi]^2}{\mathbb{P}_{\eta}[\Box(\neg\phi\wedge\psi)]}$. The desired result follows by definition of ϵ'. □

In the proof of the following lemma we step-by-step find pairs (K, η) in $\mathcal{K}$ with larger K and η still close to the optimal until finally K is equal to the whole of $\mathsf{Paths}^\star(\Pi)$.

Lemma 2.5.6 (completeness of $\mathcal{K}$)**.** There exists a scheduler η such that $(\mathsf{Paths}^\star(\Pi), \eta) \in \mathcal{K}$.

Proof. We prove that if we take a $(K, \eta) \in \mathcal{K}$ such that $|K^c|$ is minimal then $K^c = \emptyset$ or, equivalently, $K = \mathsf{Paths}^\star(\Pi)$. Note that a pair (K, η) with minimal $|K^c|$ exists because, by the previous lemma, $\mathcal{K}$ is not empty.

The proof is by contradiction: we suppose $K^c \neq \emptyset$ and arrive to a contradiction on the minimality of $|K^c|$. Formally, we show that for all $(K, \eta) \in \mathcal{K}$ such that $K^c \neq \emptyset$, there exists a cut $K^\star \supset K$ and a scheduler $\eta^\star$ such that $(K^\star, \eta^\star) \in \mathcal{K}$, i.e. such that $\eta^\star$ is HI in $K^\star \setminus (\Phi \cup \Psi)$ and $\mathbb{P}_{\eta}[\phi|\psi] \leq \mathbb{P}_{\eta^\star}[\phi|\psi]$.

To improve readability, we prove this result for the case in which ϕ is of the form $\Diamond\phi$ and ψ is of the form $\Diamond\psi$. However, all the technical details of the proof hold for arbitrary ϕ and ψ.

Let us start defining the *boundary* of a cut K as

$$\delta K \triangleq \{\sigma_1 \in K \,|\, \forall\, \sigma_2 \in \mathsf{Paths}^\star(M)\,.\, \sigma_2 \sqsubset \sigma_1 \implies \sigma_2 \notin K\}.$$

Let ρ be a path in K^c such that $\rho t \in \delta K$. Note that by assumption of $K^c \neq \emptyset$ such ρ exists. Now, if for all paths $\alpha \in K$ we have $\mathsf{last}(\alpha) = \mathsf{last}(\rho) \Longrightarrow \eta(\alpha) = \eta(\rho)$ then η is also HI in $(K \cup \{\rho\}) \setminus (\Phi \cup \Psi)$ so we have $(K \cup \{\rho\}, \eta) \in \mathcal{K}$ as we wanted to show. Now let us assume otherwise, i.e. that there exists a path $\alpha \in K \setminus (\Phi \cup \Psi)$ such that $\mathsf{last}(\alpha) = \mathsf{last}(\rho)$ and $\eta(\alpha) \neq \eta(\rho)$. We let $s \triangleq \mathsf{last}(\rho)$, $\pi_1 \triangleq \eta(\rho)$, $\pi_2 \triangleq \eta(\alpha)$, and $K_s \triangleq \{\sigma \in K \mid \mathsf{last}(\sigma) = s\} \setminus (\Phi \cup \Psi)$. Note that for all $\alpha' \in K_s$ we have $\eta(\alpha') = \pi_2$, this follows from the fact that η is HI in $K \setminus (\Phi \cup \Psi)$.

Figure 2.4 provides a graphic representation of this description. The figure shows the set $\mathsf{Paths}^\star(\Pi)$ of all finite paths of Π, the cut K of $\mathsf{Paths}^\star(\Pi)$, the path ρ reaching s (in red and dotted border line style), a path α reaching s in K (in blue and continuous border line style). The fact that η takes different decisions ρ and α is represented by the different colors and line style of their respective last states s.

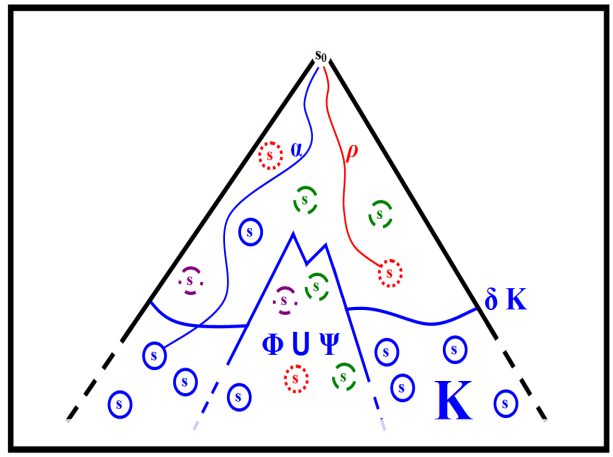


Figure 2.4: Graphic representation of $\mathsf{Paths}^\star(\Pi)$, $\Phi \cup \Psi$, K, δK, ρ, and α.

We now define two schedulers η_1 and η_2 such that they are HI in $(K \cup \{\rho\}) \setminus (\Phi \cup \Psi)$. Both η_1 and η_2 are the same than η everywhere but in K_s and ρ, respectively. The first one selects π_1 for all $\alpha \in K_s$ (instead of π_2 as η does), and the second scheduler selects π_2 in ρ (instead of π_1):

$$\eta_1(\sigma) = \begin{cases} \pi_1 & \text{if } \sigma \in K_s \\ \eta(\sigma) & \text{otherwise} \end{cases} \qquad \text{and} \qquad \eta_2(\sigma) = \begin{cases} \pi_2 & \text{if } \sigma = \rho \\ \eta(\sigma) & \text{otherwise.} \end{cases}$$

Now we plan to prove that either η_1 is "better" than η or η_2 is "better" than η. In order to prove this result, we will show that:

$$\mathbb{P}_{\eta_1}[\Diamond\phi|\Diamond\psi] \leq \mathbb{P}_{\eta_2}[\Diamond\phi|\Diamond\psi] \Longleftrightarrow \mathbb{P}_{\eta}[\Diamond\phi|\Diamond\psi] \leq \mathbb{P}_{\eta_2}[\Diamond\phi|\Diamond\psi] \tag{2.8}$$

and

$$\mathbb{P}_{\eta_2}[\Diamond\phi|\Diamond\psi] \leq \mathbb{P}_{\eta_1}[\Diamond\phi|\Diamond\psi] \Longleftrightarrow \mathbb{P}_{\eta}[\Diamond\phi|\Diamond\psi] \leq \mathbb{P}_{\eta_1}[\Diamond\phi|\Diamond\psi] \tag{2.9}$$

Therefore, if $\mathbb{P}_{\eta_1}[\Diamond\phi|\Diamond\psi] \leq \mathbb{P}_{\eta_2}[\Diamond\phi|\Diamond\psi]$ then we have $(K \cup \{\rho\}, \eta_2) \in \mathcal{K}$, and otherwise $(K \cup \{\rho\}, \eta_1) \in \mathcal{K}$. So, the desired result follows from (2.8) and (2.9). We will prove (2.8), the other case follows the same way.

In order to prove (2.8) we need to analyze more closely the conditional probability $\mathbb{P}[\Diamond\phi|\Diamond\psi] \triangleq \mathbb{P}[\Phi|\Psi]$ for each of the schedulers η, η_1, and η_2. For that purpose we partition the sets $\Phi \cap \Psi$ and Ψ into four *parts*, i.e. disjoint sets. The plan is to partition $\Phi \cap \Psi$ and Ψ in such way that we can make use of the fact that η, η_1, and η_2 are similar to each other (they only differ in the decision taken in K_s or ρ) obtaining, in this way, that the probabilities of the parts are the same under these schedulers or differ only by a factor (this intuition will become clearer later on in the proof), such condition is the key element of our proof of (2.8). Let us start by partitioning Ψ:

i) We define $\Psi_{\overline{\rho},\overline{k}_s}$ as the set of paths in Ψ neither passing through K_s nor ρ, formally
$$\Psi_{\overline{\rho},\overline{k}_s} \triangleq \Psi \setminus (\langle K_s \rangle \cup \langle \rho \rangle)$$

ii) We define $\Psi_{\rho,\overline{k}_s}$ as the set of paths in Ψ passing through ρ but not through K_s, i.e.:
$$\Psi_{\rho,\overline{k}_s} \triangleq \Psi \cap (\langle \rho \rangle \setminus \langle K_s \rangle).$$

iii) We define Ψ_{ρ,k_s} as the set of paths in Ψ passing through ρ and K_s, i.e.:
$$\Psi_{\rho,k_s} \triangleq \Psi \cap \langle \rho \rangle \cap \langle K_s \rangle.$$

iv) We define $\Psi_{\overline{\rho},k_s}$ as the set of paths in Ψ passing through K_s but not through ρ, i.e.:

$$\Psi_{\overline{\rho},k_s} \triangleq \Psi \cap (\langle K_s \rangle \setminus \langle \rho \rangle).$$

Note that $\Psi = \Psi_{\rho,s} \cup \Psi_{\rho,\overline{k}_s} \cup \Psi_{\overline{\rho},k_s} \cup \Psi_{\overline{\rho},\overline{k}_s}$.

Similarly, we can partition the set of paths $\Phi \cap \Psi$ into four parts obtaining $\Phi \cap \Psi = (\Phi \cap \Psi)_{\rho,s} \cup (\Phi \cap \Psi)_{\rho,\overline{k}_s} \cup (\Phi \cap \Psi)_{\overline{\rho},k_s} (\Phi \cap \Psi)_{\overline{\rho},\overline{k}_s}$.

In the following we analyze the probabilities (under η) of each part separately.

- The probability of $\Psi_{\rho,\overline{k}_s}$ can be written as $p_\rho \cdot x_\psi$, where p_ρ is the probability of ρ and x_ψ is the probability of reaching ψ without passing through K_s given ρ. More formally, $\mathbb{P}_\eta[\Psi_{\rho,\overline{k}_s}] = \mathbb{P}_\eta[\Psi \cap (\langle \rho \rangle \setminus \langle K_s \rangle)]$ $= \mathbb{P}_\eta[\langle \rho \rangle] \cdot \mathbb{P}_\eta[\Psi \cap (\langle \rho \rangle \setminus \langle K_s \rangle) | \langle \rho \rangle] \triangleq p_\rho \cdot x_\psi$.
- The probability of Ψ_{ρ,k_s} can be written as $p_\rho \cdot x_s \cdot \frac{y_\psi}{1-y_s}$, where x_s is the probability of passing through K_s given ρ, y_ψ is the probability of, given α, reaching ψ without passing through K_s after α; and y_s is the probability of, given α, passing through K_s again. Remember that α is any path in K_s. Formally, we have

$$\begin{aligned} \mathbb{P}_\eta[\Psi_{\rho,k_s}] &= \mathbb{P}_\eta[\Psi \cap \langle \rho \rangle \cap \langle K_s \rangle] \\ &= \mathbb{P}_\eta[\langle \rho \rangle] \cdot \mathbb{P}_\eta[\langle K_s \rangle | \langle \rho \rangle] \cdot \mathbb{P}_\eta[\Psi | \langle K_s \rangle \cap \langle \rho \rangle] \\ &= p_\rho \cdot x_s \cdot \mathbb{P}_\eta[\Psi | \langle \alpha \rangle]. \end{aligned}$$

Furthermore,

$$\begin{aligned} \mathbb{P}_\eta[\Psi | \langle \alpha \rangle] &= \mathbb{P}_\eta[\Psi | \overline{K}_{s \text{ again}} \cap \langle \alpha \rangle] \\ &= \frac{\mathbb{P}_\eta[\Psi \cap \overline{K}_{s \text{ again}} | \langle \alpha \rangle]}{\mathbb{P}_\eta[\overline{K}_{s \text{ again}} | \langle \alpha \rangle]} \\ &= \frac{y_\psi}{1-y_s}. \end{aligned}$$

where $\overline{K}_{s \text{ again}} \triangleq \langle \alpha \rangle \setminus \{\omega \in \langle \alpha\sigma \rangle \mid \alpha\sigma \in K_s\}$.

- The probability of $\Psi_{\overline{\rho},k_s}$ can be written as $p_{k_s} \cdot \frac{y_\psi}{1-y_s}$, where p_{k_s} is the probability of passing though K_s without passing through ρ. Formally, $\mathbb{P}_\eta[\Psi_{\overline{\rho},k_s}] = \mathbb{P}_\eta[\Psi \cap (\langle K_s \rangle \setminus \langle \rho \rangle)] = \mathbb{P}_\eta[\langle K_s \rangle \setminus \langle \rho \rangle] \cdot \mathbb{P}_\eta[\Psi | \langle \alpha \rangle] \triangleq p_{s_k} \cdot \frac{y_\psi}{1-y_s}$

- Finally, we write the probability of $\Psi_{\overline{\rho}\overline{k}_s}$ as p_ψ.

A similar reasoning can be used to analyze the probabilities associated to the parts of $\Phi \cap \Psi$. In this way we obtain that (1) $\mathbb{P}_\eta[(\Phi \cap \Psi)_{\rho,\overline{k}_s}] = p_\rho \cdot x_{\phi\psi}$, where $x_{\phi\psi}$ is the probability of reaching ϕ and ψ without passing through K_s given ρ, (2) $\mathbb{P}_\eta[(\Phi \cap \Psi)_{\rho,k_s}] = p_\rho \cdot x_s \cdot \frac{y_{\phi\psi}}{1-y_s}$, where $y_{\phi\psi}$ is the probability of reaching ϕ and ψ without passing through K_s afterwards given α, (3) $\mathbb{P}_\eta[(\Phi \cap \Psi)_{\overline{\rho},k_s}] = p_{k_s} \cdot \frac{y_{\phi\psi}}{1-y_s}$, and (4) $\mathbb{P}_\eta[(\Phi \cap \Psi)_{\overline{\rho},\overline{k}_s}] = p_{\phi\psi}$.

In order to help the intuition of the reader, we now provide a graphical representation of the probability (under η) of the sets $\Phi \cap \Psi$ and Ψ by means of a Markov chain (see Figure 2.5). The missing values are defined as $p_{\overline{\phi}\psi} \triangleq p_\psi - p_{\phi\psi}$, $p_\emptyset \triangleq 1 - p_{s_k} - p_\rho - p_\psi$; and similarly for $x_{\overline{\phi}\psi}$, $x_\emptyset$, $y_{\overline{\phi}\psi}$, and $y_\emptyset$. Furthermore, absorbing states $\phi\psi$ denote states where $\phi \wedge \psi$ holds, absorbing states $\overline{\phi}\psi$ denote states where $\neg\phi \wedge \psi$ holds, and $\overline{\psi}$ denote a state where $\neg\psi$ holds. Finally, the state ρ represents the state of the model where ρ has been just reached and α a state where any of the paths α in K_s as been just reached. To see how this Markov Chain is related to the probabilities of $\Phi \cap \Psi$ and Ψ on the original MDP consider, for example, the probabilities of the set $\Phi \cap \Psi$. It is easy to show that

$$\begin{aligned}\mathbb{P}_\eta[\Phi \cap \Psi] &= \mathbb{P}_\eta[\Phi_{\rho,k_s}] + \mathbb{P}_\eta[\Phi_{\overline{\rho},k_s}] + \mathbb{P}_\eta[\Phi_{\rho,\overline{k}_s}] + \mathbb{P}_\eta[\Phi_{\overline{\rho},\overline{k}_s}] \\ &= p_{\phi\psi} + p_\rho \cdot x_{\phi\psi} + p_\rho \cdot x_s \cdot \tfrac{y_{\phi\psi}}{1-y_s} + p_{s_k} \cdot \tfrac{y_{\phi\psi}}{1-y_s} = \mathbb{P}_M[\Diamond\phi\psi].\end{aligned}$$

We note that the values p_{s_k}, p_ρ, $p_{\phi\psi}$, $p_{\overline{\phi}\psi}$, and $p_\emptyset$ coincide for η, η_1, and η_2. Whereas the values $\vec{x} \triangleq (x_s, x_\psi, x_{\phi\psi}, x_\emptyset)$ coincide for η and η_1 and the values $\vec{y} \triangleq (y_s, y_\psi, y_{\phi\psi}, y_\emptyset)$ coincide for η and η_2. Thus, the variant of M_η in which $\vec{y}$ is replaced by $\vec{x}$ describes the probability of each partition under the scheduler η_1 instead of η. Similarly, the variant on which $\vec{x}$ is replaced by $\vec{y}$ represents the probability of each partition under the scheduler η_2.

Now we have all the ingredients needed to prove (2.8). Our plan is to show that:

1) $\mathbb{P}_{\eta_1}[\Phi|\Psi] \leq \mathbb{P}_{\eta_2}[\Phi|\Psi] \iff (p_\rho + p_{s_k}) \cdot d \leq 0 \iff d \leq 0$, and

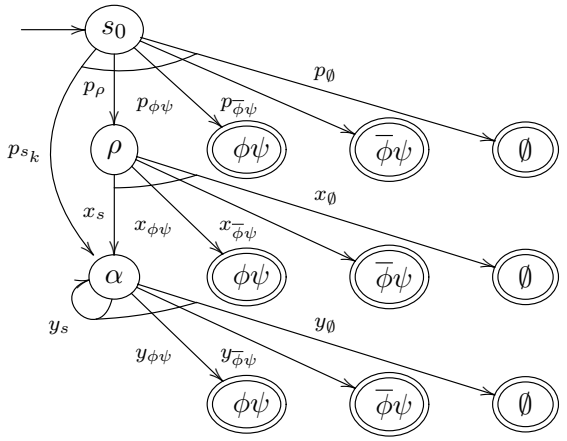


Figure 2.5: Graphical representation of how we write the probability of each partition: M_η.

2) $\mathbb{P}_\eta[\Phi|\Psi] \leq \mathbb{P}_{\eta_2}[\Phi|\Psi] \iff (1-y_s)\cdot p_\rho \cdot d \leq 0 \iff d \leq 0.$

where d is the following determinant

$$d = \begin{vmatrix} p_s + p_{s_k} & x_s - 1 & y_s - 1 \\ p_{\phi\psi} & x_{\phi\psi} & y_{\phi\psi} \\ p_\psi & x_\psi & y_\psi \end{vmatrix}.$$

We now proceed to prove 1)

$$\begin{gathered}
\mathbb{P}_{\eta_1}[\Phi|\Psi] - \mathbb{P}_{\eta_2}[\Phi|\Psi] \leq 0 \\
\iff \\
\frac{p_{\phi\psi}+p_\rho\cdot x_{\phi\psi}+(p_\rho\cdot x_s+p_{s_k})\cdot\frac{x_{\phi\psi}}{1-x_s}}{p_\psi+p_\rho\cdot x_\psi+(p_\rho\cdot x_s+p_{s_k})\cdot\frac{x_\psi}{1-x_s}} - \frac{p_{\phi\psi}+p_\rho\cdot y_{\phi\psi}+(p_\rho\cdot y_s+p_{s_k})\cdot\frac{y_{\phi\psi}}{1-y_s}}{p_\psi+p_\rho\cdot y_\psi+(p_\rho\cdot y_s+p_{s_k})\cdot\frac{y_\psi}{1-y_s}} \leq 0 \\
\iff \\
\frac{p_{\phi\psi}(1-x_s)+p_\rho x_{\phi\psi}+p_{s_k}x_{\phi\psi}}{p_\psi(1-x_s)+p_\rho x_\psi+p_{s_k}x_\psi} - \frac{p_{\phi\psi}(1-y_s)+p_s y_{\phi\psi}+p_{s_k}y_{\phi\psi}}{p_\psi(1-y_s)+p_\rho y_\psi+p_{s_k}y_\psi} \leq 0 \\
\iff \\
\begin{vmatrix} p_{\phi\psi}(1-x_s)+p_\rho x_{\phi\psi}+p_{s_k}x_{\phi\psi} & p_{\phi\psi}(1-y_s)+p_\rho y_{\phi\psi}+p_{s_k}y_{\phi\psi} \\ p_\psi(1-x_s)+p_\rho x_\psi+p_{s_k}x_\psi & p_\psi(1-y_s)+p_\rho y_\psi+p_{s_k}y_\psi \end{vmatrix} \leq 0.
\end{gathered}$$

A long but straightforward computation shows that the 2x2 determinant in the line above is equal to $(p_\rho + p_{s_k})d$.

The proof of 2) proceeds along the same lines.

$$\begin{gathered}
\mathbb{P}_\eta[\Phi|\Psi] - \mathbb{P}_{\eta 2}[\Phi|\Psi] \leq 0 \\
\Longleftrightarrow \\
\frac{p_{\phi\psi}+p_\rho\cdot x_{\phi\psi}+(p_\rho\cdot x_s+p_{s_k})\cdot\frac{y_{\phi\psi}}{1-y_s}}{p_\psi+p_\rho\cdot x_\psi+(p_\rho\cdot x_s+p_{s_k})\cdot\frac{y_\psi}{1-y_s}} - \frac{p_{\phi\psi}+p_\rho\cdot y_{\phi\psi}+(p_\rho\cdot y_s+p_{s_k})\cdot\frac{y_{\phi\psi}}{1-y_s}}{p_\psi+p_\rho\cdot y_\psi+(p_\rho\cdot y_s+p_{s_k})\cdot\frac{y_\psi}{1-y_s}} \leq 0 \\
\Longleftrightarrow \\
\frac{p_{\phi\psi}(1-y_s)+p_\rho x_{\phi\psi}(1-y_s)+(p_\rho x_s+p_{s_k})y_{\phi\psi}}{p_\psi(1-y_s)+p_\rho x_\psi(1-y_s)+(p_\rho x_s+p_{s_k})y_\psi} - \frac{p_{\phi\psi}(1-y_s)+p_\rho y_{\phi\psi}+p_{s_k}y_{\phi\psi}}{p_\psi(1-y_s)+p_\rho y_\psi+p_{s_k}y_\psi} \leq 0 \\
\Longleftrightarrow
\end{gathered}$$

$$\begin{gathered}
\begin{vmatrix} p_{\phi\psi}(1-y_s)+p_\rho x_{\phi\psi}(1-y_s)+(p_\rho x_s+p_{s_k})y_{\phi\psi} & p_{\phi\psi}(1-y_s)+p_\rho y_{\phi\psi}+p_{s_k}y_{\phi\psi} \\ p_\psi(1-y_s)+p_\rho x_\psi(1-y_s)+(p_\rho x_s+p_{s_k})y_\psi & p_\psi(1-y_s)+p_\rho y_\psi+p_{s_k}y_\psi \end{vmatrix} \\
\leq 0
\end{gathered}$$

and also here a long computation shows that this last 2x2 determinant is equal to $(1-y_s)\cdot p_\rho \cdot d$. □

Finally, we have all the ingredients needed to prove that there exists a scheduler close to the supremum which is HI before the stopping condition.

Corollary 2.5.1. [HI before stopping condition] Let Π be a MDP, $\phi, \psi \in \mathsf{Path}$. Then for all $\hat{\epsilon} > 0$, there exists a scheduler $\eta^\star$ such that $\mathbb{P}^+[\phi|\psi] - \mathbb{P}_{\eta^\star}[\phi|\psi] < \hat{\epsilon}$ and $\eta^\star$ is history independent before the stopping condition.

Proof. Follows directly from Lemma 2.5.5 and Lemma 2.5.6. □

We now proceed with the construction of a maximizing scheduler and HI after the stopping condition.

Lemma 2.5.7. [HI after stopping condition] Let Π be a MDP, $\phi, \psi \in \mathsf{Path}$, and $\varphi = \mathsf{StopC}(\phi) \vee \mathsf{StopC}(\psi)$. Then for all schedulers η there exists a scheduler $\eta^\star$ such that

1) $\eta^\star$ behaves like η before the stopping condition,
2) $\eta^\star$ is HI after the stopping condition φ, and
3) $\mathbb{P}_\eta[\phi|\psi] \leq \mathbb{P}_{\eta^\star}[\phi|\psi]$.

Proof. We will prove this result for the case in which ϕ is of the form $\Diamond\phi$ and ψ is of the form $\Diamond\psi$, the proof for the remaining cases follows in the same way.

Let us start by introducing some notation. We define, respectively, the set of paths reaching ϕ, the set of paths not reaching ϕ, the set of paths reaching ϕ without reaching ψ before, and the set of paths reaching $\psi \wedge \neg\phi$ without reaching ϕ before as follows

$$\begin{aligned}
\Delta_\phi &\triangleq \{\omega \in \mathsf{Paths}(\Pi) \mid \omega \models \Diamond\phi\},\\
\Delta_{\neg\phi} &\triangleq \{\omega \in \mathsf{Paths}(\Pi) \mid \omega \models \Box\neg\phi\},\\
\Delta_{\overline{\psi}\phi} &\triangleq \{\omega \in \mathsf{Paths}(\Pi) \mid \omega \models \neg\psi\mathcal{U}\phi\},\\
\Delta_{\overline{\phi}\psi} &\triangleq \{\omega \in \mathsf{Paths}(\Pi) \mid \omega \models \neg\phi\mathcal{U}(\psi \wedge \neg\phi)\}.
\end{aligned}$$

Note that the last two sets are disjoint. It is easy to check that

$$\Delta_\phi \cap \Delta_\psi = (\Delta_{\overline{\psi}\phi} \cap \Delta_\psi) \cup (\Delta_{\overline{\phi}\psi} \cap \Delta_\phi),$$
$$\Delta_\psi = \Delta_{\overline{\phi}\psi} \cup (\Delta_{\overline{\psi}\phi} \cap \Delta_\psi) = [(\Delta_{\overline{\phi}\psi} \cap \Delta_\phi) \cup (\Delta_{\overline{\phi}\psi} \cap \Delta_{\neg\phi})] \cup (\Delta_{\overline{\psi}\phi} \cap \Delta_\psi).$$

Let us now define the minimal set of finite paths "generating" (by their basic cylinders) $\Delta_{\overline{\psi}\phi}$ and $\Delta_{\overline{\phi}\psi}$: $K_{\overline{\psi}\phi} \triangleq \{\sigma \in \mathsf{Paths}^\star(\Pi) \mid \mathsf{last}(\sigma) \models \phi \wedge \forall\, i < |\sigma| \,:\, \sigma_i \models \neg\phi \wedge \neg\psi\}$ and similarly $K_{\overline{\phi}\psi} \triangleq \{\sigma \in \mathsf{Paths}^\star(\Pi) \mid \mathsf{last}(\sigma) \models (\psi \wedge \neg\phi) \wedge \forall\, i < |\sigma| \,:\, \sigma_i \models \neg\phi \wedge \neg\psi\}$. Note that $\Delta_{\overline{\psi}\phi} = \langle K_{\overline{\psi}\phi}\rangle$ and $\Delta_{\overline{\phi}\psi} = \langle K_{\overline{\phi}\psi}\rangle$. Now we can write

$$\mathbb{P}_\eta[\Diamond\phi|\Diamond\psi] = \frac{\mathbb{P}_\eta[\langle K_{\overline{\psi}\phi}\rangle \cap \Delta_\psi] + \mathbb{P}_\eta[\langle K_{\overline{\phi}\psi}\rangle \cap \Delta_\phi]}{\mathbb{P}_\eta[\langle K_{\overline{\psi}\phi}\rangle \cap \Delta_\psi] + \mathbb{P}_\eta[\langle K_{\overline{\phi}\psi}\rangle \cap \Delta_\phi] + \mathbb{P}_\eta[\langle K_{\overline{\phi}\psi}\rangle \cap \Delta_{\neg\phi}]}.$$

The construction of the desired scheduler $\eta^\star$ is in the spirit of the construction we proposed for the scheduler in Lemma 2.5.5. We let $S_\phi \triangleq \{s \in S \mid$

$s \models \phi\}$ and $S_\psi \triangleq \{s \in S \mid s \models (\psi \wedge \neg\phi)\}$. Note that S_ϕ and S_ψ are disjoint. Now we define two families of schedulers $\{\xi_s\}_{s\in S_\phi}$ and $\{\zeta_s\}_{s\in S_\psi}$ such that: for all $s_1, s_2 \in S_\phi$ we have $\mathbb{P}_{s_1,\xi_{s_1}}[\Diamond\psi] = \mathbb{P}^+_{s_1}[\Diamond\psi]$, $\mathbb{P}_{s_2,\xi_{s_2}}[\Diamond\psi] = \mathbb{P}^+_{s_2}[\Diamond\psi]$, and for all $\sigma_1 t \in \mathsf{Paths}^\star(s_1)$, $\sigma_2 t \in \mathsf{Paths}^\star(s_2)$ we have $\xi_{s_1}(\sigma_1 t) = \xi_{s_2}(\sigma_2 t)$. Similarly for $\{\zeta_s\}_{s\in S_\psi}$: for all $s_1, s_2 \in S_\phi$ we have $\mathbb{P}_{s_1,\zeta_{s_1}}[\Diamond\phi] = \mathbb{P}^+_{s_1}[\Diamond\phi]$, $\mathbb{P}_{s_2,\zeta_{s_2}}[\Diamond\phi] = \mathbb{P}^+_{s_2}[\Diamond\phi]$, and for all $\sigma_1 t \in \mathsf{Paths}^\star(s_1)$, $\sigma_2 t \in \mathsf{Paths}^\star(s_2)$ we have $\zeta_{s_1}(\sigma_1 t) = \zeta_{s_2}(\sigma_2 t)$.

We now proceed to define $\eta^\star$:

$$\eta^\star(\sigma) \triangleq \begin{cases} \xi_s(\sigma_{|\alpha|}\cdots\sigma_{|\sigma|}) & \text{if } \alpha \sqsubseteq \sigma \text{ for some } \alpha \in K_\phi \text{ such that } \mathsf{last}(\alpha) = s, \\ \zeta_s(\sigma_{|\alpha|}\cdots\sigma_{|\sigma|}) & \text{if } \alpha \sqsubseteq \sigma \text{ for some } \alpha \in K_\psi \text{ such that } \mathsf{last}(\alpha) = s, \\ \eta(\sigma) & \text{otherwise.} \end{cases}$$

where $K_\phi \triangleq \{\sigma \in \mathsf{Paths}^\star \mid \mathsf{last}(\sigma) \in S_\phi\}$, and similarly $K_\psi \triangleq \{\sigma \in \mathsf{Paths}^\star \mid \mathsf{last}(\sigma) \in S_\psi\}$.

It is easy to check that $\eta^\star$ satisfies 1) and 2). As for 3) we first note that $\mathbb{P}_\eta[\langle K^-_{\overline{\psi}\phi}\rangle \cap \Psi] \leq \mathbb{P}_{\eta^\star}[\langle K^-_{\overline{\psi}\phi}\rangle \cap \Psi]$, $\mathbb{P}_\eta[\langle K^-_{\overline{\phi}\psi}\rangle \cap \Delta_\phi] \leq \mathbb{P}_{\eta^\star}[\langle K^-_{\overline{\phi}\psi}\rangle \cap \Delta_\phi]$, and $\mathbb{P}_\eta[\langle K^-_{\overline{\phi}\psi}\rangle \cap \Delta_{\neg\phi}] \geq \mathbb{P}_{\eta^\star}[\langle K^-_{\overline{\phi}\psi}\rangle \cap \Delta_{\neg\phi}]$.

In addition, we need the following simple remark.

Remark 2.5.8. Let $f : \mathbb{R} \to \mathbb{R}$ be a function defined as $f(x) \triangleq \frac{a+x}{b+x}$ where a and b are constants in the interval $[0, 1]$ such that $b \geq a$. Then f is increasing.

Finally, we have

$$\begin{aligned} \mathbb{P}_\eta[\Diamond\phi \mid \Diamond\psi] &= \frac{\mathbb{P}_\eta[\langle K^-_{\overline{\psi}\phi}\rangle \cap \Delta_\psi] + \mathbb{P}_\eta[\langle K^-_{\overline{\phi}\psi}\rangle \cap \Delta_\phi]}{\mathbb{P}_\eta[\langle K^-_{\overline{\psi}\phi}\rangle \cap \Delta_\psi] + \mathbb{P}_\eta[\langle K^-_{\overline{\phi}\psi}\rangle \cap \Delta_\phi] + \mathbb{P}_\eta[\langle K^-_{\overline{\phi}\psi}\rangle \cap \Delta_{\neg\phi}]} \\ &\qquad \{\text{by Remark 2.5.8}\} \\ &\leq \frac{\mathbb{P}_{\eta^\star}[\langle K^-_{\overline{\psi}\phi}\rangle \cap \Delta_\psi] + \mathbb{P}_\eta[\langle K^-_{\overline{\phi}\psi}\rangle \cap \Delta_\phi]}{\mathbb{P}_{\eta^\star}[\langle K^-_{\overline{\psi}\phi}\rangle \cap \Delta_\psi] + \mathbb{P}_\eta[\langle K^-_{\overline{\phi}\psi}\rangle \cap \Delta_\phi] + \mathbb{P}_\eta[\langle K^-_{\overline{\phi}\psi}\rangle \cap \Delta_{\neg\phi}]} \\ &\qquad \{\text{by Remark 2.5.8}\} \end{aligned}$$

$$\leq \frac{\mathbb{P}_{\eta^\star}[\langle K^{-}_{\overline{\psi}\phi}\rangle \cap \Delta_\psi] + \mathbb{P}_{\eta^\star}[\langle K^{-}_{\overline{\phi}\psi}\rangle \cap \Delta_\phi]}{\mathbb{P}_{\eta^\star}[\langle K^{-}_{\overline{\psi}\phi}\rangle \cap \Delta_\psi] + \mathbb{P}_{\eta^\star}[\langle K^{-}_{\overline{\phi}\psi}\rangle \cap \Delta_\phi] + \mathbb{P}_{\eta}[\langle K^{-}_{\overline{\phi}\psi}\rangle \cap \Delta_{\neg\phi}]}$$

$$\leq \frac{\mathbb{P}_{\eta^\star}[\langle K^{-}_{\overline{\psi}\phi}\rangle \cap \Delta_\psi] + \mathbb{P}_{\eta^\star}[\langle K^{-}_{\overline{\phi}\psi}\rangle \cap \Delta_\phi]}{\mathbb{P}_{\eta^\star}[\langle K^{-}_{\overline{\psi}\phi}\rangle \cap \Delta_\psi] + \mathbb{P}_{\eta^\star}[\langle K^{-}_{\overline{\phi}\psi}\rangle \cap \Delta_\phi] + \mathbb{P}_{\eta^\star}[\langle K^{-}_{\overline{\phi}\psi}\rangle \cap \Delta_{\neg\phi}]}$$

$$= \mathbb{P}_{\eta^\star}[\Diamond\phi|\Diamond\psi] \qquad \square$$

Proof of Theorem 2.5.3. It follows straightforwardly from Corollary 2.5.1 and Lemma 2.5.7. $\square$

2.5.2 Deterministic Schedulers

We now proceed to show that deterministic schedulers suffice to attain optimal conditional probabilities.

The following result states that taking the convex combination of schedulers does not increase the conditional probability $\mathbb{P}[\phi|\psi]$.

Lemma 2.5.9. Let Π be a MDP, s a state, and ϕ, ψ path formulas. Suppose that the s-scheduler η is a convex combination of η_1 and η_2. Then $\mathbb{P}_{s,\eta}[\phi|\psi] \leq \mathsf{max}(\mathbb{P}_{s,\eta_1}[\phi|\psi], \mathbb{P}_{s,\eta_2}[\phi|\psi])$.

Proof. To prove this lemma we need to use the following technical result: The function $f\colon \mathbb{R} \to \mathbb{R}$ defined as below is monotonous.

$$f(x) \triangleq \frac{xv_1 + (1-x)v_2}{xw_1 + (1-x)w_2}$$

where $v_1, v_2 \in [0, \infty)$ and $w_1, w_2 \in (0, \infty)$. This claim follows from the fact that $f'(x) = \frac{v_1 w_2 - v_2 w_1}{(xw_1 - (1-x)w_2)^2}$ is always ≥ 0 or always ≤ 0.

Now, by applying the result above to

$$[0,1] \ni \alpha \mapsto \frac{\alpha\mathbb{P}_{s,\eta_1}[\phi \wedge \psi] + (1-\alpha)\mathbb{P}_{s,\eta_2}[\phi \wedge \psi]}{\alpha\mathbb{P}_{s,\eta_1}[\psi] + (1-\alpha)\mathbb{P}_{s,\eta_2}[\psi]}$$

we get that the maximum is reached at $\alpha = 0$ or $\alpha = 1$. Because η is a convex combination of η_1 and η_2, $\mathbb{P}_{s,\eta}[\phi|\psi] \leq \mathbb{P}_{s,\eta_2}[\phi|\psi]$ (in the first case) or $\mathbb{P}_{s,\eta}[\phi|\psi] \leq \mathbb{P}_{s,\eta_1}[\phi|\psi]$ (in the second case). $\square$

Lemma 2.5.10. Let Π be a MDP, s a state, and φ a state formula. Then every φ-sHI s-scheduler on Π is a convex combination of deterministic φ-sHI s-schedulers.

Proof. The result follows from the fact that sHI schedulers have only finitely many choices to make at each state (at most two) and every choice at a particular state – either before or after the stopping condition– is a convex combination of deterministic choices at that state – either before or after the stopping condition. □

Finally, combining Theorem 2.5.3 and the previous lemma we obtain:

Theorem 2.5.11. Let Π be a MDP, $\phi, \psi \in \mathsf{Path}$, and $\varphi = \mathsf{StopC}(\phi) \vee \mathsf{StopC}(\psi)$. Then we have

$$\mathbb{P}^{+}[\phi|\psi] = \sup_{\eta \in \mathrm{Sch}_d^{\varphi}(\Pi)} \mathbb{P}_{\eta}[\phi|\psi],$$

where $\mathrm{Sch}_d^{\varphi}(\Pi)$ is the set of deterministic and φ-sHI schedulers of Π.

Since the number of deterministic and semi HI schedulers is finite we know that there exists a scheduler attaining the optimal conditional probability, i.e. $\sup_{\eta \in \mathrm{Sch}_d^{\varphi}(\Pi)} \mathbb{P}_{\eta}[\phi|\psi] = \max_{\eta \in \mathrm{Sch}_d^{\varphi}(\Pi)} \mathbb{P}_{\eta}[\phi|\psi]$. Note that this implies that cpCTL is decidable.

We conclude this section showing that there exists a deterministic and semi HI scheduler maximizing the conditional probabilities of Example 2.5.1.

Example 2.5.12. Consider the MDP and cpCTL formula of Example 2.5.1. According to Theorem 2.5.11 there exists a deterministic and $(B \vee P)$-sHI scheduler that maximizes $\mathbb{P}_{s_0,\eta}[\Diamond B|\Diamond P]$. In this case, a maximizing scheduler will take always the same decision (π_3) before the system reaches s_3 (a state satisfying the until stopping condition $(B \vee P)$) and always the same decision (π_1) after the system reaches s_3.

2.6 Model Checking cpCTL

Model checking cpCTL means checking if a state s satisfies a certain state formula ϕ. We focus on formulas of the form $\mathbb{P}_{\leq a}[\phi|\psi]$ and show how to compute $\mathbb{P}^+_s[\phi|\psi]$. The case $\mathbb{P}^-_s[\phi|\psi]$ is similar.

Recall that model checking pCTL is based on the Bellman-equations. For instance, $\mathbb{P}^+_s[\Diamond B] = \mathsf{max}_{\pi\in\tau(s)} \sum_{t\in\mathsf{succ}(s)} \pi(t)\cdot\mathbb{P}^+_t[\Diamond B]$ whenever $s \not\models B$. So a scheduler η that maximizes $\mathbb{P}_s[\Diamond B]$ chooses $\pi \in \tau(s)$ maximizing $\sum_{t\in\mathsf{succ}(s)}\pi(t)\cdot\mathbb{P}^+_t[\Diamond B]$. In a successor state t, η still behaves as a scheduler that maximizes $\mathbb{P}_t[\Diamond B]$. As shown below, such a local Bellman-equation is not true for conditional probabilities: a scheduler that maximizes a conditional probability such as $\mathbb{P}_s[\Diamond B|\Box P]$ does not necessarily maximize $\mathbb{P}_t[\Diamond B|\Box P]$ for successors t of s.

Example 2.6.1. Consider the MDP and cpCTL formula $\mathbb{P}_{\leq a}[\Diamond B|\Box P]$ of Figure 2.1. There are only two deterministic schedulers. The first one, η_1, chooses π_2 when the system reaches the state s_2 and the second one, η_2, chooses π_3 when the system reaches s_2. For the first one $\mathbb{P}_{s_0,\eta_1}[\Diamond B|\Box P] = 1 - \frac{2\alpha}{7}$, and for the second one $\mathbb{P}_{s_0,\eta_2}[\Diamond B|\Box P] = \frac{30}{31}$. So $\mathbb{P}^+_{s_0}[\Diamond B|\Box P] = \mathsf{max}(1 - \frac{2\alpha}{7}, \frac{30}{31})$. Therefore, if $\alpha \geq \frac{7}{62}$ the scheduler that maximizes $\mathbb{P}_{s_0}[\Diamond B|\Box P]$ is η_2 ($\mathbb{P}_{s_0,\eta_2}[\Diamond B|\Box P] = \mathbb{P}^+_{s_0}[\Diamond B|\Box P]$) and otherwise it is η_1 ($\mathbb{P}_{s_0,\eta_1}[\Diamond B|\Box P] = \mathbb{P}^+_{s_0}[\Diamond B|\Box P]$).

Furthermore, $\mathbb{P}^+_{s_1}[\Diamond B|\Box P] = 1$ and $\mathbb{P}^+_{s_2}[\Diamond B|\Box P] = 1 - 2\alpha$; the scheduler that obtains this last maximum is the one that chooses π_2 in s_2.

Thus, if $\alpha \geq \frac{7}{62}$ the scheduler that maximizes the conditional probability from s_0 is taking a different decision than the one that maximize the conditional probability from s_2. Furthermore, $\mathsf{max}(1 - \frac{2\alpha}{7}, \frac{30}{31}) = \mathbb{P}^+_{s_0}[\Diamond B|\Box P] \neq \frac{3}{4}\mathbb{P}^+_{s_1}[\Diamond B|\Box P] + \frac{1}{4}\mathbb{P}^+_{s_2}[\Diamond B|\Box P] = 1 - \frac{\alpha}{2}$ for all $\alpha \in (0, 1]$, showing that the Bellman-equation from above does not generalize to cpCTL.

As consequence of this observation, it is not possible to "locally maximize" cpCTL properties (i.e. to obtain the global maximum $\mathbb{P}^+_{s_0}[\phi|\psi]$ by maximizing $\mathbb{P}_t[\phi|\psi]$ for all states t). This has a significant impact in terms of model-checking complexity: as we will show in the rest of this section, to verify a cpCTL property it is necessary to compute and keep track of

several conditional probabilities and the desired maximum value can only be obtained after all these probabilities have been collected.

2.6.1 Model Checking $\mathbb{P}_{\leq a}[\phi|\psi]$

An obvious way to compute $\mathbb{P}^+_s[\phi|\psi]$ is by computing the pairs $(\mathbb{P}_{s,\eta}[\phi \wedge \psi], \mathbb{P}_{s,\eta}[\psi])$ for all deterministic sHI schedulers η, and then taking the maximum quotient $\mathbb{P}_{s,\eta}[\phi \wedge \psi]/\mathbb{P}_{s,\eta}[\psi]$. This follows from the fact that there exist finitely many deterministic semi history-independent schedulers and that one of them attains the maximal conditional probability; however, the number of such schedulers grows exponentially in the size of the MDP so computing these pairs for all of them is computationally expensive. Our plan is to first present the necessary techniques to naively compute $(\mathbb{P}_{s,\eta}[\phi \wedge \psi], \mathbb{P}_{s,\eta}[\psi])$ for all deterministic sHI schedulers η and then present an algorithm that allows model checking $\mathbb{P}_{\leq a}[\phi|\psi]$ without collecting such pairs for all sHI scheduler.

1) A naive approach to compute $\mathbb{P}^+[\phi|\psi]$

The algorithm is going to keep track of a list of pairs of probabilities of the form $(\mathbb{P}_{t,\eta}[\phi \wedge \psi], \mathbb{P}_{t,\eta}[\psi])$ for all states t and η a deterministic sHI scheduler. We start by defining a data structure to keep track of the these pairs of probabilities.

Definition 2.6.2. Let L be the set of expressions of the form $(p_1, q_1) \vee \cdots \vee (p_n, q_n)$ where $p_i, q_i \in [0, \infty)$ and $q_i \geq p_i$, for all $n \in \mathbb{N}^\star$. On L we consider the smallest congruence relation $\equiv_1$ satisfying idempotence, commutativity, and associativity, i.e.:

$$\begin{aligned} (p_1, q_1) \vee (p_1, q_1) &\equiv_1 (p_1, q_1) \\ (p_1, q_1) \vee (p_2, q_2) &\equiv_1 (p_2, q_2) \vee (p_1, q_1) \\ ((p_1, q_1) \vee (p_2, q_2)) \vee (p_3, q_3) &\equiv_1 (p_1, q_1) \vee ((p_2, q_2) \vee (p_3, q_3)) \end{aligned}$$

Note that $(p_1, q_1) \vee \cdots \vee (p_n, q_n) \equiv_1 (p'_1, q'_1) \vee \cdots \vee (p'_{n'}, q'_{n'})$ if and only if $\{(p_1, q_1), \ldots, (p_n, q_n)\} = \{(p'_1, q'_1), \ldots, (p'_{n'}, q'_{n'})\}$.

We let L_1 be the set of equivalence classes and denote the projection map $L \to L_1$ that maps each expression to its equivalence class by f_1. On L we also define *maximum quotient* $\top : L \to [0, \infty)$ by

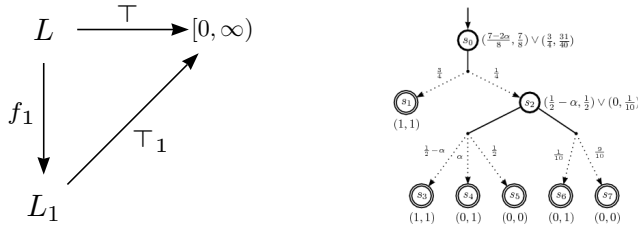
$$\top\left(\bigvee_{i=1}^{n}(p_i, q_i)\right) \triangleq \mathsf{max}\left(\left\{\frac{p_i}{q_i} | q_i \neq 0, i = 1, \ldots, n\right\} \cup \{0\}\right)$$

Note that $\top$ induces a map $\top_1 \colon L_1 \to [0, \infty)$ making the diagram in Figure 2.6 (a) commute, i.e., such that $\top_1 \circ f_1 = \top$.

Definition 2.6.3. Let Π be a MDP. We define the function $\delta : S \times \mathsf{Stat} \times \mathsf{Path} \times \mathsf{Path} \to L$ by

$$\delta(s, \varphi, \phi, \psi) \triangleq \bigvee_{\eta \in \mathsf{Sch}_d^{\varphi}(\Pi)} \left(\mathbb{P}_{s,\eta}[\phi \wedge \psi], \mathbb{P}_{s,\eta}[\psi]\right)$$

and we define $\delta_1 \colon S \times \mathsf{Stat} \times \mathsf{Path} \times \mathsf{Path} \to L_1$ by $\delta_1 \triangleq f_1 \circ \delta$.



(a) Commutative diagram (b) δ-values

Figure 2.6: Commutative diagram and δ-values.

When no confusion arises, we omit the subscripts 1 and omit the projection map f_1, writing $(p_1, q_1) \vee \cdots \vee (p_n, q_n)$ for the equivalence class it generates.

Example 2.6.4. In Figure 2.6 (b) we show the value $\delta(s, B \vee \neg P, \Diamond B, \Box P)$ associated to each state s of the MDP in Figure 2.1.

The following lemma states that it is possible to obtain maximum conditional probabilities using δ.

Lemma 2.6.5. Given $\Pi = (S, s_0, L, \tau)$ an acyclic MDP, and $\phi_1, \phi_2, \psi_1, \psi_2 \in \mathsf{Stat}$. Then

$$\mathbb{P}^+_s[\phi_1 \mathcal{U} \phi_2 | \psi_1 \mathcal{U} \psi_2] = \top\left(\delta^{\mathcal{U}}_s(\phi_1 \mathcal{U} \phi_2 \mid \psi_1 \mathcal{U} \psi_2)\right)$$

and

$$\mathbb{P}^+_s[\phi_1 \mathcal{U} \phi_2 | \Box \psi_1] = \top\left(\delta^{\Box}_s(\phi_1 \mathcal{U} \phi_2 \mid \Box \psi_1)\right),$$

where $\delta^{\mathcal{U}}_s(\phi_1 \mathcal{U} \phi_2 \mid \psi_1 \mathcal{U} \psi_2) \triangleq \delta(s, \mathsf{StopC}(\phi_1 \mathcal{U} \phi_2) \vee \mathsf{StopC}(\psi_1 \mathcal{U} \psi_2), \phi_1 \mathcal{U} \phi_2, \psi_1 \mathcal{U} \psi_2)$ and $\delta^{\Box}_s(\phi_1 \mathcal{U} \phi_2 \mid \Box \psi_1) \triangleq \delta(s, \mathsf{StopC}(\phi_1 \mathcal{U} \phi_2) \vee \mathsf{StopC}(\Box \psi_1), \phi_1 \mathcal{U} \phi_2, \Box \psi_1)$.

Proof. The lemma follows straightforwardly from the definitions of δ and $\top$ and the fact that the maximum conditional probability is indeed reached by a deterministic sHI scheduler. □

Remember that there are finitely many sHI schedulers. Thus, δ (and therefore $\mathbb{P}^+[-|-]$) can in principle be computed by explicitly listing them all. However, this is of course an inefficient way to compute maximum conditional probabilities.

We now show how to compute $\mathbb{P}^+[-|-]$ in a more efficient way. We will first provide an algorithm to compute maximum conditional probabilities for acyclic MDPs. We then show how to apply this algorithm to MDPs with cycles by mean of a technique, based on SCC analysis, that allows the transformation of an MDP with cycles to an equivalent acyclic MDP.

2) An algorithm to compute $\mathbb{P}^+[\phi|\psi]$ for Acyclic MDPs

We will now present a recursive algorithm to compute $\mathbb{P}^+[\phi|\psi]$ for acyclic MDPs using a variant of δ (changing its image). As we mentioned before, to compute maximum conditional probabilities it is not necessary to consider all the pairs $(\mathbb{P}_\eta[\phi \wedge \psi], \mathbb{P}_\eta[\psi])$ (with η a deterministic and semi HI scheduler). In particular, we will show that it is sufficient to consider only deterministic and semi HI schedulers (see definition of D below) that behave as an optimizing scheduler (i.e. either maximizing or minimizing

a pCTL formula ϕ) after reaching the stopping condition (i.e. a state s satisfying $\mathsf{StopC}(\varphi)$).

We plan to compute a function $\hat{\delta}(-) \subseteq \delta(-)$ such that $\top(\delta) = \top(\hat{\delta})$. Intuitively, $\hat{\delta}(-)$ can be thought as

$$\hat{\delta}(s, \varphi, \phi, \psi) = \bigvee_{\eta \in D} \left(\mathbb{P}_{s,\eta}[\phi \wedge \psi], \mathbb{P}_{s,\eta}[\psi]\right)$$

where D contains all deterministic and semi HI schedulers η such that η optimizes $\mathbb{P}_{s,\eta}[\phi]$ for some $s \models \mathsf{StopC}(\varphi)$ and $\phi \in$ pCTL formula.

This intuition will become evident when we present our recursive algorithm to compute conditional probabilities (see Theorem 2.6.11 below). The states s involved in the definition of D correspond to the base case of the algorithm and the formula ϕ corresponds to the formula that the algorithm maximizes/minimizes when such s is reached.

We will present algorithms to recursively (in s) compute $\hat{\delta}_s^{\mathcal{U}}$ and $\hat{\delta}_s^{\Box}$ in acyclic MDPs. The base cases of the recursion are the states where the stopping condition holds. In the recursive case we can express $\hat{\delta}_s^{\mathcal{U}}$ (respectively $\hat{\delta}_s^{\Box}$) in terms of the $\hat{\delta}_t^{\mathcal{U}}$ (respectively $\hat{\delta}_t^{\Box}$) of the successors states t of s.

We start by formalizing the notion of acyclic MDP. We call a MDP *acyclic* if it contains no cycles other than the trivial ones (i.e., other than selfloops associated to absorbing states).

Definition 2.6.6. A MDP Π is called *acyclic* if for all states $s \in S$ and all $\pi \in \tau(s)$ we have $\pi(s) = 0$ or $\pi(s) = 1$, and, furthermore, for all paths ω, if there exist i, j such that $i < j$ and $\omega_i = \omega_j$, then we have $\omega_i = \omega_k$ for all $k > i$.

In addition, in order to formally define $\hat{\delta}$ we define a new congruence $\equiv_2$.

Definition 2.6.7. Consider the set of expressions L defined in Definition 2.6.2. On L we now consider the smallest congruence relation $\equiv_2$ containing $\equiv_1$ and satisfying

(1) $(p_1, q_1) \vee (p_1, q_2) \equiv_2 (p_1, \mathsf{min}(q_1, q_2))$, and

(2) $(p_1, q_1) \vee (p_2, q_1) \equiv_2 (\mathsf{max}(p_1, p_2), q_1)$, and

(3) $(p_1 + a, q_1 + a) \vee (p_1, q_1) \equiv_2 (p_1 + a, q_1 + a)$,

where $a \in [0, \infty)$. We write L_2 for the set of equivalence classes and denote the projection map $L \to L_2$ by f_2.

Since $\equiv_1 \subseteq \equiv_2$, this projection maps factors through f_1, say $g\colon L_1 \to L_2$ is the unique map such that $g \circ f_1 = f_2$.

Definition 2.6.8. We define $\hat{\delta} : S \times \mathsf{Stat} \times \mathsf{Path} \times \mathsf{Path} \to L_2$ by $\hat{\delta} \triangleq f_2 \circ \delta$.

Now, in order to prove that $\top(\delta) = \top(\hat{\delta})$ we need to define a *scalar multiplication operator* $\odot$ and an *addition operator* $\oplus$ on L.

Definition 2.6.9. We define $\odot : [0, \infty) \times L \to L$ and $\oplus : L \times L \to L$ by

$$c \odot \bigvee_{i=1}^{n} (p_i, q_i) \triangleq \bigvee_{i=1}^{n} (c \cdot p_i, c \cdot q_i) \text{ and}$$
$$\bigvee_{i=1}^{n} (p_i, q_i) \oplus \bigvee_{j=1}^{m} (p'_j, q'_j) \triangleq \bigvee_{i=1}^{n} \bigvee_{j=1}^{m} (p_i + p'_j, q_i + q'_j).$$

Note that $\odot$ and $\oplus$ induce maps $\odot_1\colon [0, \infty) \times L_1 \to L_1$ and $\oplus_1 : L_1 \times L_1 \to L_1$ as shown in Figure 2.7 below. As before, we omit the subscript 1 if that will not cause confusion.

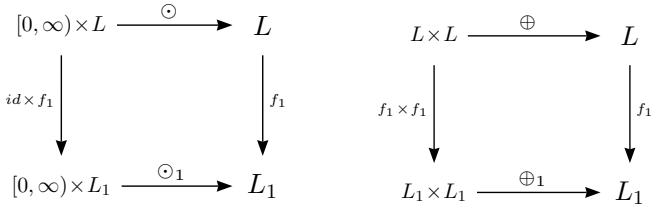


Figure 2.7: Commutative diagrams

The following seemingly innocent lemma is readily proven, but it contains the key to allow us to discard certain pairs of probabilities. The fact

that $\top$ induces operations on L_2 means that it is correct to "simplify" expressions using $\equiv_2$ when we are interested in the maximum or minimum quotient.

The intuition is as follows. Normally, which decision is best in a certain state (or rather, at a certain finite path) to optimize the conditional probability, might depend on probabilities or choices in a totally different part of the automaton (see Example 2.6.1). Sometimes, however, it is possible to decide locally what decision the scheduler should take. The congruence $\equiv_2$ encodes three such cases, each of them corresponding to one clause in Definition 2.6.7. (1) If from a state t the scheduler η can either take a transition after which $\mathbb{P}_\eta[\phi \wedge \psi] = p_1$ and $\mathbb{P}_\eta[\psi] = q_1$ or a transition after which $\mathbb{P}_\eta[\phi \wedge \psi] = p_1$ and $\mathbb{P}_\eta[\psi] = q_2$, then in order to maximize the conditional probability is always best to take the decision where $\mathbb{P}_\eta[\psi] = \mathsf{min}(q_1, q_2)$. (2) Similarly, if the scheduler can either take a transition after which $\mathbb{P}_\eta[\phi \wedge \psi] = p_1$ and $\mathbb{P}_\eta[\psi] = q_1$ or one after which $\mathbb{P}_\eta[\phi \wedge \psi] = p_2$ and $\mathbb{P}_\eta[\psi] = q_1$, then it is always best to take the decision where $\mathbb{P}_\eta[\phi \wedge \psi] = \mathsf{max}(p_1, p_2)$. (3) Finally, if η has the option to either take a transition after which $\mathbb{P}_\eta[\phi \wedge \psi] = p_1 + a$ and $\mathbb{P}_\eta[\psi] = q_1 + a$ or one after which $\mathbb{P}_\eta[\phi \wedge \psi] = p_1$ and $\mathbb{P}_\eta[\psi] = q_1$, for some $a > 0$, then a maximizing scheduler should always take the first of these two options.

Lemma 2.6.10. The operators $\odot$, $\oplus$, and $\top$ on L induce operators $\odot_2$, $\oplus_2$, and $\top_2$ on L_2.

Proof. The idempotence, commutativity and associativity cases are trivial; we only treat the other three cases.

($\odot$) For (1) we have

$$\begin{aligned} c \odot ((p,q) \vee (p,q')) &\triangleq (c \cdot p, c \cdot q) \vee (c \cdot p, c \cdot q') \\ &\equiv (c \cdot p, \mathsf{min}(c \cdot q, c \cdot q') \\ &= (c \cdot p, c \cdot \mathsf{min}(q, q')) \\ &\triangleq c \odot (p, \mathsf{min}(q, q')) \end{aligned}$$

Additionally, note that since $q \geq p$ and $q' \geq p$ we have $\mathsf{min}(q, q') \geq p$.

For (2) the proof goes like in (1). For (3) we have the following

$$\begin{array}{rcl} c \odot ((p+a, q+a) \vee (p,q)) & \triangleq & (c\cdot p + c\cdot a, c\cdot q + c\cdot a) \vee (c\cdot p, c\cdot q) \\ & \equiv & (c\cdot p + c\cdot a, c\cdot q + c\cdot a) \\ & \triangleq & c \odot (p+a, q+a) \end{array}$$

($\oplus$) For (1) we have

$$\begin{array}{rcl} & & ((p,q) \vee (p,q')) \oplus \bigvee_{i=1}^{n} (p_i, q_i) \\ & \triangleq & \bigvee_{i=1}^{n} (p + p_i, q + q_i) \vee \bigvee_{i=1}^{n} (p + p_i, q' + q_i) \\ & \equiv & \bigvee_{i=1}^{n} ((p + p_i, q + q_i) \vee (p + p_i, q' + q_i)) \\ & \equiv & \bigvee_{i=1}^{n} (p + p_i, \mathsf{min}(q + q_i, q' + q_i)) \\ & = & \bigvee_{i=1}^{n} (p + p_i, \mathsf{min}(q, q') + q_i) \\ & \triangleq & (p, \mathsf{min}(q, q')) \oplus \bigvee_{i=1}^{n} (p_i, q_i) \end{array}$$

For (2) the proof goes like in (1). For (3) we have the following

$$\begin{array}{rcl} & & ((p+a, q+a) \vee (p,q)) \oplus \bigvee_{i=1}^{n} (p_i, q_i) \\ & \triangleq & \bigvee_{i=1}^{n} (p + a + p_i, q + a + q_i) \vee \bigvee_{i=1}^{n} (p + p_i, q + q_i) \\ & = & \bigvee_{i=1}^{n} (p + a + p_i, q + a + q_i) \vee (p + p_i, q' + q_i) \\ & \equiv & \bigvee_{i=1}^{n} (p + a + p_i, q + a + q_i) \\ & \triangleq & (p + a, q + a) \oplus \bigvee_{i=1}^{n} (p_i, +q_i) \end{array}$$

($\top$) For (1) we will start by assuming that $q, q' \neq 0$. Then

$$\begin{array}{rcl} & & \top ((p,q) \vee (p,q') \vee \bigvee_{i=1}^{n} (p_i, q_i)) \\ & \triangleq & \mathsf{max}\left(\{\frac{p}{q}\} \cup \{\frac{p}{q'}\} \cup \{\frac{p_i}{q_i} | \forall_{1 \leq i \leq n}. q_i \neq 0\} \cup \{0\} \right) \\ & = & \mathsf{max}\left(\{\frac{p}{\mathsf{min}(q,q')}\} \cup \{\frac{p_i}{q_i} | \forall_{1 \leq i \leq n}. q_i \neq 0\} \cup \{0\} \right) \\ & \triangleq & \top ((p, \mathsf{min}(q,q')) \vee \bigvee_{i=1}^{n} (p_i, q_i)) \end{array}$$

Now we assume that $q = 0, q' \neq 0$ (the case $q \neq 0, q' = 0$ is similar).

Note that we now have that $p = 0$. Then

$$\begin{array}{rl} & \top((p,q) \vee (p,q') \vee \bigvee_{i=1}^{n}(p_i,q_i)) \\ \triangleq & \mathsf{max}\left(\{\frac{0}{q'}\} \cup \{\frac{p_i}{q_i} | \forall_{1\leq i\leq n}.q_i \neq 0\} \cup \{0\}\right) \\ = & \mathsf{max}\left(\{\frac{p_i}{q_i} | \forall_{1\leq i\leq n}.q_i \neq 0\} \cup \{0\}\right) \\ \triangleq & \top((p,0) \vee \bigvee_{i=1}^{n}(p_i,q_i)) \\ = & \top((p,\mathsf{min}(q,q')) \vee \bigvee_{i=1}^{n}(p_i,q_i)) \end{array}$$

Finally, assume that $q = q' = 0$, then also $p = 0$, so

$$\begin{array}{rcl} \top((p,q) \vee (p,q') \vee \bigvee_{i=1}^{n}(p_i,q_i)) & \triangleq & \mathsf{max}\left(\{\frac{p_i}{q_i} | \forall_{1\leq i\leq n}.q_i \neq 0\} \cup \{0\}\right) \\ & \triangleq & \top((p,0) \vee \bigvee_{i=1}^{n}(p_i,q_i)) \\ & = & \top((p,\mathsf{min}(q,q')) \vee \bigvee_{i=1}^{n}(p_i,q_i)) \end{array}$$

For (2) the proof goes like in (1). For (3) we first need the following.

Let $f : \mathbb{R} \to \mathbb{R}$ be a function defined as $f(x) \triangleq \frac{a+x}{b+x}$ where a and b are constants in the interval $(0, 1]$. Then f is increasing. Let us now assume that $q \neq 0$ or $a \neq 0$. Then

$$\begin{array}{rl} & \top((p+a,q+a) \vee (p,q)) \vee \bigvee_{i=1}^{n}(p_i,q_i)) \\ \triangleq & \mathsf{max}\left(\{\frac{p+a}{q+a}\} \cup \{\frac{p}{q}\} \cup \{\frac{p_i}{q_i} | \forall_{1\leq i\leq n}.q_i \neq 0\} \cup \{0\}\right) \\ = & \mathsf{max}\left(\{\frac{p+a}{q+a}\} \cup \{\frac{p_i}{q_i} | \forall_{1\leq i\leq n}.q_i \neq 0\} \cup \{0\}\right) \\ & \{\text{By obervation above about } f\} \\ \triangleq & \top((p+a,q+a) \bigvee_{i=1}^{n}(p_i,q_i)) \end{array}$$

Now assume that $q = a = 0$. Then

$$\begin{array}{rl} & \top((p+a,q+a) \vee (p,q)) \vee \bigvee_{i=1}^{n}(p_i,q_i)) \\ \triangleq & \mathsf{max}\left(\{\frac{p_i}{q_i} | \forall_{1\leq i\leq n}.q_i \neq 0\} \cup \{0\}\right) \\ = & \top((p+a,0) \vee \bigvee_{i=1}^{n}(p_i,q_i)) \\ \triangleq & \top((p+a,q+a) \vee \bigvee_{i=1}^{n}(p_i,q_i)) \end{array}$$

□

The fact that $\top(\delta) = \top(\hat{\delta})$ follows from previous lemma.

Finally, the following theorem provides recursive equations for the values of $\hat{\delta}_s^{\mathcal{U}}$ and $\hat{\delta}_s^{\Box}$. If the MDPs is acyclic, it can be used to compute these values.

Theorem 2.6.11. Let Π be a MDP, $s \in S$, and $\phi_1 \mathcal{U} \phi_2, \psi_1 \mathcal{U} \psi_2, \Box\psi_1 \in \mathsf{Path}$. Then $\hat{\delta}_s^{\mathcal{U}}(\phi_1 \mathcal{U} \phi_2 \mid \psi_1 \mathcal{U} \psi_2) =$

$$\begin{cases} (\mathbb{P}_s^+[\psi_1 \mathcal{U} \psi_2], \mathbb{P}_s^+[\psi_1 \mathcal{U} \psi_2]) & \text{if } s \models \phi_2, \\ (\mathbb{P}_s^+[\phi_1 \mathcal{U} \phi_2], 1) & \text{if } s \models \neg\phi_2 \wedge \psi_2, \\ (0, \mathbb{P}_s^-[\psi_1 \mathcal{U} \psi_2]) & \text{if } s \models \neg\phi_1 \wedge \neg\phi_2 \wedge \neg\psi_2, \\ (0,0) & \text{if } s \models \phi_1 \wedge \neg\phi_2 \wedge \neg\psi_1 \wedge \neg\psi_2, \\ \bigvee\limits_{\pi \in \tau(s)} \left(\sum\limits_{t \in \mathsf{succ}(s)} \pi(t) \odot \hat{\delta}_t^{\mathcal{U}}(\phi_1 \mathcal{U} \phi_2 \mid \psi_1 \mathcal{U} \psi_2) \right) & \text{if } s \models \phi_1 \wedge \neg\phi_2 \wedge \psi_1 \wedge \neg\psi_2, \end{cases}$$

and $\hat{\delta}_s^{\Box}(\phi_1 \mathcal{U} \phi_2 \mid \Box\psi_1) =$

$$\begin{cases} (\mathbb{P}_s^+[\Box\psi_1], \mathbb{P}_s^+[\Box\psi_1]) & \text{if } s \models \phi_2, \\ (0,0) & \text{if } s \models \neg\phi_2 \wedge \neg\psi_1, \\ (0, \mathbb{P}_s^-[\Box\psi_1]) & \text{if } s \models \neg\phi_1 \wedge \neg\phi_2 \wedge \psi_1, \\ \bigvee\limits_{\pi \in \tau(s)} \left(\sum\limits_{t \in \mathsf{succ}(s)} \pi(t) \odot \hat{\delta}_t^{\Box}(\phi_1 \mathcal{U} \phi_2 \mid \Box\psi_1) \right) & \text{if } s \models \phi_1 \wedge \neg\phi_2 \wedge \psi_1. \end{cases}$$

Proof. We will consider the case $\hat{\delta}_s^{\mathcal{U}}$. We will use φ to denote $\phi_1 \wedge \neg\phi_2 \wedge \psi_1 \wedge \neg\psi_2$, i.e., the stopping condition of cpCTL formula under consideration.

(a) Note that if $s \models \phi_2$, then deterministic semi HI schedulers are exactly the HI schedulers, i.e., $\mathsf{Sch}^{\varphi}(\Pi) = \mathsf{Sch}_d^{\mathsf{HI}}(\Pi)$.

$$
\begin{aligned}
&\hat{\delta}_s^{\mathcal{U}}(\phi_1 \mathcal{U} \phi_2 \mid \psi_1 \mathcal{U} \psi_2) \\
=& \bigvee_{\eta \in \mathsf{Sch}_s^{\varphi}(\Pi)} \big(\mathbb{P}_{s,\eta}[\phi_1 \mathcal{U} \phi_2 \wedge \psi_1 \mathcal{U} \psi_2], \mathbb{P}_{s,\eta}[\psi_1 \mathcal{U} \psi_2]\big) \\
&\{s \models \phi_2\} \\
=& \bigvee_{\eta \in \mathsf{Sch}_s^{\varphi}(\Pi)} \big(\mathbb{P}_{s,\eta}[\psi_1 \mathcal{U} \psi_2], \mathbb{P}_{s,\eta}[\psi_1 \mathcal{U} \psi_2]\big) \\
=&(\mathbb{P}_s^+[\psi_1 \mathcal{U} \psi_2], \mathbb{P}_s^+[\psi_1 \mathcal{U} \psi_2]) \quad \{\text{Case (3)}\}
\end{aligned}
$$

(b)

$$
\begin{aligned}
&\hat{\delta}_s^{\mathcal{U}}(\phi_1 \mathcal{U} \phi_2 \mid \psi_1 \mathcal{U} \psi_2) \\
=& \bigvee_{\eta \in \mathsf{Sch}_s^{\varphi}(\Pi)} \big(\mathbb{P}_{s,\eta}[\phi_1 \mathcal{U} \phi_2 \wedge \psi_1 \mathcal{U} \psi_2], \mathbb{P}_{s,\eta}[\psi_1 \mathcal{U} \psi_2]\big) \\
&\{s \models \psi_2\} \\
=& \bigvee_{\eta \in \mathsf{Sch}_s^{\varphi}(\Pi)} \big(\mathbb{P}_{s,\eta}[\phi_1 \mathcal{U} \phi_2], \mathbb{P}_{s,\eta}[\mathbf{true}]\big) \\
&\{\text{Case (2) and definition of } \mathbb{P}[\mathbf{true}]\} \\
=&(\mathbb{P}_s^+[\phi_1 \mathcal{U} \phi_2], 1)
\end{aligned}
$$

(c)

$$
\begin{aligned}
&\hat{\delta}_s^{\mathcal{U}}(\phi_1 \mathcal{U} \phi_2 \mid \psi_1 \mathcal{U} \psi_2) \\
=& \bigvee_{\eta \in \mathsf{Sch}_s^{\varphi}(\Pi)} \big(\mathbb{P}_{s,\eta}[\phi_1 \mathcal{U} \phi_2 \wedge \psi_1 \mathcal{U} \psi_2], \mathbb{P}_{s,\eta}[\psi_1 \mathcal{U} \psi_2]\big) \\
&\{s \models \neg\phi_1 \wedge \neg\phi_2 \wedge \neg\psi_2\} \\
=& \bigvee_{\eta \in \mathsf{Sch}_s^{\varphi}(\Pi)} \big(\mathbb{P}_{s,\eta}[\mathbf{false}], \mathbb{P}_{s,\eta}[\psi_1 \mathcal{U} \psi_2]\big) \\
&\{\text{Case (1) and definition of } \mathbb{P}[\mathbf{false}]\} \\
=& (0, \mathbb{P}_s^-[\psi_1 \mathcal{U} \psi_2])
\end{aligned}
$$

(d)
$$\begin{aligned}
&\hat{\delta}_s^{\mathcal{U}}(\phi_1 \mathcal{U} \phi_2 \mid \psi_1 \mathcal{U} \psi_2) \\
&= \bigvee_{\eta \in \mathsf{Sch}_s^{\varphi}(\Pi)} \left(\mathbb{P}_{s,\eta}[\phi_1 \mathcal{U} \phi_2 \wedge \psi_1 \mathcal{U} \psi_2], \mathbb{P}_{s,\eta}[\psi_1 \mathcal{U} \psi_2]\right) \\
&\quad \{\text{Since } s \models \neg\phi_1 \wedge \neg\phi_2 \wedge \psi_1 \wedge \neg\psi_2\} \\
&= \bigvee_{\eta \in \mathsf{Sch}_s^{\varphi}(\Pi)} \left(\mathbb{P}_{s,\eta}[\mathbf{false}], \mathbb{P}_{s,\eta}[\mathbf{false}]\right) = (0,0)
\end{aligned}$$

(e)
$$\begin{aligned}
&\hat{\delta}_s^{\mathcal{U}}(\phi_1 \mathcal{U} \phi_2 \mid \psi_1 \mathcal{U} \psi_2) \\
&= \bigvee_{\eta \in \mathsf{Sch}_s^{\varphi}(\Pi)} \left(\mathbb{P}_{s,\eta}[\phi_1 \mathcal{U} \phi_2 \wedge \psi_1 \mathcal{U} \psi_2], \mathbb{P}_{s,\eta}[\psi_1 \mathcal{U} \psi_2]\right) \\
&= \bigvee_{\eta \in \mathsf{Sch}_s^{\varphi}(\Pi)} \left(\sum_{t \in \mathsf{succ}(s)} \eta(s)(t) \odot \left(\mathbb{P}_{t,\eta}[\phi_1 \mathcal{U} \phi_2 \wedge \psi_1 \mathcal{U} \psi_2], \mathbb{P}_{t,\eta}[\psi_1 \mathcal{U} \psi_2]\right) \right) \\
&= \{ \text{ Rewrite of the sum} \} \\
&\bigvee_{\pi \in \tau(s)} \left(\sum_{t \in \mathsf{succ}(s)} \pi(t) \odot \bigvee_{\eta_t \in \mathsf{Sch}_t^{\varphi}(\Pi)} \left(\mathbb{P}_{t,\eta_t}[\phi_1 \mathcal{U} \phi_2 \wedge \psi_1 \mathcal{U} \psi_2], \mathbb{P}_{t,\eta_t}[\psi_1 \mathcal{U} \psi_2]\right) \right) \\
&= \bigvee_{\pi \in \tau(s)} \left(\sum_{t \in \mathsf{succ}(s)} \pi(t) \odot \hat{\delta}_t^{\mathcal{U}} \right)
\end{aligned}$$

□

From MDPs to Acyclic MDPs

We now show how to reduce a MDP with cycles to an acyclic one, thus generalizing our results to MDPs with cycles. For that purpose we first reduce all cycles in Π and create a new acyclic MDP $[\Pi]$ such that the probabilities involved in the computation of $\mathbb{P}^+[-|-]$ are preserved. We do so by removing every strongly connected component (SCC) k of (the

graph of) Π, keeping only input states and transitions to output states (in the spirit of [ADvR08]). We show that $\mathbb{P}^+[-|-]$ on $[\Pi]$ is equal to the corresponding value on Π. For this, we have to make sure that states satisfying the stopping condition are ignored when removing SCCs.

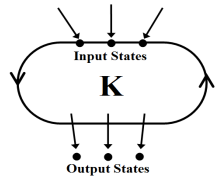
(1) Identifying SCCs. Our first step is to make states satisfying the stopping condition absorbing.

Definition 2.6.12. Let $\Pi = (S, s_0, \tau, L)$ be a MDP and $\varphi \in \mathsf{Stat}$ a state formula. We define a new MDP $\langle\Pi\rangle_\varphi = (S, s_0, \langle\tau\rangle_\varphi, L)$ where $\langle\tau\rangle_\varphi(s)$ is equal to $\tau(s)$ if $s \not\models \varphi$ and to 1_s otherwise.

To recognize cycles in the MDP we define a graph associated to it.

Definition 2.6.13. Let $\Pi = (S, s_0, \tau, L)$ be MDP and $\varphi \in \mathsf{Stat}$. We define the digraph $G = G_{\Pi,\varphi} = (S, \to)$ associated to $\langle\Pi\rangle_\varphi = (S, s_0, \langle\tau\rangle_\varphi, L)$ where $\to$ satisfies $u \to v \Leftrightarrow \exists \pi \in \langle\tau\rangle_\varphi(u).\pi(v) > 0$.

Now we let $\mathsf{SCC} = \mathsf{SCC}_{\Pi,\varphi} \subseteq \wp(S)$ be the set of SCC of G. For each SCC k we define the sets Inp_k of all states in k that have an incoming transition of Π from a state outside of k; we also define the set Out_k of all states outside of k that have an incoming transition from a state of k. Formally, for each $k \in \mathsf{SCC}$ we define



$$Inp_k \triangleq \{u \in k \mid \exists\, s \in S \setminus \{k\} \text{ such that } (s,u) \in \varrho\},$$
$$Out_k \triangleq \{s \in S \setminus \{k\} \mid \exists\, u \in k \text{ such that } (u,s) \in \varrho\}.$$

where ϱ is the successor relation defined in Section 2.2.
We then associate a MDP Π_k to each SCC k of G. The space of states of Π_k is $k \cup Out_k$ and the transition relation is induced by the transition relation of Π.

Definition 2.6.14. Let Π be a MDP and $k \in \mathsf{SCC}$ be a SCC in Π. We pick an arbitrary element s_k of Inp_k and define the MDP $\Pi_k = (S_k, s_k, \tau_k, L)$ where $S_k = \{k\} \cup Out_k$ and $\tau_k(s)$ is equal to $\{1_s\}$ if $s \in Out_k$ and to $\tau(s)$ otherwise.

(2) Constructing an acyclic MDP. To obtain a reduced acyclic MDP from the original one we first define the probability of reaching one state from another according to a given HI scheduler in the following way.

Definition 2.6.15. Let $\Pi = (S, s_0, \tau, L)$ be a MDP, and η be a HI scheduler on Π. Then for each $s, t \in S$ we define the function R such that $R_\Pi(s \overset{\eta}{\rightsquigarrow} t) \triangleq \mathbb{P}_{s,\eta}(\{\omega \in \mathsf{Paths}(s) \mid \exists i.\omega_i = t\})$.

We note that such reachability values can be efficiently computed using steady-state analysis techniques [Cas93].
Now we are able to define an acyclic MDP $[\Pi]$ related to Π such that $\mathbb{P}^+_{[\Pi]}[-|-] = \mathbb{P}^+_\Pi[-|-]$.

Definition 2.6.16. Let $\Pi = (S, s_0, \tau, L)$ be a MDP. Then we define $[\Pi]$ as $([S], s_0, [\tau], L)$ where

$$[S] = \overbrace{S \setminus \bigcup_{k \in \mathsf{SCC}} k}^{S_{com}} \cup \overbrace{\bigcup_{k \in \mathsf{SCC}} Inp_k}^{S_{inp}}$$

and for all $s \in [S]$ the set $[\tau](s)$ of probabilistic distributions on $[S]$ is given by

$$[\tau](s) = \begin{cases} \tau(s) & \text{if } s \in S_{com}, \\ \{\lambda \in [S].R_{\Pi_{k_s}}(s \overset{\eta}{\rightsquigarrow} t)) \mid \eta \in \mathsf{Sch}^{\mathsf{HI}}_d(\Pi_{k_s})\} & \text{if } s \in S_{inp}. \end{cases}$$

Here k_s is the SCC associated to s and $\mathsf{Sch}^{\mathsf{HI}}_d(\Pi_{k_s})$ denotes the set of deterministic and history independent s-schedulers of Π_{k_s}.

Theorem 2.6.17. Let $\Pi = (S, s_0, \tau, L)$ be a MDP, and $\mathbb{P}_{\leq a}[\phi|\psi] \in$ cpCTL. Then $[\Pi]$ is an acyclic MDP and $\mathbb{P}^+_{s_0,\Pi}[\phi|\psi] = \mathbb{P}^+_{s_0,[\Pi]}[\phi|\psi]$, where $\mathbb{P}^+_{s,\Pi'}[-|-]$ represents $\mathbb{P}^+_s[-|-]$ on the MDP Π'.

Proof. The proof follows straightforwardly by the construction of $[\Pi]$ and Theorem 2.5.11. □

Finally we can use the technique for acyclic MDPs on the reduced MDP in order to obtain $\mathbb{P}^+_{s_0}[-|-]$.

2.6.2 Complexity

As mentioned before, when computing maximum or minimum conditional probabilities it is not possible to locally optimize. Therefore, it is necessary to carry on, for each deterministic and HI scheduler η, the pair of probabilities $(\mathbb{P}_\eta[\phi \wedge \psi], \mathbb{P}_\eta[\psi])$ from the leafs (states satisfying the stopping condition) to the initial state. As the number of HI schedulers in a MDP grows exponentially on the state space, our algorithm to verify cpCTL formulas has exponential time complexity.

We believe that the complexity of computing optimal conditional probabilities is intrinsically exponential, i.e. computing such probabilities is an NP-complete problem. However, a deeper study on this direction is still missing,

Conditional probability bounds Even if computing exact conditional probabilities is computationally expensive (exponential time), it is still possible to efficiently compute upper and lower bounds for such probabilities (polynomial time).

Observation 2.6.1. Let Π be a MDP and ϕ, ψ two path pCTL formulas. Then we have

$$\frac{\mathbb{P}^-[\phi \wedge \psi]}{1 - \mathbb{P}^-[\psi]} \leq \mathbb{P}^+[\phi|\psi] \leq \frac{\mathbb{P}^+[\phi \wedge \psi]}{1 - \mathbb{P}^+[\psi]}.$$

2.7 Counterexamples for cpCTL

Counterexamples in model checking provide important diagnostic information used, among others, for debugging, abstraction-refinement [CGJ+00], and scheduler synthesis [LBB+01]. For systems without probability, a counterexample typically consists of a path violating the property under consideration. Counterexamples in MCs are sets of paths. For instance, a counterexample for the formula $\mathbb{P}_{\leq a}[\phi]$ is a set Δ of paths, satisfying ϕ, and such that the probability mass of Δ is greater than a [HK07a, ADvR08, AL06].

In MDPs, we first have to find the scheduler achieving the optimal probability. Both for pCTL and cpCTL, this scheduler can be derived from

the algorithms computing the optimal probabilities [ADvR08]. Once the optimal scheduler is fixed, the MDP can be turned into a Markov Chain and the approaches mentioned before can be used to construct counterexamples for pCTL. For cpCTL however, the situation is slightly more complex. It follows directly from the semantics that:

$$s \not\models \mathbb{P}_{\leq a}[\phi|\psi] \quad \text{iff} \quad \exists \eta \in \mathsf{Sch}_s(\Pi). \frac{\mathbb{P}_{s,\eta}(\{\omega \in \mathsf{Paths}(s) | \omega \models \phi \wedge \psi\})}{\mathbb{P}_{s,\eta}(\{\omega \in \mathsf{Paths}(s) | \omega \models \psi\})} > a.$$

Lemma 2.7.1. Let $a \in [0,1]$ and consider the formula $\mathbb{P}_{\leq a}[\phi|\psi]$. Let $\Delta_\phi \triangleq \{\omega \in \mathsf{Paths} \mid \omega \models \phi\}$, $\Delta_1 \subseteq \Delta_{\phi \wedge \psi}$, and $\Delta_2 \subseteq \Delta_{\neg\psi}$. Then $a < \mathbb{P}_\eta(\Delta_1)/(1 - \mathbb{P}_\eta(\Delta_2))$ implies $a < \mathbb{P}_\eta[\phi|\psi]$.

Proof. We first note that

$$\mathbb{P}_\eta(\Delta_1) \leq \mathbb{P}_\eta(\Delta_{\phi\wedge\psi}) \quad \text{and} \quad \mathbb{P}_\eta(\Delta_2) \leq \mathbb{P}_\eta(\Delta_{\neg\psi}).$$

Then, it is easy to see that

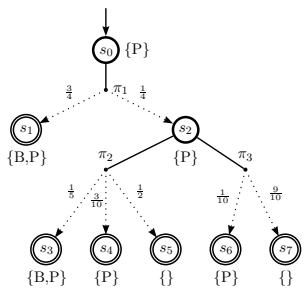
$$a < \frac{\mathbb{P}_\eta(\Delta_1)}{1 - \mathbb{P}_\eta(\Delta_2)} \leq \frac{\mathbb{P}_\eta(\Delta_{\phi\wedge\psi})}{1 - \mathbb{P}_\eta(\Delta_{\neg\psi})} = \frac{\mathbb{P}_\eta(\Delta_{\phi\wedge\psi})}{\mathbb{P}_\eta(\Delta_\psi)} = \mathbb{P}_\eta[\phi|\psi]. \quad \square$$

This leads to the following notion of counterexample.

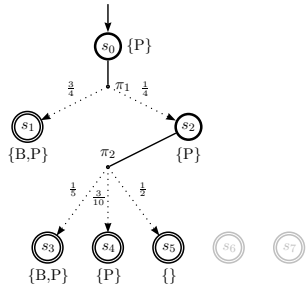
Definition 2.7.2. A *counterexample* for $\mathbb{P}_{\leq a}[\phi|\psi]$ is a pair (Δ_1, Δ_2) of measurable sets of paths satisfying $\Delta_1 \subseteq \Delta_{\phi\wedge\psi}$, $\Delta_2 \subseteq \Delta_{\neg\psi}$, and $a < \mathbb{P}_\eta(\Delta_1)/(1 - \mathbb{P}_\eta(\Delta_2))$, for some scheduler η.

Note that such sets Δ_1 and Δ_2 can be computed using standard techniques on Markov Chains (see Chapter 5 for more detailed information about these techniques).

Example 2.7.3. Consider the evaluation of $s_0 \models \mathbb{P}_{\leq 0.75}[\Diamond B|\Box P]$ on the MDP obtained by taking $\alpha = \frac{1}{10}$ in the MDP depictured in Figure 2.1. The corresponding MDP is shown in Figure 2.7(a). In this case the maximizing scheduler, say η, chooses π_2 in s_2. In Figure 2.7(b) we show the Markov Chain derived from MDP using η. In this setting we have $\mathbb{P}_{s_0,\eta}[\Diamond B|\Box P] = \frac{68}{70}$ and consequently s_0 does not satisfy this formula.



2.8(a) MDP



2.8(b) Markov Chain

We show this fact with the notion of counterexample of Definition 2.7.2. Note that $\Delta_{\Diamond B \wedge \Box P} = \langle s_0 s_1 \rangle \cup \langle s_0 s_2 s_3 \rangle$ and $\Delta_{\neg \Box P} = \langle s_0 s_2 s_5 \rangle$. Using Lemma 2.7.1 with $\Delta_1 = \langle s_0 s_1 \rangle$ and $\Delta_2 = \langle s_0 s_2 s_5 \rangle$ we have $\frac{3}{4} < \frac{\mathbb{P}_\eta(\Delta_1)}{1-\mathbb{P}_\eta(\Delta_2)} = \frac{3/4}{1-1/8} = \frac{6}{7}$. Consequently $\frac{3}{4} < \mathbb{P}_{s_0,\eta}[\Diamond B | \Box P]$, which proves that $s_0 \not\models \mathbb{P}_{\leq 3/4}[\Diamond B | \Box P]$.

Chapter 3

Computing the Leakage of Information Hiding Systems

In this chapter we address the problem of computing the information leakage of a system in an efficient way. We propose two methods: one based on reducing the problem to reachability, and the other based on techniques from quantitative counterexample generation. The second approach can be used either for exact or approximate computation, and provides feedback for debugging. These methods can be applied also in the case in which the input distribution is unknown. We then consider the interactive case and we point out that the definition of associated channel proposed in literature is not sound. We show however that the leakage can still be defined consistently, and that our methods extend smoothly.

3.1 Introduction

By *information hiding*, we refer generally to the problem of constructing protocols or programs that protect sensitive information from being deduced by some adversary. In *anonymity protocols* [CPP08a], for example,

the concern is to design mechanisms to prevent an observer of network traffic from deducing who is communicating. In *secure information flow* [SM03], the concern is to prevent programs from leaking their secret input to an observer of their public output. Such leakage could be accidental or malicious.

Recently, there has been particular interest in approaching these issues *quantitatively*, using concepts of information theory. See for example [MNCM03, CHM05b, DPW06, CMS09, CPP08a]. The secret input S and the observable output O of an information-hiding system are modeled as random variables related by a *channel matrix*, whose (s, o) entry specifies $P(o|s)$, the conditional probability of observing output o given input s. If we define the *vulnerability* of S as the probability that the adversary could correctly guess the value of S in one try, then it is natural to measure the information leakage by comparing the *a priori* vulnerability of S with the *a posteriori* vulnerability of S after observing O. We consider two measures of leakage: *additive*, which is the difference between the *a posteriori* and *a priori* vulnerabilities; and *multiplicative*, which is their quotient [Smi09, BCP09].

We thus view a protocol or program as a *noisy channel*, and we calculate the leakage from the channel matrix and the *a priori* distribution on S. But, given an operational specification of a protocol or program, how do we calculate the parameters of the noisy channel: the sets of inputs and outputs, the *a priori* distribution, the channel matrix, and the associated leakage? These are the main questions we address in this chapter. We focus on *probabilistic automata*, whose transitions are labeled with probabilities and *actions*, each of which is classified as secret, observable, or internal.

We first consider the simple case in which the secret inputs take place at the beginning of runs, and their probability is fixed. The interpretation in terms of noisy channel of this kind of systems is well understood in literature. The framework of probabilistic automata, however, allows to represent more general situations. Thanks to the nondeterministic choice, indeed, we can model the case in which the input distribution is unknown, or variable. We show that the definition of channel matrix extends smoothly also to this case. Finally, we turn our attention to the interactive scenario

in which inputs can occur again after outputs. This case has also been considered in literature, and there has been an attempt to define the channel matrix in terms of the probabilities of traces [DJGP02]. However it turns out that the notion of channel is unsound. Fortunately the leakage is still well defined, and it can be obtained in the same way as the simple case.

We consider two different approaches to computing the channel matrix. One uses a system of linear equations as in reachability computations. With this system of equations one can compute the *joint matrix*, the matrix of probabilities of observing both s and o; the channel matrix is trivially derived from this joint matrix. The other approach starts with a 0 channel matrix, which we call a *partial matrix* at this point. We iteratively add the contributions in conditional probabilities of complete paths to this partial matrix, obtaining, in the limit, the channel matrix itself. We then group paths with the same secret and the same observable together using ideas from quantitative counterexample generation, namely by using regular expressions and strongly connected component analysis. In this way, we can add the contribution of (infinitely) many paths at the same time to the partial matrices. This second approach also makes it possible to identify which parts of a protocol contribute most to the leakage, which is useful for debugging.

Looking ahead, after reviewing some preliminaries (Section 3.2) we present restrictions on probabilistic automata to ensure that they have well-defined and finite channel matrices (Section 3.3). This is followed by the techniques to calculate the channel matrix efficiently (Section 3.4 and Section 3.5). We then turn our attention to extensions of our information-hiding system model. We use nondeterministic choice to model the situation where the *a priori* distribution on the secret is unknown (Section 3.6). Finally, we consider interactive systems, in which secret actions and observable actions can be interleaved arbitrarily (Section 3.7).

3.2 Preliminaries

3.2.1 Probabilistic automata

This section recalls some basic notions on probabilistic automata. More details can be found in [Seg95]. A function $\mu\colon Q \to [0,1]$ is a *discrete probability distribution* on a set Q if the support of μ is countable and $\sum_{q\in Q}\mu(q) = 1$. The set of all discrete probability distributions on Q is denoted by $\mathcal{D}(Q)$.

A *probabilistic automaton* is a quadruple $M = (Q, \Sigma, \hat{q}, \alpha)$ where Q is a countable set of *states*, Σ a finite set of *actions*, $\hat{q}$ the *initial* state, and α a *transition function* $\alpha : Q \to \wp_f(\mathcal{D}(\Sigma \times Q))$. Here $\wp_f(X)$ is the set of all finite subsets of X. If $\alpha(q) = \emptyset$ then q is a *terminal* state. We write $q{\to}\mu$ for $\mu \in \alpha(q),\ q \in Q$. Moreover, we write $q \xrightarrow{a} r$ for $q, r \in Q$ whenever $q{\to}\mu$ and $\mu(a, r) > 0$. A *fully probabilistic automaton* is a probabilistic automaton satisfying $|\alpha(q)| \leq 1$ for all states. In case $\alpha(q) \neq \emptyset$ we will overload notation and use $\alpha(q)$ to denote the distribution outgoing from q.

A *path* in a probabilistic automaton is a sequence $\sigma = q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots$ where $q_i \in Q$, $a_i \in \Sigma$ and $q_i \xrightarrow{a_{i+1}} q_{i+1}$. A path can be *finite* in which case it ends with a state. A path is *complete* if it is either infinite or finite ending in a terminal state. Given a path σ, $\mathsf{first}(\sigma)$ denotes its first state, and if σ is finite then $\mathsf{last}(\sigma)$ denotes its last state. A *cycle* is a path σ such that $\mathsf{last}(\sigma) = \mathsf{first}(\sigma)$. We denote the set of actions occurring in a cycle as $\mathsf{CyclesA}(M)$. Let $\mathsf{Paths}_q(M)$ denote the set of all paths, $\mathsf{Paths}^\star{}_q(M)$ the set of all finite paths, and $\mathsf{CPaths}_q(M)$ the set of all complete paths of an automaton M, starting from the state q. We will omit q if $q = \hat{q}$. Paths are ordered by the prefix relation, which we denote by $\leq$. The *trace* of a path is the sequence of actions in $\Sigma^* \cup \Sigma^\infty$ obtained by removing the states, hence for the above σ we have $\mathit{trace}(\sigma) = a_1 a_2 \ldots$. If $\Sigma' \subseteq \Sigma$, then $\mathit{trace}_{\Sigma'}(\sigma)$ is the projection of $\mathit{trace}(\sigma)$ on the elements of Σ'. The *length* of a finite path σ, denoted by $|\sigma|$, is the number of actions in its trace.

Let $M = (Q, \Sigma, \hat{q}, \alpha)$ be a (fully) probabilistic automaton, $q \in Q$ a state, and let $\sigma \in \mathsf{Paths}^\star_q(M)$ be a finite path starting in q. The *cone* generated by σ is the set of complete paths $\langle\sigma\rangle = \{\sigma' \in \mathsf{CPaths}_q(M) \mid \sigma \leq \sigma'\}$. Given a

fully probabilistic automaton $M = (Q, \Sigma, \hat{q}, \alpha)$ and a state q, we can calculate the *probability value*, denoted by $\mathbb{P}_q(\sigma)$, of any finite path σ starting in q as follows: $\mathbb{P}_q(q) = 1$ and $\mathbb{P}_q(\sigma \xrightarrow{a} q') = \mathbb{P}_q(\sigma)\ \mu(a, q')$, where $\mathsf{last}(\sigma) \to \mu$.

Let $\Omega_q \triangleq \mathsf{CPaths}_q(M)$ be the sample space, and let $\mathcal{F}_q$ be the smallest σ-algebra generated by the cones. Then $\mathbb{P}$ induces a unique *probability measure* on $\mathcal{F}_q$ (which we will also denote by $\mathbb{P}_q$) such that $\mathbb{P}_q(\langle\sigma\rangle) = \mathbb{P}_q(\sigma)$ for every finite path σ starting in q. For $q = \hat{q}$ we write $\mathbb{P}$ instead of $\mathbb{P}_{\hat{q}}$.

Given a probability space $(\Omega, \mathcal{F}, P)$ and two events $A, B \in F$ with $P(B) > 0$, the *conditional probability* of A given B, $P(A \mid B)$, is defined as $P(A \cap B)/P(B)$.

3.2.2 Noisy Channels

This section briefly recalls the notion of noisy channels from Information Theory [CT06].

A *noisy channel* is a tuple $\mathcal{C} \triangleq (\mathcal{X}, \mathcal{Y}, P(\cdot|\cdot))$ where $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ is a finite set of *input values*, modeling the *secrets* of the channel, and $\mathcal{Y} = \{y_1, y_2, \ldots, y_m\}$ is a finite set of *output values*, modeling the *observables* of the channel. For $x_i \in \mathcal{X}$ and $y_j \in \mathcal{Y}$, $P(y_j|\, x_i)$ is the conditional probability of obtaining the output y_j given that the input is x_i. These conditional probabilities constitute the so called *channel matrix*, where $P(y_j|x_i)$ is the element at the intersection of the i-th row and the j-th column. For any input distribution P_X on $\mathcal{X}$, P_X and the channel matrix determine a joint probability $P_\wedge$ on $\mathcal{X} \times \mathcal{Y}$, and the corresponding marginal probability P_Y on $\mathcal{Y}$ (and hence a random variable Y). P_X is also called *a priori distribution* and it is often denoted by π. The probability of the input given the output is called *a posteriori distribution*.

3.2.3 Information leakage

We recall now some notions of information leakage which allow us to quantify the probability of success of a *one-try attacker*, i.e. an attacker that tries to obtain the value of the secret in just one guess. In particular, we

consider Smith's definition of *multiplicative leakage* [Smi09][1], and the *additive leakage* definition from Braun et al. [BCP09]. We assume given a noisy channel $\mathcal{C} = (\mathcal{X}, \mathcal{Y}, P(\cdot|\cdot))$ and a random variable X on $\mathcal{X}$. The *a priori vulnerability* of the secrets in $\mathcal{X}$ is the probability of guessing the right secret, defined as $V(X) \triangleq \mathsf{max}_{x\in\mathcal{X}}\, P_X(x)$. The rationale behind this definition is that the adversary's best bet is on the secret with highest probability.

The *a posteriori vulnerability* of the secrets in $\mathcal{X}$ is the probability of guessing the right secret, after the output has been observed, averaged over the probabilities of the observables. The formal definition is $V(X|\,Y) \triangleq \sum_{y\in\mathcal{Y}} P_Y(y)\, \mathsf{max}_{x\in\mathcal{X}}\, P(x\,|\,y)$. Again, this definition is based on the principle that the adversary will choose the secret with the highest a posteriori probability.

Note that, using Bayes theorem, we can write the a posteriori vulnerability in terms of the channel matrix and the a priori distribution, or in terms of the joint probability:

$$V(X|\,Y) \;=\; \sum_{y\in\mathcal{Y}} \mathsf{max}_{x\in\mathcal{X}}(P(y\,|\,x)P_X(x)) \;=\; \sum_{y\in\mathcal{Y}} \mathsf{max}_{x\in\mathcal{X}}\, P_\wedge(x,y). \tag{3.1}$$

The *multiplicative* leakage is then defined as the *quotient* between the a posteriori and a priori vulnerabilities, $\mathcal{L}_\times(\mathcal{C}, P_X) \triangleq V(X|\,Y) \,/\, V(X)$. Similarly, the *additive* leakage is defined as the *difference* between both vulnerabilities, $\mathcal{L}_+(\mathcal{C}, P_X) \triangleq V(X|\,Y) - V(X)$.

3.3 Information Hiding Systems

To formally analyze the information-hiding properties of protocols and programs, we propose to model them as a particular kind of probabilistic automata, which we call *Information-Hiding Systems* (IHS). Intuitively, an IHS is a probabilistic automaton in which the actions are divided in three

[1]The notion proposed by Smith in [Smi09] was given in a (equivalent) logarithmic form, and called simply *leakage*. For uniformity's sake we use here the terminology and formulation of [BCP09].

(disjoint) categories: those which are supposed to remain secret (to an external observer), those which are visible, and those which are internal to the protocol.

First we consider only the case in which the choice of the secret takes place entirely at the beginning, and is based on a known distribution. Furthermore we focus on fully probabilistic automata. Later in the chapter we will relax these constraints.

Definition 3.3.1 (Information-Hiding System)**.** An information-hiding system (IHS) is a quadruple $\mathcal{I} = (M, \Sigma_{\mathcal{S}}, \Sigma_{\mathcal{O}}, \Sigma_{\tau})$ where $M = (Q, \Sigma, \hat{q}, \alpha)$ is a fully probabilistic automaton, $\Sigma = \Sigma_{\mathcal{S}} \cup \Sigma_{\mathcal{O}} \cup \Sigma_{\tau}$ where $\Sigma_{\mathcal{S}}$, $\Sigma_{\mathcal{O}}$, and Σ_{τ} are pairwise disjoint sets of secret, observable, and internal actions, and α satisfies the following restrictions:

1. $\alpha(\hat{q}) \in \mathcal{D}(\Sigma_{\mathcal{S}} \times Q)$,
2. $\forall s \in \Sigma_{\mathcal{S}}\ \exists! q\ .\ \alpha(\hat{q})(s, q) \neq 0$,
3. $\alpha(q) \in \mathcal{D}(\Sigma_{\mathcal{O}} \cup \Sigma_{\tau} \times Q)$ for $q \neq \hat{q}$,
4. $\mathsf{CyclesA}(M) \subseteq \Sigma_{\tau}$,
5. $\mathbb{P}(\mathsf{CPaths}(M) \cap \mathsf{Paths}^{\star}(M)) = 1$.

The first two restrictions are on the initial state and mean that only secret actions can happen there (1) and each of those actions must have non null probability and occur only once (2), Restriction 3 forbids secret actions to happen in the rest of the automaton, and Restriction 4 specifies that only internal actions can occur inside cycles, this restriction is necessary in order to make sure that the channel associated to the IHS has finitely many inputs and outputs. Finally, Restriction 5 means that infinite computations have probability 0 and therefore we can ignore them.

We now show how to interpret an IHS as a noisy channel. We call $\mathit{trace}_{\Sigma_{\mathcal{S}}}(\sigma)$ and $\mathit{trace}_{\Sigma_{\mathcal{O}}}(\sigma)$ the *secret* and *observable* traces of σ, respectively. For $s \in \Sigma_{\mathcal{S}}^*$, we define $[s] \triangleq \{\sigma \in \mathsf{CPaths}(M) \mid \mathit{trace}_{\Sigma_{\mathcal{S}}}(\sigma) = s\}$; similarly for $o \in \Sigma_{\mathcal{O}}^*$, we define $[o] \triangleq \{\sigma \in \mathsf{CPaths}(M) \mid \mathit{trace}_{\Sigma_{\mathcal{O}}}(\sigma) = o\}$.

Definition 3.3.2. Given an IHS $\mathcal{I} = (M, \Sigma_{\mathcal{S}}, \Sigma_{\mathcal{O}}, \Sigma_\tau)$, its noisy channel is $(\mathcal{S}, \mathcal{O}, P)$, where $\mathcal{S} \triangleq \Sigma_{\mathcal{S}}$, $\mathcal{O} \triangleq \mathit{trace}_{\Sigma_{\mathcal{O}}}(\mathsf{CPaths}(M))$, and $P(o \mid s) \triangleq \mathbb{P}([o] \mid [s])$. The a priori distribution $\pi \in \mathcal{D}(\mathcal{S})$ of $\mathcal{I}$ is defined by $\pi(s) \triangleq \alpha(\hat{q})(s, \cdot)$. If $\mathcal{C}$ is the noisy channel of $\mathcal{I}$, the multiplicative and additive leakage of $\mathcal{I}$ are naturally defined as

$$\mathcal{L}_\times(\mathcal{I}) \triangleq \mathcal{L}_\times(\mathcal{C}, \pi) \quad \text{and} \quad \mathcal{L}_+(\mathcal{I}) \triangleq \mathcal{L}_+(\mathcal{C}, \pi).$$

Example 3.3.3. Crowds [RR98] is a well-known anonymity protocol, in which a user (called the *initiator*) wants to send a message to a web server without revealing his identity. To achieve this, he routes the message through a crowd of users participating in the protocol. Routing is as follows. In the beginning, the initiator randomly selects a user (called a *forwarder*), possibly himself, and forwards the request to him. A forwarder performs a probabilistic choice. With probability p (a parameter of the protocol) he selects a new user and again forwards the message. With probability $1-p$ he sends the message directly to the server. One or more users can be *corrupted* and collaborate with each other to try to find the identity of the initiator.

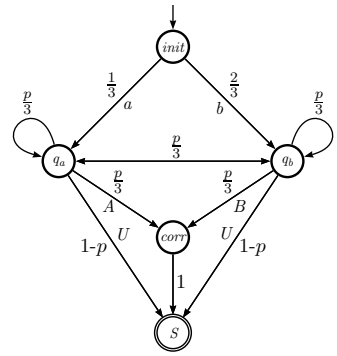


Figure 3.1: Crowds Protocol

We now show how to model Crowds as an IHS for 2 honest and 1 corrupted user. We assume that the corrupted user immediately forwards messages to the server, as there is no further information to be gained for him by bouncing the message back.

Figure 3.1 shows the corresponding automaton[2]. Actions a and b are secret and represent who initiates the protocol (the sender's identity); actions A, B, and U are observable; A and B represent who forwards the message to the corrupted user; U represents the fact that the message arrives at the

[2]For the sake of simplicity, we allow the initiator of the protocol to send the message to the server also in the first step of the protocol.

server undetected by the corrupted user. We assume U to be observable to represent the possibility that the message is made publically available at the server's site.

The channel associated to this IHS has $\mathcal{S} = \{a, b\}$, $\mathcal{O} = \{A, B, U\}$, and a priori distribution $\pi(a) = \frac{1}{3}, \pi(b) = \frac{2}{3}$. We now proceed to show how to compute its channel matrix.

3.4 Reachability analysis approach

This section presents a method to compute the matrix of joint probabilities $P_\wedge$ associated to an IHS, defined as

$$P_\wedge(s, o) \triangleq \mathbb{P}([s] \cap [o]) \text{ for all } s \in \mathcal{S} \text{ and } o \in \mathcal{O}.$$

We omit the subscript $\wedge$ when no confusion arises. From $P_\wedge$ we can derive the channel matrix by dividing $P_\wedge(s, o)$ by $\pi(s)$. The leakage can be computed directly from $P_\wedge$, using the second form of the a posteriori vulnerability in (4.1).

We write x_q^λ for the probability of the set of paths with trace $\lambda \in (\Sigma_\mathcal{S} \cup \Sigma_\mathcal{O})^\star$ starting from the state q of M:

$$x_q^\lambda \triangleq \mathbb{P}_q([\lambda]_q),$$

where $[\lambda]_q \triangleq \{\sigma \in \mathsf{CPaths}_q(M) \mid \mathit{trace}_{\Sigma_\mathcal{S} \cup \Sigma_\mathcal{O}}(\sigma) = \lambda\}$. The following key lemma shows the linear relation between the x_q^λ's. We assume, w.l.o.g., that the IHS has a unique final state q_f.

Lemma 3.4.1. Let $\mathcal{I} = (M, \Sigma_\mathcal{S}, \Sigma_\mathcal{O}, \Sigma_\tau)$ be an IHS. For all $\lambda \in (\Sigma_\mathcal{S} \cup \Sigma_\mathcal{O})^\star$ and $q \in Q$ we have

$$\begin{aligned}
x_{q_f}^\epsilon &= 1, \\
x_{q_f}^\lambda &= 0 \quad \text{for } \lambda \neq \epsilon, \\
x_q^\epsilon &= \textstyle\sum_{h \in \Sigma_\tau} \sum_{q' \in \mathsf{succ}(q)} \alpha(q)(h, q') \cdot x_{q'}^\epsilon \qquad \text{for } q \neq q_f, \\
x_q^\lambda &= \textstyle\sum_{q' \in \mathsf{succ}(q)} \alpha(q)(\mathsf{first}(\lambda), q') \cdot x_{q'}^{\mathsf{tail}(\lambda)} \\
&\quad + \textstyle\sum_{h \in \Sigma_\tau} \alpha(q)(h, q') \cdot x_{q'}^\lambda \qquad \text{for } \lambda \neq \epsilon \text{ and } q \neq q_f.
\end{aligned}$$

Note that, for $s \in \mathcal{S}$ and $o \in \mathcal{O}$ we have $\mathbb{P}([s] \cap [o]) = x_{\hat{q}}^{so}$.

Using this lemma, one can compute joint probabilities by solving the system of linear equations in the variables x_q^λ's. It is possible that the system has multiple solutions; in that case the required solution is the minimal one.

Example 3.4.2. Continuing with the Crowds example, we show how to compute joint probabilities. Note that $q_f = S$. The linear equations from Lemma 3.4.1 are

$$\begin{array}{llll}
x_{init}^{aA} = \frac{1}{3} \cdot x_{q_a}^A, & x_{q_a}^A = \frac{p}{3} \cdot x_{q_a}^A + \frac{p}{3} \cdot x_{q_b}^A + \frac{p}{3} \cdot x_{corr}^\epsilon, & x_{corr}^A = x_S^A, \\
x_{init}^{bA} = \frac{2}{3} \cdot x_{q_b}^A, & x_{q_b}^A = \frac{p}{3} \cdot x_{q_a}^A + \frac{p}{3} \cdot x_{q_b}^A + \frac{p}{3} \cdot x_{corr}^A, & x_S^A = 0, \\
x_{init}^{aB} = \frac{1}{3} \cdot x_{q_a}^B, & x_{q_a}^B = \frac{p}{3} \cdot x_{q_a}^B + \frac{p}{3} \cdot x_{q_b}^B + \frac{p}{3} \cdot x_{corr}^B, & x_{corr}^B = x_S^B, \\
x_{init}^{bB} = \frac{2}{3} \cdot x_{q_b}^B, & x_{q_b}^B = \frac{p}{3} \cdot x_{q_a}^B + \frac{p}{3} \cdot x_{q_b}^B + \frac{p}{3} \cdot x_{corr}^\epsilon, & x_S^B = 0, \\
x_{init}^{aU} = \frac{1}{3} \cdot x_{q_a}^U, & x_{q_a}^U = \frac{p}{3} \cdot x_{q_a}^U + \frac{p}{3} \cdot x_{q_b}^U + (1-p) \cdot x_S^\epsilon, & x_{corr}^\epsilon = x_S^\epsilon, \\
x_{init}^{bU} = \frac{2}{3} \cdot x_{q_b}^U, & x_{q_b}^U = \frac{p}{3} \cdot x_{q_a}^U + \frac{p}{3} \cdot x_{q_b}^U + (1-p) \cdot x_S^\epsilon, & x_S^\epsilon = 1.
\end{array}$$

Let us fix $p = 0.9$. By solving the system of linear equations we obtain

$$\begin{array}{lll}
x_{init}^{aA} = \frac{7}{40}, & x_{init}^{aB} = \frac{3}{40}, & x_{init}^{aU} = \frac{1}{12}, \\
x_{init}^{bA} = \frac{3}{20}, & x_{init}^{bB} = \frac{7}{20}, & x_{init}^{bU} = \frac{1}{6}.
\end{array}$$

	A	B	U
a	$\frac{21}{40}$	$\frac{9}{40}$	$\frac{1}{4}$
b	$\frac{9}{40}$	$\frac{21}{40}$	$\frac{1}{4}$

We can now compute the channel matrix by dividing each x_{init}^{so} by $\pi(s)$. This matrix is shown in the figure above.

3.4.1 Complexity Analysis

We now analyze the computational complexity for the computation of the channel matrix of a simple IHS. Note that the only variables (from the system of equations in Lemma 3.4.1) that are relevant for the computation of the channel matrix are those x_q^λ for which it is possible to get the trace

λ starting from state q. As a rough overestimate, for each state q, there are at most $|\mathcal{S}| \cdot |\mathcal{O}|$ λ's possible: in the initial state one can have every secret and every observable, in the other states no secret is possible and only a suffix of an observable can occur. This gives at most $|Q| \cdot |\mathcal{S}| \cdot |\mathcal{O}|$ variables. Therefore, we can straightforwardly obtain the desired set of values in $O((|Q| \cdot |\mathcal{S}| \cdot |\mathcal{O}|)^3)$ time (using Gaussian Elimination). Note that using Strassen's methods the exponent reduces to 2.807, this consideration applies to similar results in the rest of the chapter as well.

Because secret actions can happen only at the beginning, the system of equations has a special form. The variables of the form $x^{so}_{\hat{q}}$ only depend on variables of the form x^o_q (with varying o and $q \neq \hat{q}$) and not on each other. Hence, we can first solve for all variables of the form x^o_q and then compute the remaining few of the form $x^{so}_{\hat{q}}$. Required time for the first step is $O((|\mathcal{O}| \cdot |Q|)^3)$ and the time for the second step can be ignored.

Finally, in some cases not only do the secret actions happen only at the beginning of the protocol, but the observable actions happen only at the end of the protocol, i.e., after taking a transition with an observable action, the protocol only performs internal actions (this is, for instance, the case for our model of Crowds). In this case, one might as well enter a unique terminal state q_f after an observable action happens. Then the only relevant variables are of the form $x^{so}_{\hat{q}}$, x^o_q, and $x^\epsilon_{q_f}$; the $x^{so}_{\hat{q}}$ only depends on the x^o_q, the x^o_q only depend on $x^o_{q'}$ (with the same o, but varying q's) and on $x^\epsilon_{q_f}$ and $x^\epsilon_{q_f} = 1$. Again ignoring the variables $x^{so}_{\hat{q}}$ for complexity purposes, the system of equations has a block form with $|\mathcal{O}|$ blocks of (at most) $|Q|$ variables each. Hence the complexity in this case decreases to $O(|\mathcal{O}| \cdot |Q|^3)$.

3.5 The Iterative Approach

We now propose a different approach to compute channel matrices and leakage. The idea is to iteratively construct the channel matrix of a system by adding probabilities of sets of paths containing paths with the same observable trace o and secret trace s to the (s, o) entry of the matrix.

One reason for this approach is that it allows us to borrow techniques

from quantitative counterexample generation. This includes the possibility of using or extending counterexample generation tools to compute channel matrices or leakage. Another reason for this approach is the relationship with debugging. If a (specification of a) system has a high leakage, the iterative approach allows us to determine which parts of the system contribute most to the high leakage, possibly pointing out flaws of the protocol. Finally, if the system under consideration is very large, the iterative approach allows us to only approximate the leakage (by not considering all paths, but only the most relevant ones) under strict guarantees about the accuracy of the approximation. We will focus on the multiplicative leakage; similar results can be obtained for the additive case.

3.5.1 Partial matrices

We start by defining a sequence of matrices converging to the channel matrix by adding the probability of complete paths one by one. We also define partial version of the a posteriori vulnerability and the leakage. Later, we show how to use techniques from quantitative counterexample generation to add probabilities of many (maybe infinitely many) complete paths all at once.

Definition 3.5.1. Let $\mathcal{I} = (M, \Sigma_{\mathcal{S}}, \Sigma_{\mathcal{O}}, \Sigma_\tau)$ be an IHS, π its a priori distribution, and $\sigma_1, \sigma_2, \ldots$ an enumeration of the set of complete paths of M. We define the *partial matrices* $P^k : \mathcal{S} \times \mathcal{O} \to [0,1]$ as follows

$$P^0(o|s) \triangleq 0, \quad P^{k+1}(o|s) \triangleq \begin{cases} P^k(o|s) + \frac{\mathbb{P}(\langle \sigma_{k+1} \rangle)}{\pi(s)} & \text{if } \mathit{trace}_{\Sigma_{\mathcal{O}}}(\sigma_{k+1}) = o \\ & \text{and } \mathit{trace}_{\Sigma_{\mathcal{S}}}(\sigma_{k+1}) = s, \\ P^k(o|s) & \text{otherwise.} \end{cases}$$

We define the *partial a posteriori vulnerability* $V^k_{S,O}$ as $\sum_o \mathsf{max}_s\, P^k(o|s) \cdot \pi(s)$, and the *partial multiplicative leakage* $\mathcal{L}^k_\times(\mathcal{I})$ as $V^k_{S,O} / \mathsf{max}_s\, \pi(s)$.

The following lemma states that partial matrices, a posteriori vulnerability, and leakage converge to the correct values.

Lemma 3.5.2. Let $\mathcal{I} = (M, \Sigma_{\mathcal{S}}, \Sigma_{\mathcal{O}}, \Sigma_{\tau})$ be an IHS. Then

1. $P^k(o|s) \leq P^{k+1}(o|s)$, and $\mathsf{lim}_{k\to\infty} P^k(o|s) = P(o|s)$,
2. $V^k_{S,O} \leq V^{k+1}_{S,O}$, and $\mathsf{lim}_{k\to\infty} V^k_{S,O} = V(S|O)$,
3. $\mathcal{L}^k_{\times}(\mathcal{I}) \leq \mathcal{L}^{k+1}_{\times}(\mathcal{I})$, and $\mathsf{lim}_{k\to\infty} \mathcal{L}^k_{\times}(\mathcal{I}) = \mathcal{L}_{\times}(\mathcal{I})$.

Since rows must sum up to 1, this technique allow us to compute matrices up to given error ϵ. We now show how to estimate the error in the approximation of the multiplicative leakage.

Proposition 3.5.1. *Let $(M, \Sigma_{\mathcal{S}}, \Sigma_{\mathcal{O}}, \Sigma_{\tau})$ be an IHS. Then we have*

$$\mathcal{L}^k_{\times}(\mathcal{I}) \leq \mathcal{L}_{\times}(\mathcal{I}) \leq \mathcal{L}^k_{\times}(\mathcal{I}) + \sum_{i=1}^{|\mathcal{S}|}(1 - p^k_i),$$

where p^k_i denotes the mass probability of the i-th row of P^k, i.e. $p^k_i \triangleq \sum_o P^k(o|s_i)$.

3.5.2 On the computation of partial matrices.

After showing how partial matrices can be used to approximate channel matrices and leakage we now turn our attention to accelerating the convergence. Adding most likely paths first is an obvious way to increase the convergence rate. However, since automata with cycles have infinitely many paths, this (still) gives an infinite amount of path to process. Processing many paths at once (all having the same observable and secret trace) tackles both issues at the same time: it increases the rate of convergence and can deal with infinitely many paths at the same time,

Interestingly enough, these issues also appear in *quantitative counterexample generation*. In that area, several techniques have already been provided to meet the challenges; we show how to apply those techniques in the current context. We consider two techniques: one is to group paths together using regular expressions, the other is to group paths together using strongly connected component analysis.

Regular expressions. In [Daw05], regular expressions containing probability values are used to reason about traces in Markov Chains. This idea is used in [DHK08] in the context of counterexample generation to group together paths with the same observable behaviour. The regular expression there are over pairs $\langle p, q\rangle$ with p a probability value and q a state, to be able to track both probabilities and observables. We now use the same idea to group together paths with the same secret action and the same observable actions.

We consider regular expressions over triples of the form $\langle a, p, q\rangle$ with $p \in [0, 1]$ a probability value, $a \in \Sigma$ an action label and $q \in Q$ a state. Regular expressions represent sets of paths as in [DHK08]. We also take the probability value of such a regular expression from that article.

Definition 3.5.3. The function $val : \mathcal{R}(\Sigma) \to \mathbb{R}$ evaluates regular expressions:

$$\begin{array}{rcl@{\qquad}rcl} val(\epsilon) & \triangleq & 1, & val(r \cdot r') & \triangleq & val(r) \times val(r'), \\ val(\langle a, p, q\rangle) & \triangleq & p, & val(r^*) & \triangleq & 1 \quad \text{if } val(r) = 1, \\ val(r + r') & \triangleq & val(r) + val(r'), & val(r^*) & \triangleq & \frac{1}{1-val(r)} \quad \text{if } val(r) \neq 1. \end{array}$$

The idea is to obtain regular expressions representing sets of paths of M, each regular expression will contribute in the approximation of the channel matrix and leakage. Several algorithms to translate automata into regular expressions have been proposed (see [Neu05]). Finally, each term of the regular expression obtained can be processed separately by adding the corresponding probabilities [Daw05] to the partial matrix.

As mentioned before, all paths represented by the regular expression should have the same observable and secret trace in order to be able to add its probability to a single element of the matrix. To ensure that condition we request the regular expression to be normal, i.e., of the form $r_1 + \cdots + r_n$ with the r_i containing no $+$'s.

We will now describe this approach by an example.

Example 3.5.4. We used JFLAP 7.0 [JFL] to obtain the regular expression

$r \triangleq r_1 + r_2 + \cdots + r_{10}$ equivalent to the automaton in Figure 3.1.

$$
\begin{aligned}
r_1 &\triangleq \langle b, \tfrac{2}{3}, q_b\rangle \cdot \hat{r}^\star \cdot \langle B, 0.3, \mathit{corr}\rangle \cdot \langle \tau, 1, S\rangle,\\
r_2 &\triangleq \langle b, \tfrac{2}{3}, q_b\rangle \cdot \hat{r}^\star \cdot \langle \tau, 0.3, q_a\rangle \cdot \langle \tau, 0.3, q_a\rangle^\star \cdot \langle A, 0.3, \mathit{corr}\rangle \cdot \langle \tau, 1, S\rangle,\\
r_3 &\triangleq \langle a, \tfrac{1}{3}, q_a\rangle \cdot \langle \tau, 0.3, q_a\rangle^\star \cdot \langle A, 0.3, \mathit{corr}\rangle \cdot \langle \tau, 1, S\rangle,\\
r_4 &\triangleq \langle b, \tfrac{2}{3}, q_b\rangle \cdot \hat{r}^\star \cdot \langle U, 0.1, S\rangle,\\
r_5 &\triangleq \langle a, \tfrac{1}{3}, q_a\rangle \cdot \langle \tau, 0.3, q_a\rangle^\star \cdot \langle \tau, 0.3, q_b\rangle \cdot \hat{r}^\star \cdot \langle B, 0.3, \mathit{corr}\rangle \cdot \langle \tau, 1, S\rangle,\\
r_6 &\triangleq \langle b, \tfrac{2}{3}, q_b\rangle \cdot \hat{r}^\star \cdot \langle \tau, 0.3, q_a\rangle \cdot \langle \tau, 0.3, q_a\rangle^\star \cdot \langle U, 0.1, S\rangle,\\
r_7 &\triangleq \langle a, \tfrac{1}{3}, q_a\rangle \cdot \langle \tau, 0.3, q_a\rangle^\star \cdot \langle U, 0.1, S\rangle,\\
r_8 &\triangleq \langle a, \tfrac{1}{3}, q_a\rangle \cdot \langle \tau, 0.3, q_a\rangle^\star \cdot \langle \tau, 0.3, q_b\rangle \cdot \hat{r}^\star \cdot \langle \tau, 0.3, q_a\rangle \cdot \langle \tau, 0.3, q_a\rangle^\star \cdot\\
&\quad \langle A, 0.3, \mathit{corr}\rangle \cdot \langle \tau, 1, S\rangle,\\
r_9 &\triangleq \langle a, \tfrac{1}{3}, q_a\rangle \cdot \langle \tau, 0.3, q_a\rangle^\star \cdot \langle \tau, 0.3, q_b\rangle \cdot \hat{r}^\star \cdot \langle U, 0.1, S\rangle,\\
r_{10} &\triangleq \langle a, \tfrac{1}{3}, q_a\rangle \cdot \langle \tau, 0.3, q_a\rangle^\star \cdot \langle \tau, 0.3, q_b\rangle \cdot \hat{r}^\star \cdot \langle \tau, 0.3, q_a\rangle \cdot \langle \tau, 0.3, q_a\rangle^\star \cdot \langle U, 0.1, S\rangle,
\end{aligned}
$$

where $\hat{r} \triangleq (\langle \tau, 0.3, q_b\rangle^\star \cdot (\langle \tau, 0.3, q_a\rangle \cdot \langle \tau, 0.3, q_a\rangle^\star \cdot \langle \tau, 0.3, q_b\rangle)^\star)$. We also note

$$
\begin{aligned}
\mathit{val}(r_1) &= \tfrac{7}{20}\ (b, B), & \mathit{val}(r_2) &= \tfrac{3}{20}\ (b, A), & \mathit{val}(r_3) &= \tfrac{1}{7}\ (a, A),\\
\mathit{val}(r_4) &= \tfrac{7}{60}\ (b, U) & \mathit{val}(r_5) &= \tfrac{3}{40}\ (a, B), & \mathit{val}(r_6) &= \tfrac{1}{20}\ (b, U),\\
\mathit{val}(r_7) &= \tfrac{1}{21}\ (a, U), & \mathit{val}(r_8) &= \tfrac{9}{280}\ (a, A) & \mathit{val}(r_9) &= \tfrac{1}{40}\ (a, U),\\
\mathit{val}(r_{10}) &= \tfrac{3}{280}\ (a, U). & & & &
\end{aligned}
$$

where the symbols between parenthesis denote the secret and observable traces of each regular expression.

Now we have all the ingredients needed to define partial matrices using regular expressions.

Definition 3.5.5. Let $\mathcal{I} = (M, \Sigma_\mathcal{S}, \Sigma_\mathcal{O}, \Sigma_\tau)$ be an IHS, π its a priori distribution, and $r = r_1 + r_2 + \cdots + r_n$ a regular expression equivalent to M in normal form. We define for $k = 0, 1, \ldots, n$ the matrices $P^k : S \times O \to [0, 1]$ as follows

$$P^k(o|s) \quad = \quad \begin{cases} 0 & \text{if } k = 0, \\ P^{k-1}(o|s) + \frac{val(r_k)}{\pi(s)} & \text{if } k \neq 0 \text{ and } trace_{\Sigma_{\mathcal{O}}}(r_k) = o \\ & \text{and } trace_{\Sigma_{\mathcal{S}}}(r_k) = s, \\ P^{k-1}(o|s) & \text{otherwise.} \end{cases}$$

Note that in the context of Definition 3.5.5, we have $P^n = P$.

SCC analysis approach. In [ADvR08], paths that only differ in the way they traverse strongly connected components (SCC's) are grouped together. Note that in our case, such paths have the same secret and observable trace since secret and observable actions cannot occur on cycles. Following [ADvR08], we first abstract away the SCC's, leaving only probabilistic transitions that go immediately from an entry point of the SCC to an exit point (called input and output states in [ADvR08]). This abstraction happens in such a way that the observable behaviour of the automaton does not change.

Instead of going into technical details (which also involves translating the work [ADvR08] from Markov Chains to fully probabilistic automata), we describe the technique by an example.

Example 3.5.6. Figure 3.2 shows the automaton obtained after abstracting SCC. In the following we show the set of complete paths of the automaton, together with their corresponding probabilities and traces

$$\begin{array}{lcll}
\sigma_1 & \triangleq & init \xrightarrow{a} q_a \xrightarrow{A} corr \xrightarrow{\tau} S, & \mathbb{P}(\sigma_1) = \frac{7}{40}, \quad (a, A), \\
\sigma_2 & \triangleq & init \xrightarrow{b} q_b \xrightarrow{B} corr \xrightarrow{\tau} S, & \mathbb{P}(\sigma_2) = \frac{7}{20}, \quad (b, B), \\
\sigma_3 & \triangleq & init \xrightarrow{a} q_a \xrightarrow{U} S, & \mathbb{P}(\sigma_3) = \frac{1}{12}, \quad (a, U), \\
\sigma_4 & \triangleq & init \xrightarrow{b} q_b \xrightarrow{U} S, & \mathbb{P}(\sigma_4) = \frac{1}{6}, \quad (b, U), \\
\sigma_5 & \triangleq & init \xrightarrow{a} q_a \xrightarrow{B} corr \xrightarrow{\tau} S, & \mathbb{P}(\sigma_5) = \frac{3}{40}, \quad (a, B), \\
\sigma_6 & \triangleq & init \xrightarrow{b} q_b \xrightarrow{A} corr \xrightarrow{\tau} S, & \mathbb{P}(\sigma_6) = \frac{3}{20}, \quad (b, A).
\end{array}$$

Note that the SCC analysis approach groups more paths together (for instance σ_1 group together the same paths than the regular expressions r_3 and r_8 in the examples of this section), as a result the convergence rate of this technique is higher. On the other hand, regular expressions are more informative providing more precise feedback.

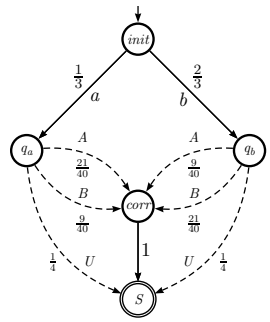


Figure 3.2: Crowds after the SCC analysis

3.5.3 Identifying high-leakage sources

We now describe how to use the techniques presented in this section to identify sources of high leakage of the system. Remember that the a posteriori vulnerability can be expressed in terms of joint probabilities

$$V(S \mid O) \;=\; \sum_{o} \max_{s} \mathbb{P}([s] \cap [o]).$$

This suggests that, in case we want to identify parts of the system generating high leakage, we should look at the sets of paths $[o_1] \cap [s_1], \ldots, [o_n] \cap [s_n]$ where $\{o_1, \ldots o_n\} = \mathcal{O}$ and $s_i \in \arg\left(\max_s \mathbb{P}([o_i] \cap [s])\right)$. In fact, the multiplicative leakage is given dividing $V(S \mid O)$ by $V(S)$, but since $V(S)$ is a constant value (i.e., it does not depend on the row) it does not play a role here. Similarly for the additive case.

The techniques presented in this section allow us to obtain such sets and, furthermore, to partition them in a convenient way with the purpose of identifying states/parts of the system that contribute the most to its high probability. Indeed, this is the aim of the counterexample generation techniques previously presented. For further details on how to debug sets of paths and why these techniques meet that purpose we refer to [AL08, DHK08, ADvR08].

Example 3.5.7. To illustrate these ideas, consider the path σ_1 of the previous example; this path has maximum probability for the observable A. By inspecting the path we find the transition with high probability

$q_a \xrightarrow{A} corr$. This suggests to the debugger that the corrupted user has an excessively high probability of intercepting a message from user a in case he is the initiator.

In case the debugger requires further information on how corrupted users can intercept messages, the regular expression approach provides further/more-detailed information. For instance, we obtain further information by looking at regular expressions r_3 and r_8 instead of path σ_1 (in particular it is possible to visualize the different ways the corrupted user can intercept the message of user a when he is the generator of the message).

3.6 Information Hiding Systems with Variable a Priori

In Section 3.3 we introduced a notion of IHS in which the distribution over secrets is fixed. However, when reasoning about security protocols this is often not the case. In general we may assume that an adversary knows the distribution over secrets in each particular instance, but the protocol should not depend on it. In such scenario we want the protocol to be secure, i.e. ensuring low enough leakage, for every possible distribution over secrets. This leads to the definition of maximum leakage.

Definition 3.6.1 ([Smi09, BCP09])**.** Given a noisy channel $\mathcal{C} = (\mathcal{S}, \mathcal{O}, P)$, we define the maximum multiplicative and additive leakage (respectively) as

$$\mathcal{ML}_\times(\mathcal{C}) \triangleq \max_{\pi \in \mathcal{D}(\mathcal{S})} \mathcal{L}_\times(\mathcal{C}, \pi), \qquad \text{and} \qquad \mathcal{ML}_+(\mathcal{C}) \triangleq \max_{\pi \in \mathcal{D}(\mathcal{S})} \mathcal{L}_+(\mathcal{C}, \pi).$$

In order to model this new scenario where the distribution over secrets may change, the selection of the secret is modeled as *nondeterministic choice*. In this way such a distribution remains undefined in the protocol/automaton. We still assume that the choice of the secret happens at the beginning, and that we have only one secret per run. We call such automaton an IHS *with variable a priori*.

Definition 3.6.2. An IHS with variable a priori is a quadruple $\mathcal{I} = (M, \Sigma_{\mathcal{S}}, \Sigma_{\mathcal{O}}, \Sigma_{\tau})$ where $M = (Q, \Sigma, \hat{q}, \alpha)$ is a probabilistic automaton, $\Sigma = \Sigma_{\mathcal{S}} \cup \Sigma_{\mathcal{O}} \cup \Sigma_{\tau}$ where $\Sigma_{\mathcal{S}}$, $\Sigma_{\mathcal{O}}$, and Σ_{τ} are pairwise disjoint sets of secret, observable, and internal actions, and α satisfies the following restrictions:

1. $\alpha(\hat{q}) \subseteq \mathcal{D}(\Sigma_{\mathcal{S}} \times Q)$,
2. $|\alpha(\hat{q})| = |\mathcal{S}| \wedge \forall s \in \Sigma_{\mathcal{S}} \,.\, \exists\, q \,.\, \pi(s, q) = 1$, for some $\pi \in \alpha(\hat{q})$,
3. $\alpha(q) \subseteq \mathcal{D}(\Sigma_{\mathcal{O}} \cup \Sigma_{\tau} \times Q)$ and $|\alpha(q)| \leq 1$, for all $q \neq \hat{q}$,
4. $\mathsf{CyclesA}(M) \subseteq \Sigma_{\tau}$,
5. $\forall q, s\ \forall \pi \in \alpha(\hat{q}) \,.\, (\pi(s, q) = 1 \Rightarrow \mathbb{P}(\mathsf{CPaths}_q(M) \cap \mathsf{Paths}^*_q(M)) = 1)$.

Restrictions 1, 2 and 3 imply that the secret choice is non deterministic and happens only at the beginning. Additionally, 3 means that all the other choices are probabilistic. Restriction 4 ensures that the channel associated to the IHS has finitely many inputs and outputs. Finally, 5 implies that, after we have chosen a secret, every computation terminates except for a set with null probability.

Given an IHS with variable a priori, by fixing the a priori distribution we can obtain a standard IHS in the obvious way:

Definition 3.6.3. Let $\mathcal{I} = ((Q, \Sigma, \hat{q}, \alpha), \Sigma_{\mathcal{S}}, \Sigma_{\mathcal{O}}, \Sigma_{\tau})$ be an IHS with variable a priori and π a distribution over $\mathcal{S}$. We define the IHS associated to $(\mathcal{I}, \pi)$ as $\mathcal{I}_{\pi} = ((Q, \Sigma, \hat{q}, \alpha'), \Sigma_{\mathcal{S}}, \Sigma_{\mathcal{O}}, \Sigma_{\tau})$ with $\alpha'(q) = \alpha(q)$ for all $q \neq \hat{q}$ and $\alpha'(\hat{q})(s, \cdot) = \pi(s)$.

The following result says that the conditional probabilities associated to an IHS with variable a priori are *invariant* with respect to the a priori distribution. This is fundamental in order to interpret the IHS as a channel.

Proposition 3.6.1. *Let $\mathcal{I}$ be an* IHS *with variable a priori. Then for all $\pi, \pi' \in \mathcal{D}(\mathcal{S})$ such that $\pi(s) \neq 0$ and $\pi'(s) \neq 0$ for all $s \in \mathcal{S}$ we have that $P_{\mathcal{I}_{\pi}} = P_{\mathcal{I}_{\pi'}}$.*

Proof. The secret s appears only once in the tree and only at the beginning of paths, hence $\mathbb{P}([s] \cap [o]) = \alpha'(\hat{q})(s,\cdot)\,\mathbb{P}_{q_s}([o])$ and $\mathbb{P}([s]) = \alpha'(\hat{q})(s,\cdot)$. Therefore $\mathbb{P}([o] \mid [s]) = \mathbb{P}_{q_s}([o])$, where q_s is the state after performing s. While $\alpha'(\hat{q})(s,\cdot)$ is different in $\mathcal{I}_\pi$ and $\mathcal{I}_{\pi'}$, $\mathbb{P}_{q_s}([o])$ is the same, because it only depends on the parts of the paths after the choice of the secret. □

Note that, although in the previous proposition we exclude input distributions with zeros, the concepts of vulnerability and leakage also make sense for these distributions[3].

This result implies that we can define the channel matrix of an IHS $\mathcal{I}$ with variable a priori as the channel matrix of $\mathcal{I}_\pi$ for any π, and we can compute it, or approximate it, using the same techniques of previous sections. Similarly we can compute or approximate the leakage for any given π.

We now turn the attention to the computation of the maximum leakage. The following result from the literature is crucial for our purposes.

Proposition 3.6.2 ([BCP09])**.** *Given a channel $\mathcal{C}$, we have* $\arg\max_{\pi\in\mathcal{D}(\mathcal{S})} \mathcal{L}_\times(\mathcal{C},\pi)$ *is the uniform distribution, and* $\arg\max_{\pi\in\mathcal{D}(\mathcal{S})} \mathcal{L}_+(\mathcal{C},\pi)$ *is a* corner point *distribution, i.e. a distribution π such that $\pi(s) = \frac{1}{\kappa}$ on κ elements of $\mathcal{S}$, and $\pi(s) = 0$ on all the other elements.*

As an obvious consequence, we obtain:

Corollary 3.6.3. Given an IHS $\mathcal{I}$ with variable a priori, we have $\mathcal{ML}_\times(\mathcal{I}) = \mathcal{L}_\times(\mathcal{I}_\pi)$, where π is the uniform distribution, and $\mathcal{ML}_+(\mathcal{I}) = \mathcal{L}_+(\mathcal{I}_{\pi'})$, where π' is a corner point distribution.

Corollary 3.6.3 gives us a method to compute the maxima leakages of $\mathcal{I}$. In the multiplicative case the complexity is the same as for computing the leakage[4]. In the additive case we need to find the right corner point,

[3] We assume that conditional probabilities are extended by continuity on such distributions.

[4] Actually we can compute it even faster using an observation from [Smi09] which says that the leakage on the uniform distribution can be obtained simply by summing up the maximum elements of each column of the channel matrix.

which can be done by computing the leakages for all corner points and then comparing them. This method has exponential complexity (in $|\mathcal{S}|$) as the size of the set of corner points is $2^{|\mathcal{S}|}-1$. We conjecture that this complexity is intrinsic, i.e. that the problem is NP-complete[5].

3.7 Interactive Information Hiding Systems

We now consider extending the framework to interactive systems, namely to IHSs in which the secrets and the observables can alternate in an arbitrary way. The secret part of a run is then an element of $\Sigma_{\mathcal{S}}^*$, like the observable part is an element of $\Sigma_{\mathcal{O}}^*$. The idea is that such system models an interactive play between a source of secret information, and a protocol or program that may produce, each time, some observable in response. Since each choice is associated to one player of this "game", it seems natural to impose that in a choice the actions are either secret or observable/hidden, but not both.

The main novelty and challenge of this extension is that part of the secrets come after observable events, and may depend on them.

Definition 3.7.1. Interactive IHS's are defined as IHS's (Definition 3.3.1), except that Restrictions 1 to 3 are replaced by $\alpha(q) \in \mathcal{D}(\Sigma_{\mathcal{S}} \times Q) \cup \mathcal{D}(\Sigma - \Sigma_{\mathcal{S}} \times Q)$.

Example 3.7.2. Consider an Ebay-like auction protocol with one seller and two possible buyers, one rich and one poor. The seller first publishes the item he wants to sell, which can be either cheap or expensive. Then the two buyers start bidding. At the end, the seller looks at the profile of the bid winner and decides whether to sell the item or cancel the transaction. Figure 3.7 illustrates the automaton representing the protocol, for certain given probability distributions.

[5] After the publication of the article related to this chapter we have proved this conjecture. The proof will appear, together with other results, in a journal version of the article.

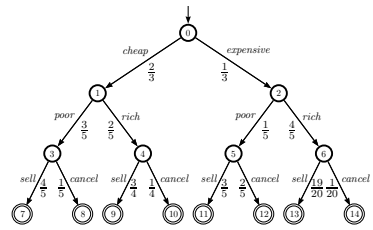


Figure 3.3: Ebay Protocol

We assume that the identities of the buyers are secret, while the price of the item and the seller's decision are observable. We ignore for simplicity the internal actions which are performed during the bidding phase. Hence $\Sigma_{\mathcal{O}} = \{cheap, expensive, sell, cancel\}$, $\Sigma_\tau = \emptyset$, $\mathcal{S} = \Sigma_{\mathcal{S}} = \{poor, rich\}$, and $\mathcal{O} = \{cheap, expensive\} \times \{sell, cancel\}$. The distributions on $\mathcal{S}$ and $\mathcal{O}$ are defined as usual. For instance we have $\mathbb{P}([cheap \;\; sell]) = \mathbb{P}(\{q_0 \xrightarrow{cheap} q_1 \xrightarrow{poor} q_3 \xrightarrow{sell} q_7, q_0 \xrightarrow{cheap} q_1 \xrightarrow{rich} q_3 \xrightarrow{sell} q_7\}) = \frac{2}{3} \cdot \frac{3}{5} \cdot \frac{4}{5} + \frac{2}{3} \cdot \frac{2}{5} \cdot \frac{3}{4} = \frac{13}{25}$.

Let us now consider how to model the protocol in terms of a noisy channel. It would seem natural to define the channel associated to the protocol as the triple $(\mathcal{S}, \mathcal{O}, P)$ where $P(o \mid s) = \mathbb{P}([o] \mid [s]) = \frac{\mathbb{P}([s] \cap [o])}{\mathbb{P}([s])}$. This is, indeed, the approach taken in [DJGP02]. For instance, with the protocol of Example 3.7.2, we would have:

$$\mathbb{P}([cheap \;\; sell] \mid [poor]) = \frac{\mathbb{P}([poor] \cap [cheap \;\; sell])}{\mathbb{P}([poor])} = \frac{\frac{2}{3} \cdot \frac{3}{5} \cdot \frac{4}{5}}{\frac{2}{3} \cdot \frac{3}{5} + \frac{1}{3} \cdot \frac{1}{5}} = \frac{24}{35}. \quad (3.2)$$

However, it turns out that in the interactive case (in particular when the secrets are not in the initial phase), it does not make sense to model the protocol in terms of a channel. At least, not a channel with input $\mathcal{S}$. In fact, the matrix of a channel is supposed to be *invariant* with respect to the input distribution (like in the case of the IHS's with variable a priori considered in previous section), and this is not the case here. The following is a counterexample.

Example 3.7.3. Consider the same protocol as in Example 3.7.2, but assume now that the distribution over the choice of the buyer is uniform, i.e. $\alpha(q_1)(poor, q_3) = \alpha(q_1)(rich, q_4) = \alpha(q_2)(poor, q_5) = \alpha(q_2)(rich, q_6) = \frac{1}{2}$. Then the conditional probabilities are different than those for Example

3.7.2. In particular, in contrast to (3.2), we have

$$\mathbb{P}([cheap \;\; sell] \,|\, [poor]) = \frac{\mathbb{P}([poor] \cap [cheap \;\; sell])}{\mathbb{P}([poor])} = \frac{\frac{2}{3} \cdot \frac{1}{2} \cdot \frac{4}{5}}{\frac{2}{3} \cdot \frac{1}{2} + \frac{1}{3} \cdot \frac{1}{2}} = \frac{8}{15}.$$

The above observation, i.e. the fact that the conditional probabilities depend on the input distribution, makes it unsound to reason about certain information-theoretic concepts in the standard way. For instance, the *capacity* is defined as the maximum mutual information over all possible input distributions, and the traditional algorithms to compute it are based on the assumption that the channel matrix remains the same while the input distribution variates. This does not make sense anymore in the interactive setting.

However, when the input distribution is fixed, the matrix of the joint probabilities is well defined as $P_\wedge(s, o) = \mathbb{P}([s] \cap [o])$, and can be computed or approximated using the same methods as for simple IHS's. The a priori probability and the channel matrix can then be derived in the standard way:

$$\pi(s) = \sum_o P_\wedge(s, o), \qquad P(o \mid s) = \frac{P_\wedge(s, o)}{\pi(s)}.$$

Thanks to the formulation (4.1) of the a posteriori vulnerability, the leakage can be computed directly using the joint probabilities.

Example 3.7.4. Consider the Ebay protocol $\mathcal{I}$ presented in Example 3.7.2. The matrix of the joint probabilities $P_\wedge(s, o)$ is:

	cheap sell	*cheap cancel*	*expensive sell*	*expensive cancel*
poor	$\frac{8}{25}$	$\frac{2}{25}$	$\frac{1}{25}$	$\frac{2}{75}$
rich	$\frac{1}{5}$	$\frac{1}{15}$	$\frac{19}{75}$	$\frac{1}{75}$

Furthermore $\pi(poor) = \frac{7}{15}$ and $\pi(rich) = \frac{8}{15}$. Hence we have $\mathcal{L}_\times(\mathcal{I}) = \frac{51}{40}$ and $\mathcal{L}_+(\mathcal{I}) = \frac{11}{75}$.

We note that our techniques to compute channel matrices and leakage extend smoothly to the case where secrets are not required to happen at the

beginning. However, no assumptions can be made about the occurrences of secrets (they do not need to occur at the beginning anymore). This increases the complexity of the reachability technique to $O((|\mathcal{S}|\cdot|\mathcal{O}|\cdot|Q|)^3)$. On the other hand, complexity bounds for the iterative approach remain the same.

3.8 Related Work

To the best of our knowledge, this is the first work dealing with the efficient computation of channel matrices and leakage. However, for the simple scenario, channel matrices can be computed using standard model checking techniques. Chatzikokolakis et al. [CPP08a] have used Prism [PRI] to model Crowds as a Markov Chain and compute its channel matrix. Each conditional probability $P(o|s)$ is computed as the probability of reaching a state where o holds starting from *the* state where s holds. Since for the simple version of IHS's secrets occur only once and before observables (as in Crowds), such a reachability probability equals $P(o|s)$. This procedure leads to $O(|\mathcal{S}|\cdot|\mathcal{O}|\cdot|\overline{Q}|^3)$ time complexity to compute the channel matrix, where $\overline{Q}$ is the space state of the Markov Chain.

Note that the complexity is expressed in terms of the space state of a Markov Chain instead of automaton. Since Markov Chains do not carry information in transitions they have a larger state space than an equivalent automaton. Figure 3.4 illustrates this: to model the automaton (left hand side) we need to encode the information in its transitions into states of the Markov Chain (right hand side). Therefore, the probability of seeing observation a and then c in the automaton can be computed as the probability of reaching the state ac. The Markov Chain used for modeling Crowds (in our two honest and one corrupted user configuration) has 27 states.

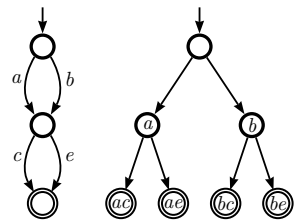


Figure 3.4: Automaton vs Markov Chain

For this reason we conjecture that our complexity $O(|\mathcal{O}|\cdot|Q|^3)$ is a

considerable improvement over the one on Markov Chains $O(|\mathcal{S}| \cdot |\mathcal{O}| \cdot |\overline{Q}|^3)$.

With respect to the interactive scenario, standard model checking techniques do not extend because multiple occurrences of the same secret are allowed. For example in our Ebay example, $P(cheap \;\; sell | rich)$ cannot be directly derived from reachability probabilities from the two different states of the automaton where *rich* holds.

Chapter 4

Information Hiding in Probabilistic Concurrent Systems

In this chapter we study the problem of information hiding in systems characterized by the coexistence of randomization and concurrency. Anonymity and Information Flow are examples of this notion. It is well known that the presence of nondeterminism, due to the possible interleavings and interactions of the parallel components, can cause unintended information leaks. The most established approach to solve this problem is to fix the strategy of the scheduler beforehand. In this chapter, we propose a milder restriction on the schedulers, and we define the notion of strong (probabilistic) information hiding under various notions of adversaries. Furthermore, we propose a method, based on the notion of automorphism, to verify that a system satisfies the property of strong information hiding, namely strong anonymity or non-interference, depending on the context. Through the chapter, we use the canonical example of the Dining Cryptographers to illustrate our ideas and techniques.

4.1 Introduction

The problem of information hiding consists in trying to prevent the adversary to infer confidential information from the observables. Instances of this issue are Anonymity and Information Flow. In both fields there is a growing interest in the quantitative aspects of the problem, see for instance [HO05, BP05, ZB05, CHM05a, CHM05b, Mal07, MC08, BCP08, CMS09, CPP08a, CPP08b, Smi09]. This is justified by the fact that often we have some a priori knowledge about the likelihood of the various secrets (which we can usually express in terms of a probability distribution), and by the fact that protocols often use randomized actions to obfuscate the link between secret and observable, like in the case of the anonymity protocols of DC Nets [Cha88], Crowds [RR98], Onion Routing [SGR97], and Freenet [CSWH00].

In a concurrent setting, like in the case of multi-agent systems, there is also another source of uncertainty, which derives from the fact that the various entities may interleave and interact in ways that are usually unpredictable, either because they depend on factors that are too complex to analyze, or because (in the case of specifications) they are implementation-dependent.

The formal analysis of systems which exhibit probabilistic and nondeterministic behavior usually involves the use of so-called *schedulers*, which are functions that, for each path, select only one possible (probabilistic) transition, thus delivering a purely probabilistic execution tree, where each event has a precise probability.

In the area of security, there is the problem that secret choices, like all choices, give rise to different paths. On the other hand, the decision of the scheduler may influence the observable behavior of the system. Therefore the security properties are usually violated if we admit as schedulers all possible functions of the paths: certain schedulers induce a dependence of the observables on the secrets, and protocols which would not leak secret information when running in "real" systems (where the scheduling devices cannot "see" the internal secrets of the components and therefore cannot depend on them), do leak secret information under this more permissive

notion of scheduler. This is a well known problem for which various solutions have already been proposed [CCK+06a, CCK+06b, CP10, CNP09]. We will come back to these in Section 4.7.

4.1.1 Contribution

We now list the main contribution of this chapter:

- We define a class of partial-information schedulers (which we call *admissible*), schedulers in this class are a restricted version of standard (full-information) schedulers. The restriction is rather flexible and has strong structural properties, thus facilitating the reasoning about security properties. In short, our systems consist of parallel components with certain restrictions on the secret choices and nondeterministic choices. The scheduler selects the next component (or components, in case of synchronization) for the subsequent step independently of the secret choices. We then formalize the notion of quantitative information flow, or degree of anonymity, using this restricted notion of scheduler.
- We propose alternative definitions to the property of strong anonymity defined in [BP05]. Our proposal differs from the original definition in two aspects: (1) the system should be strongly anonymous for all admissible schedulers instead of all schedulers (which is a very strong condition, never satisfied in practice), (2) we consider several variants of adversaries, namely (in increasing level of power): external adversaries, internal adversaries, and adversaries in collusion with the scheduler (in a Dolev-Yao fashion). Additionally, we use admissible schedulers to extend the notions of multiplicative and additive leakage (proposed in [Smi09] and [BCP09] respectively) to the case of a concurrent system.
- We propose a sufficient technique to prove probabilistic strong anonymity, and probabilistic noninterference, based on automorphisms. The idea is the following: in the purely nondeterministic setting, the strong anonymity of a system is often proved (or defined) as follows:

take two users A and B and a trace in which user A is 'the culprit'. Now find a trace that looks the same to the adversary, but in which user B is 'the culprit' [HO05, GHvRP05, MVdV04, HK07c]. This new trace is often most easily obtained by *switching the behavior of A and B*. Non-interference can be proved in the same way (where A and B are high information and the trace is the low information).

In this work, we make this technique explicit for anonymity in systems where probability and nondeterminism coexist, and we need to cope with the restrictions on the schedulers. We formalize the notion of *switching behaviors* by using automorphism (it is possible to switch the behavior of A and B if there exist an automorphism between them) and then show that the existence of an automorphism implies strong anonymity.

- We illustrate the problem with full-information schedulers in security, our solution providing admissible schedulers, and the application of our prove technique by means of the well known Dining Cryptographers anonymity protocol.

4.2 Preliminaries

In this section we gather preliminary notions and results related to probabilistic automata [SL95, Seg95], information theory [CT06], and information leakage [Smi09, BCP09].

4.2.1 Probabilistic automata

A function $\mu\colon Q \to [0,1]$ is a *discrete probability distribution* on a set Q if $\sum_{q\in Q} \mu(q) = 1$. The set of all discrete probability distributions on Q is denoted by $\mathcal{D}(Q)$.

A *probabilistic automaton* is a quadruple $M = (Q, \Sigma, \hat{q}, \theta)$ where Q is a countable set of *states*, Σ a finite set of *actions*, $\hat{q}$ the *initial* state, and θ a *transition function* $\theta : Q \to \mathcal{P}(\mathcal{D}(\Sigma \times Q))$. Here $\mathcal{P}(X)$ is the set of all subsets of X.

If $\theta(q) = \emptyset$, then q is a *terminal* state. We write $q{\to}\mu$ for $\mu \in \theta(q),\ q \in Q$. Moreover, we write $q \xrightarrow{a} r$ for $q, r \in Q$ whenever $q{\to}\mu$ and $\mu(a, r) > 0$. A *fully probabilistic automaton* is a probabilistic automaton satisfying $|\theta(q)| \leq 1$ for all states. In case $\theta(q) \neq \emptyset$ in a fully probabilistic automaton, we will overload notation and use $\theta(q)$ to denote the distribution outgoing from q. A *path* in a probabilistic automaton is a sequence $\sigma = q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots$ where $q_i \in Q$, $a_i \in \Sigma$ and $q_i \xrightarrow{a_{i+1}} q_{i+1}$. A path can be *finite* in which case it ends with a state. A path is *complete* if it is either infinite or finite ending in a terminal state. Given a path σ, $\mathsf{first}(\sigma)$ denotes its first state, and if σ is finite then $\mathsf{last}(\sigma)$ denotes its last state. A *cycle* is a path σ such that $\mathsf{last}(\sigma) = \mathsf{first}(\sigma)$. Let $\mathsf{Paths}_q(M)$ denote the set of all paths, $\mathsf{Paths}^\star_q(M)$ the set of all finite paths, and $\mathsf{CPaths}_q(M)$ the set of all complete paths of an automaton M, starting from the state q. We will omit q if $q = \hat{q}$. Paths are ordered by the prefix relation, which we denote by $\leq$. The *trace* of a path is the sequence of actions in $\Sigma^* \cup \Sigma^\infty$ obtained by removing the states, hence for the above path σ we have $\mathit{trace}(\sigma) = a_1 a_2 \ldots$. If $\Sigma' \subseteq \Sigma$, then $\mathit{trace}_{\Sigma'}(\sigma)$ is the projection of $\mathit{trace}(\sigma)$ on the elements of Σ'.

Let $M = (Q, \Sigma, \hat{q}, \theta)$ be a (fully) probabilistic automaton, $q \in Q$ a state, and let $\sigma \in \mathsf{Paths}^\star_q(M)$ be a finite path starting in q. The *cone* generated by σ is the set of complete paths $\langle\sigma\rangle = \{\sigma' \in \mathsf{CPaths}_q(M) \mid \sigma \leq \sigma'\}$. Given a fully probabilistic automaton $M = (Q, \Sigma, \hat{q}, \theta)$ and a state q, we can calculate the *probability value*, denoted by $\mathbb{P}_q(\sigma)$, of any finite path σ starting in q as follows: $\mathbb{P}_q(q) = 1$ and $\mathbb{P}_q(\sigma \xrightarrow{a} q') = \mathbb{P}_q(\sigma)\ \mu(a, q')$, where $\mathsf{last}(\sigma) \to \mu$.

Let $\Omega_q \triangleq \mathsf{CPaths}_q(M)$ be the sample space, and let $\mathcal{F}_q$ be the smallest σ-algebra generated by the cones. Then $\mathbb{P}_q$ induces a unique *probability measure* on $\mathcal{F}_q$ (which we will also denote by $\mathbb{P}_q$) such that $\mathbb{P}_q(\langle\sigma\rangle) = \mathbb{P}_q(\sigma)$ for every finite path σ starting in q. For $q = \hat{q}$ we write $\mathbb{P}$ instead of $\mathbb{P}_{\hat{q}}$.

A (full-information) scheduler for a probabilistic automaton M is a function $\zeta\colon \mathsf{Paths}^\star(M) \to (\mathcal{D}(\Sigma \times Q) \cup \{\bot\})$ such that for all finite paths σ, if $\theta(\mathsf{last}(\sigma)) \neq \emptyset$ then $\zeta(\sigma) \in \theta(\mathsf{last}(\sigma))$, and $\zeta(\sigma) = \bot$ otherwise. Hence, a scheduler ζ selects one of the available transitions in each state, and determines therefore a fully probabilistic automaton, obtained by pruning from M the alternatives that are not chosen by ζ. Note that a scheduler is

history dependent since it can take different decisions for the same state s according to the past evolution of the system.

4.2.2 Noisy Channels

This section briefly recalls the notion of noisy channels from Information Theory [CT06].

A *noisy channel* is a tuple $\mathcal{C} \triangleq (\mathcal{X}, \mathcal{Y}, P(\cdot|\cdot))$ where $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ is a finite set of *input values*, modeling the *secrets* of the channel, and $\mathcal{Y} = \{y_1, y_2, \ldots, y_m\}$ is a finite set of *output values* modeling the *observables* of the channel. For $x_i \in \mathcal{X}$ and $y_j \in \mathcal{Y}$, $P(y_j | x_i)$ is the conditional probability of obtaining the output y_j given that the input is x_i. These conditional probabilities constitute the so called *channel matrix*, where $P(y_j|x_i)$ is the element at the intersection of the i-th row and the j-th column. For any input distribution P_X on $\mathcal{X}$, P_X and the channel matrix determine a joint probability $P_\wedge$ on $\mathcal{X} \times \mathcal{Y}$, and the corresponding marginal probability P_Y on $\mathcal{Y}$ (and hence a random variable Y). P_X is also called *a priori distribution* and it is often denoted by π. The probability of the input given the output is called *a posteriori distribution*.

4.2.3 Information leakage

We recall here the definitions of *multiplicative leakage* proposed in [Smi09], and of *additive leakage* proposed in [BCP09][1]. We assume given a noisy channel $\mathcal{C} = (\mathcal{X}, \mathcal{Y}, P(\cdot|\cdot))$ and a random variable X on $\mathcal{X}$. The *a priori vulnerability* of the secrets in $\mathcal{X}$ is the probability of guessing the right secret, defined as $V(X) \triangleq \mathsf{max}_{x \in \mathcal{X}} P_X(x)$. The rationale behind this definition is that the adversary's best bet is on the secret with highest probability. The *a posteriori vulnerability* of the secrets in $\mathcal{X}$ is the probability of guessing the right secret, after the output has been observed, averaged over the probabilities of the observables. The formal definition is

[1]The notion proposed by Smith in [Smi09] was given in a (equivalent) logarithmic form, and called simply *leakage*. For uniformity sake we use here the terminology and formulation of [BCP09].

$V(X|\ Y) \triangleq \sum_{y\in\mathcal{Y}} P_Y(y)\,\max_{x\in\mathcal{X}} P(x\,|\,y)$. Again, this definition is based on the principle that the adversary will choose the secret with the highest a posteriori probability.

Note that, using Bayes theorem, we can write the a posteriori vulnerability in terms of the channel matrix and the a priori distribution, or in terms of the joint probability:

$$V(X|\ Y) = \sum_{y\in\mathcal{Y}} \max_{x\in\mathcal{X}} (P(y\,|\,x)P_X(x)) = \sum_{y\in\mathcal{Y}} \max_{x\in\mathcal{X}} P_\wedge(x,y).$$

The *multiplicative* leakage is $\mathcal{L}_\times(\mathcal{C}, P_X) \triangleq \frac{V(X|\,Y)}{V(X)}$ whereas the *additive* leakage is $\mathcal{L}_+(\mathcal{C}, P_X) \triangleq V(X|\,Y) - V(X)$.

4.2.4 Dining Cryptographers

This problem, described by Chaum in [Cha88], involves a situation in which three cryptographers are dining together. At the end of the dinner, each of them is secretly informed by a central agency (master) whether he should pay the bill, or not. So, either the master will pay, or one of the cryptographers will be asked to pay. The cryptographers (or some external observer) would like to find out whether the payer is one of them or the master. However, if the payer is one of them, they also wish to maintain anonymity over the identity of the payer.

A possible solution to this problem, described in [Cha88], is that each cryptographer tosses a coin, which is visible to himself and his neighbor to the left. Each cryptographer observes the two coins that he can see and announces *agree* or *disagree*. If a cryptographer is not paying, he will announce *agree* if the two sides are the same and *disagree* if they are not. The paying cryptographer will say the opposite. It can be proved that if the number of disagrees is even, then the master is paying; otherwise, one of the cryptographers is paying. Furthermore, in case one of the cryptographers is paying, neither an external observer nor the other two cryptographers can identify, from their individual information, who exactly is paying (provided that the coins are fair). The Dining Cryptographers (DC) will be a running example through the chapter.

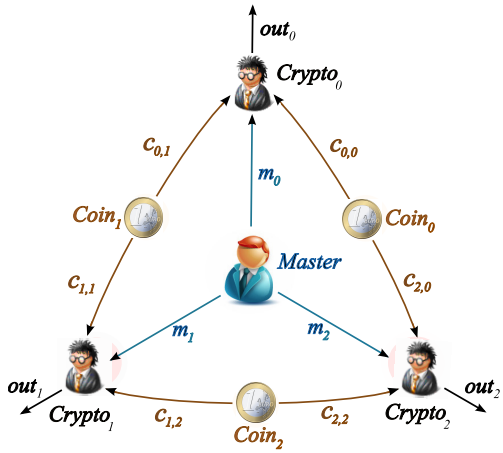


Figure 4.1: Chaum's system for the Dining Cryptographers ([Cha88])

4.3 Systems

In this section we describe the kind of systems we are dealing with. We start by introducing a variant of probabilistic automata, that we call *tagged probabilistic automata* (TPA). These systems are parallel compositions of purely probabilistic processes, that we call *components*. They are equipped with a unique identifier, that we call *tag*, or *label*, of the component. Note that, because of the restriction that the components are fully deterministic, nondeterminism is generated only from the interleaving of the parallel components. Furthermore, because of the uniqueness of the tags, each transition from a node is associated to a different tag / pair of two tags (one in case only one component makes a step, and two in case of a synchronization step among two components).

4.3.1 Tagged Probabilistic Automata

We now formalize the notion of TPA.

Definition 4.3.1. A *tagged probabilistic automaton* (TPA) is a tuple $(Q, L, \Sigma, \hat{q}, \theta)$, where

- Q is a set of *states*,
- L is a set of *tags*, or *labels*,
- Σ is a set of *actions*,
- $\hat{q} \in Q$ is the *initial state*,
- $\theta \colon Q \to \mathcal{P}(L \times D(\Sigma \times Q))$ is a *transition function.*

with the additional requirement that for every $q \in Q$ and every $\ell \in L$ there is at most one $\mu \in D(\Sigma \times Q)$ such that $(\ell, \mu) \in \theta(q)$.

A path for a TPA is a sequence $\sigma = q_0 \xrightarrow{l_1,a_1} q_1 \xrightarrow{l_2,a_2} q_2 \cdots$. In this way, the process with identifier l_i induces the system to move from q_{i-1} to q_i performing the action a_i, and it does so with probability $\mu_{l_i}(a_i, q_i)$, where μ_{l_i} is *the* distribution associated to the choice made by the component l_i. Finite paths and complete paths are defined in a similar manner.

In a TPA, the scheduler's choice is determined by the choice of the tag. We will use $enab(q)$ to denote the tags of the components that are enabled to make a transition. Namely,

$$enab(q) \triangleq \{\ell \in L \mid \exists\, \mu \in D(\Sigma \times Q) \,:\, (\ell, \mu) \in \theta(q)\} \tag{4.1}$$

We assume that the scheduler is forced to select a component among those which are enabled, i.e., that the execution does not stop unless all components are blocked (suspended or terminated). This is in line with the spirit of process algebra, and also with the tradition of Markov Decision Processes, but contrasts with that of the Probabilistic Automata of Lynch and Segala [SL95]. However, the results in this chapter do not depend on this assumption; we could as well allow schedulers which decide to terminate the execution even though there are transitions which are possible from the last state.

Definition 4.3.2. A *scheduler* for a TPA $M = (Q, L, \Sigma, \hat{q}, \theta)$ is a function $\zeta \colon \mathsf{Paths}^\star(M) \to (L \cup \{\bot\})$ such that for all finite paths σ, $\zeta(\sigma) \in enab(\mathsf{last}(\sigma))$ if $enab(\mathsf{last}(\sigma)) \neq \emptyset$ and $\zeta(\sigma) = \bot$ otherwise.

4.3.2 Components

To specify the components we use a sort of probabilistic version of CCS [Mil89, Mil99]. We assume a set of *secret actions* Σ_S with elements $s, s_1, s_2, \cdots$, and a disjoint set of *observable actions* Σ_O with elements $a, a_1, a_2, \cdots$. Furthermore we have *communication actions* of the form $c(x)$ (receive x on channel c, where x is a formal parameter), or $\bar{c}\langle v\rangle$ (send v on channel c, where v is a value on some domain V). Sometimes we need only to synchronize without transmitting any value, in which case we will use simply c and $\bar{c}$. We denote the set of channel names by C.

A component q is specified by the following grammar:

Components

q	::=	0	termination
	|	$a.q$	observable prefix
	|	$\sum_i p_i : q_i$	blind choice
	|	$\sum_i p_i : s_i.q_i$	secret choice
	|	$\mathit{if}\ x = v\ \mathit{then}\ q_1\ \mathit{else}\ q_2$	conditional
	|	A	process call

Observables

a	::=	$c \mid \bar{c}$	simple synchronization
	|	$c(x) \mid \bar{c}\langle v\rangle$	synchronization and communication

The p_i, in the blind and secret choices, represents the probability of the i-th branch and must satisfy $0 \leq p_i \leq 1$ and $\sum_i p_i = 1$. When no confusion arises, we use simply $+$ for a binary choice. The process call A is a simple process identifier. For each of them, we assume a corresponding unique process declaration of the form $A \stackrel{\text{def}}{=} q$. The idea is that, whenever A is executed, it triggers the execution of q. Note that q can contain A or another process identifier, which means that our language allows (mutual) recursion.

Note that each component contains only probabilistic and sequential constructs. In particular, there is no internal parallelism. Hence each component corresponds to a purely probabilistic automaton (apart from the input nondeterminism, which disappears in the definition of a system), as described by the operational semantics below. The main reason to dismiss the use of internal parallelism is verification: as mentioned in the Introduction we will present a proof technique for the different definitions of anonymity proposed in this work. This result would not be possible without such restriction on the components (see Example 4.6.4).

For an extension of this framework allowing the use of internal parallelism we refer to [AAPvR10]. There, the authors combine *global nondeterminism* (arising from the interleaving of the components) and *local nondeterminism* (arising from the internal parallelism of the components). The authors use such (extended) framework for a different purpose than ours, namely to define a notion of equivalence suitable for security analysis. No verification mechanisms are provided in [AAPvR10].

Components' semantics: The operational semantics consists of probabilistic transitions of the form $q \to \mu$ where $q \in Q$ is a process, and $\mu \in \mathcal{D}(\Sigma \times Q)$ is a distribution on actions and processes. They are specified by the following rules:

$$\text{PRF1} \quad \frac{v \in V}{c(x).q \to \delta(c(v), q[v/x])}$$

$$\text{PRF2} \quad \frac{}{a.q \to \delta(a, q)} \quad \text{if } a \neq c(x)$$

$$\text{INT} \quad \frac{}{\sum_i p_i : q_i \to \sum_i p_i \cdot \delta(\tau, q_i)}$$

$$\text{SECR} \quad \frac{}{\sum_i p_i : s_i.q_i \to \sum_i p_i \cdot \delta(s_i, q_i)}$$

$$\text{CND1}\quad \frac{}{\textit{if } v = v \textit{ then } q_1 \textit{ else } q_2 \to \delta(\tau, q_1)}$$

$$\text{CND2}\quad \frac{v \neq v'}{\textit{if } v = v' \textit{ then } q_1 \textit{ else } q_2 \to \delta(\tau, q_2)}$$

$$\text{CALL}\quad \frac{q \to \mu}{A \to \mu} \quad \text{if } A \triangleq q$$

$\sum_i p_i \cdot \mu_i$ is the distribution μ such that $\mu(x) = \sum_i p_i \mu_i(x)$. We use $\delta(x)$ to represent the delta of Dirac, which assigns probability 1 to x. The silent action, τ, is a special action different from all the observable and the secret actions. $q[v/x]$ stands for the process q in which any occurrence of x has been replaced by v. To shorten the notation, in the examples throughout the chapter, we omit writing explicit termination, i.e., we omit the symbol 0 at the end of a term.

4.3.3 Systems

A system consists of n processes (components) in parallel, restricted at the top-level on the set of channel names C:

$$(C)\ q_1 \parallel q_2 \parallel \cdots \parallel q_n.$$

The restriction on C enforces synchronization (and possibly communication) on the channel names belonging to C, in accordance with the CCS spirit. Since C is the set of all channels, all of them are forced to synchronize. This is to eliminate, at the level of systems, the nondeterminism generated by the rule for the receive prefix, PRF1.

Systems' semantics: The semantics of a system gives rise to a TPA, where the states are terms representing systems during their evolution. A transition now is of the form $q \xrightarrow{\ell} \mu$ where $\mu \in (\mathcal{D}(\Sigma \times Q))$ and $\ell \in L$ is either

the identifier of the component which makes the move, or a two-element set of identifiers representing the two partners of a synchronization. The following two rules provide the operational semantics rules in the case of interleaving and synchronisation/communication, respectively.

Interleaving If $a_j \notin C$

$$\frac{q_i \to \sum_j p_j \cdot \delta(a_j, q_{ij})}{(C)\ q_1 \parallel \cdots \parallel q_i \parallel \cdots \parallel q_n \xrightarrow{i} \sum_j p_j \cdot \delta(a_j, (C)\ q_1 \parallel \cdots \parallel q_{ij} \parallel \cdots \parallel q_n)}$$

where i indicates the tag of the component making the step.

Synchronization/Communication

$$\frac{q_i \to \delta(\bar{c}\langle v\rangle, q_i') \qquad q_j \to \delta(c(v), q_j')}{(C)\ q_1 \parallel \cdots \parallel q_i \parallel \cdots \parallel q_n \xrightarrow{\{i,j\}} \delta(\tau, (C)\ q_1 \parallel \cdots \parallel q_i' \parallel \cdots \parallel q_j' \parallel \cdots \parallel q_n)}$$

here $\{i, j\}$ is the tag indicating that the components making the step are i and j. For simplicity we write $\xrightarrow{i,j}$ instead of $\xrightarrow{\{i,j\}}$. The rule for synchronization without communication is similar, the only difference is that we do not have $\langle v\rangle$ and (v) in the actions. Note that c can only be an observable action (neither a secret nor τ), by the assumption that channel names can only be observable actions.

We note that both interleaving and synchronization rules generate nondeterminism. The only other source of nondeterminism is PRF1, the rule for a receive prefix $c(x)$. However the latter is not real nondeterminism: it is introduced in the semantics of the components but it disappears in the semantics of the systems, given that the channel c is restricted at the

top-level. In fact the restriction enforces communication, and when communication takes place, only the branch corresponding to the actual value v transmitted by the corresponding send action is maintained, all the others disappear.

Proposition 4.3.3. *The operational semantics of a system is a TPA with the following characteristics:*

(a) Every step $q \xrightarrow{\ell} \mu$ *is either*

a blind choice: $\mu = \sum_i p_i \cdot \delta(\tau, q_i)$, *or*

a secret choice: $\mu = \sum_i p_i \cdot \delta(s_i, q_i)$, *or*

a delta of Dirac: $\mu = \delta(\alpha, q')$ *with* $\alpha \in \Sigma_O$ *or* $\alpha = \tau$.

(b) If $q \xrightarrow{\ell} \mu$ *and* $q \xrightarrow{\ell} \mu'$ *then* $\mu = \mu'$.

Proof. For (a) , we have that the rules for the components and the rule for synchronization / communication can only produce blind choices, secret choices, or deltas of Dirac. Furthermore, because of the restriction on all channels, the transitions at the system level cannot contain communication actions. Finally, observe that the interleaving rule maintains these properties.

As for (b), we know that at the component level, the only source of nondeterminism is PRF1, the rule for a receive prefix $c(x)$. At the system level, this action is forced to synchronize with a corresponding send action, and, in a component, there can be only one such action available at a time. Hence the tag determines the value to be sent, which in turn determines the selection of exactly one branch in the receiving process. The only other sources of nondeterminism are the interleaving and the synchronization/communication rules, and they induce a different tag for each alternative. □

Example 4.3.1. We now present the components for the Dining Cryptographers using the introduced syntax. They correspond to Figure 4.1 and

to the automata depicted in Figure 4.3. As announced before, we omit the symbol 0 for explicit termination at the end of each term. The secret actions s_i represent the identity of the payer. The operators $\oplus, \ominus$ represent the sum modulo 2 and the difference modulo 2, respectively. The test $i == n$ returns 1 (true) if $i = n$, and 0 otherwise. The set of restricted channel names is $C = \{c_{0,0}, c_{0,1}, c_{1,1}, c_{1,2}, c_{2,0}, c_{2,2}, m_0, m_1, m_2\}$.

$$
\begin{array}{rcl}
\text{Master} & \triangleq & p : \overline{m}_0\langle 0\rangle \,.\, \overline{m}_1\langle 0\rangle \,.\, \overline{m}_2\langle 0\rangle + (1-p) : \sum_{i=0}^{2} p_i : s_i \,. \\
 & & \overline{m}_0\langle i == 0\rangle \,.\, \overline{m}_1\langle i == 1\rangle \,.\, \overline{m}_2\langle i == 2\rangle \\
\text{Crypt}_i & \triangleq & m_i(pay) \,.\, c_{i,i}(coin_1) \,.\, c_{i,i\oplus 1}(coin_2) \,.\, \overline{out}_i\langle pay \oplus coin_1 \oplus coin_2\rangle \\
\text{Coin}_i & \triangleq & 0.5 : \bar{c}_{i,i}\langle 0\rangle \,.\, \bar{c}_{i\ominus 1,i}\langle 0\rangle \;+\; 0.5 : \bar{c}_{i,i}\langle 1\rangle \,.\, \bar{c}_{i\ominus 1,i}\langle 1\rangle \\
\text{System} & \triangleq & (C)\ \text{Master} \parallel \prod_{i=0}^{2} \text{Crypt}_i \parallel \prod_{i=0}^{2} \text{Coin}_i
\end{array}
$$

Figure 4.2: Dining Cryptographers CCS

The operation $pay \oplus coin_1 \oplus coin_2$ in Figure 4.2 is syntactic sugar, it can be defined using the *if-then-else* operator. Note that, in this way, if a cryptographer is not paying (pay = 0), then he announces 0 if the two coins are the same (agree) and 1 if they are not (disagree).

4.4 Admissible Schedulers

We now introduce the class of admissible schedulers.

Standard (full-information) schedulers have access to all the information about the system and its components, and in particular the secret choices. Hence, such schedulers can leak secrets by making their decisions depend on the secret choice of the system. This is the case with the Dining Cryptographers protocol of Section 4.2.4: among all possible schedulers for the protocol, there are several that leak the identity of the payer. In fact the scheduler has the freedom to decide the order of the announcements of the cryptographers (interleaving), so a scheduler could choose to let the payer announce lastly. In this way, the attacker learns the identity of the payer simply by looking at the interleaving of the announcements.

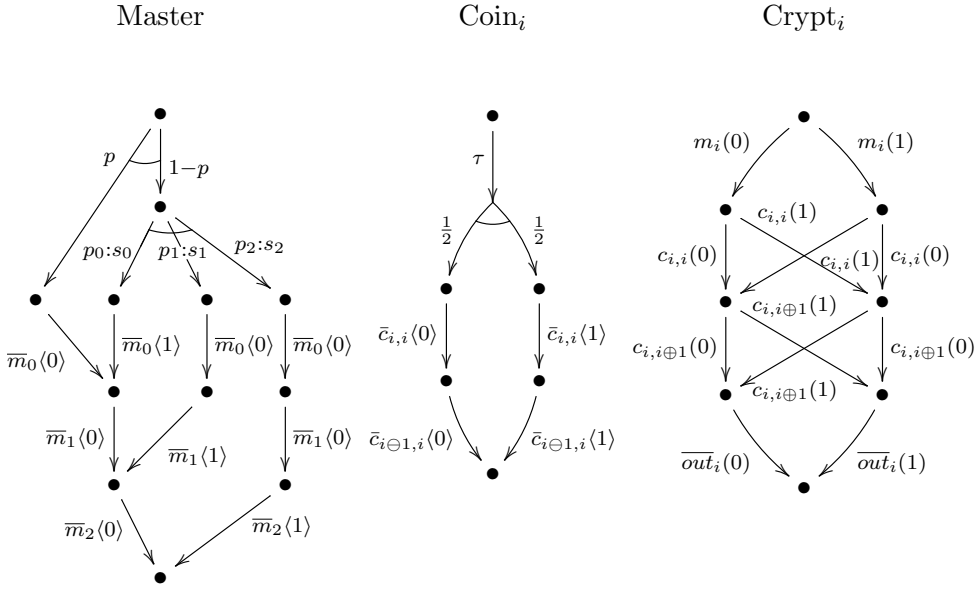


Figure 4.3: Dining Cryptographers Automata

4.4.1 The screens intuition

Let us first describe admissible schedulers informally. As mentioned in the introduction, admissible schedulers can base their decisions only on partial information about the evolution of the system, in particular admissible schedulers cannot base their decisions on information concerned with the internal behavior of components (such as secret choices).

We follow the subsequent intuition: admissible schedulers are entities that have access to a screen with buttons, where each button represents one (current) available option. At each point of the execution the scheduler decides the next step among the available options (by pressing the corresponding button). Then the output (if any) of the selected component becomes available to the scheduler and the screen is refreshed with the new available options (the ones corresponding to the system after mak-

ing the selected step). We impose that the scheduler can base its decisions only on such information, namely: the screens and outputs he has seen up to that point of the execution (and, of course, the decisions he has made).

Example 4.4.1. Consider $S \triangleq (\{c_1, c_2\})\ r \,\|\, q \,\|\, t$, where

$$r \triangleq 0.5 : s_1.\bar{c}_1.\bar{c}_2 + 0.5 : s_2.\bar{c}_1.\bar{c}_2,$$
$$q \triangleq c_1.(0.5 : a_1 + 0.5 : b_1), \quad t \triangleq c_2.(0.5 : a_2 + 0.5 : b_2).$$

Figure 4.4 shows the sequence of screens corresponding to a particular sequence of choices taken by the scheduler[2]. Interleaving and communication options are represented by yellow and red buttons, respectively. An arrow between two screens represents the transition from one to the other (produced by the scheduler pressing a button), additionally, the decision taken by the scheduler and corresponding outputs are depicted above each arrow.

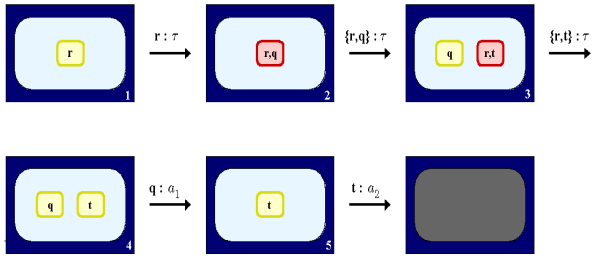


Figure 4.4: Screens intuition

Note that this system has exactly the same problem as the DC protocol: a full-information scheduler could reveal the secret by basing the interleaving order (q first or t first) on the secret choice of the component r. However, the same does not hold anymore for admissible schedulers (the scheduler cannot deduce the secret choice by just looking at the screens and outputs). This is also the case for the DC protocol, i.e., admissible schedulers cannot leak the secret of the protocol.

[2]The transitions from screens 4 and 5 represent 2 steps each (for simplicity we omit the τ-steps generated by blind choices)

4.4.2 The formalization

Before formally defining admissible schedulers we need to formalize the ingredients of the screens intuition. The buttons on the screen (available options) are the enabled options given by the function *enab* (see (4.1) in Section 4.3.1), the decision made by the scheduler is the tag of the selected enabled option, observable actions are obtained by sifting the secret actions to the schedulers by means of the following function:

$$sift(\alpha) \triangleq \begin{cases} \alpha & \text{if } \alpha \in \Sigma_O \cup \{\tau\}, \\ \tau & \text{if } \alpha \in \Sigma_S. \end{cases}$$

The partial information of a certain evolution of the system is given by the map t defined as follows.

Definition 4.4.1. Let $\hat{q} \xrightarrow{\ell_1,\alpha_1} \cdots \xrightarrow{\ell_n,\alpha_n} q_{n+1}$ be a finite path of the system, then we define t as:

$$t\left(\hat{q} \xrightarrow{\ell_1,\alpha_1} \cdots \xrightarrow{\ell_n,\alpha_n} q_{n+1}\right) \triangleq (enab(\hat{q}), \ell_1, sift(\alpha_1)) \cdots (enab(q_n), \ell_n, sift(\alpha_n)) \cdot enab(q_{n+1}).$$

Finally, we have all the ingredients needed to define admissible schedulers.

Definition 4.4.2 (Admissible schedulers)**.** A scheduler ζ is admissible if for all $\sigma, \sigma' \in \mathsf{Paths}^\star$

$$t(\sigma) = t(\sigma') \quad \text{implies} \quad \zeta(\sigma) = \zeta(\sigma').$$

In this way, admissible schedulers are forced to take the same decisions on paths that they cannot tell apart. Note that this is a restriction on the original definition of (full-information) schedulers where t is the identity map over finite paths (and consequently the scheduler is free to choose differently).

In the kind of systems we consider (the TPAs), the only source of nondeterminism are the interleaving and interactions of the parallel components.

Consequently, in a TPA the notion of scheduler is quite simple: its role, indeed, is to select, at each step, the component or pair of components which will perform the next transition. In addition, the TPA model allows us to express in a simple way the notion of admissibility: in fact the transitions available in the last state of σ are determined by the set of components enabled in the last state of σ, and $t(\sigma)$ gives (among other information) such set. Therefore $t(\sigma) = t(\sigma')$ implies that the last states of σ and σ' have the same possible transitions, hence it is possible to require that $\zeta(\sigma) = \zeta(\sigma')$ without being too restrictive or too permissive. In more general systems, where the sources of nondeterminism can be arbitrary, it is difficult to impose that the scheduler "does not depend on the secret choices", because different secret choices in general may give rise to states with different sets of transitions, and it is unclear whether such difference should be ruled out as "inadmissible", or should be considered as part of what a "real" scheduler can detect.

4.5 Information-hiding properties in presence of nondeterminism

In this section we revise the standard definition of information flow and anonymity in our framework of controlled nondeterminism.

We first consider the notion of adversary. We consider three possible notions of adversaries, increasingly more powerful.

4.5.1 Adversaries

External adversaries: Clearly, an adversary should be able, by definition, to see at least the observable actions. For an adversary external to the system S, it is natural to assume that these are also the only actions that he is supposed to see. Therefore, we define the observation domain, for an external adversary, as the set of the (finite) sequences of observable actions, namely:

$$\mathcal{O}_{\mathrm{e}} \triangleq \Sigma_O^*.$$

Correspondingly, we need a function $t_{\mathrm{e}} : \mathsf{Paths}^\star(S) \to \mathcal{O}_{\mathrm{e}}$ that extracts the observables from the executions:

$$t_{\mathrm{e}}\left(q_0 \xrightarrow{\ell_1,\alpha_1} \cdots \xrightarrow{\ell_n,\alpha_n} q_{n+1}\right) \triangleq \mathit{sieve}(\alpha_1)\cdots\mathit{sieve}(\alpha_n)$$

where

$$\mathit{sieve}(\alpha) \triangleq \begin{cases} \alpha & \text{if } \alpha \in \Sigma_O, \\ \epsilon & \text{if } \alpha \in \Sigma_S \cup \{\tau\}. \end{cases}$$

Internal adversaries: An internal adversary may be able to see, besides the observables, also the intearleaving and synchronizations of the various components, i.e. which component(s) are active, at each step of the execution. Hence it is natural to define the observation domain, for an internal adversary, as the sequence of pairs of observable action and tag (i.e. the identifier(s) of the active component(s)), namely:

$$\mathcal{O}_{\mathrm{i}} \triangleq (L \times (\Sigma_O \cup \{\tau\}))^*.$$

Correspondingly, we need a function $t_{\mathrm{i}} : \mathsf{Paths}^\star(S) \to \mathcal{O}_{\mathrm{i}}$ that extracts the observables from the executions:

$$t_{\mathrm{i}}\left(q_0 \xrightarrow{\ell_1,\alpha_1} \cdots \xrightarrow{\ell_n,\alpha_n} q_{n+1}\right) \triangleq (\ell_1, \mathit{sieve}(\alpha_1))\cdots(\ell_n, \mathit{sieve}(\alpha_n)).$$

Note that in this definition we could have equivalently used *sift* instead than *sieve*.

Adversaries in collusion with the scheduler: Finally, we consider the case in which the adversary is in collusion with the scheduler, or possibly the adversary *is* the scheduler. To illustrate the difference between this kind of adversaries and internal adversaries, consider the scheduler of an operating system. In such scenario an internal adversary is able to see which process has been scheduled to run next (process in the "running state") whereas an adversary in collusion with the scheduler can see as much as the scheduler, thus being able to see (in addition) which processes are in the "ready state" and which processes are in the "waiting / blocked" state. We will show later that such additional information does not help the adversary to leak

information (see Proposition 4.5.9). The observation domain of adversaries in collusion with the scheduler coincides with the one of the scheduler:

$$\mathcal{O}_{\mathrm{s}} \triangleq (\mathcal{P}(L) \times L \times (\Sigma_O \cup \{\tau\}))^*.$$

The corresponding function

$$t_{\mathrm{s}} : \mathsf{Paths}^\star(S) \to \mathcal{O}_{\mathrm{s}}$$

is defined as the one of the scheduler, i.e. $t_{\mathrm{s}} = t$.

4.5.2 Information leakage

In Information Flow and Anonymity there is a converging consensus for formalizing the notion of leakage as the difference or the ratio between the a priori uncertainty that the adversary has about the secret, and the a posteriori uncertainty, that is, the residual uncertainty of the adversary once it has seen the outcome of the computation. The uncertainty can be measured in different ways. One popular approach is the information-theoretic one, according to which the system is seen as a noisy channel between the secret inputs and the observable output, and uncertainty corresponds to the Shannon entropy of the system (see preliminaries – Section 4.2). In this approach, the leakage is represented by the so-called mutual information, which expresses the correlation between the input and the output.

The above approach, however, has been recently criticized by Smith [Smi09], who has argued that Shannon entropy is not suitable to represent the security threats in the typical case in which the adversary is interested in figuring out the secret in one-try attempt, and he has proposed to use Rényi's min entropy instead, or equivalently, the average probability of succeeding. This leads to interpret the uncertainty in terms of the notion of *vulnerability* defined in the preliminaries (Section 4.2). The corresponding notion of leakage, in the pure probabilistic case, has been investigated in [Smi09] (multiplicative case) and in [BCP09] (additive case).

Here we adopt the vulnerability-based approach to define the notion of leakage in our probabilistic and nondeterministic context. The Shannon-entropy-based approach could be extended to our context as well, because

in both cases we only need to specify how to determine the conditional probabilities which constitute the channel matrix, and the marginal probabilities that constitute the input and the output distribution.

We will denote by S the random variable associated to the set of secrets $\mathcal{S} = \Sigma_S$, and by O_{x} the random variables associated to the set of observables $\mathcal{O}_{\mathrm{x}}$, where $\mathrm{x} \in \{\mathrm{e}, \mathrm{i}, \mathrm{s}\}$. So, $\mathcal{O}_{\mathrm{x}}$ represents the observation domains for the various kinds of adversaries defined above.

As mentioned before, our results require some structural properties for the system: we assume that there is a single component in the system containing a secret choice and this component contains a single secret choice. This hypothesis is general enough to allow expressing protocols like the Dining Cryptographers, Crowds, voting protocols, etc., where the secret is chosen only once.

Assumption 4.5.1. A system contains exactly one component with a syntactic occurrence of a secret choice, and such a choice does not occur in the scope of a recursive call.

Note that the assumption implies that the secret choice appears exactly once in the operational semantics of the component. It would be possible to relax the assumption and allow more than one secret choice in a component, as long as there are no observable actions between the secret choices. For the sake of simplicity in this paper we impose the more restrictive requirement. As a consequence, we have that the operational semantics of systems satisfies the following property:

Proposition 4.5.2. *If $q \xrightarrow{\ell} \mu$ and $q' \xrightarrow{\ell'} \mu'$ are both secret choices, then $\ell = \ell'$ and there exist p_i's, q_i's and q_i''s such that:*

$$\mu = \sum_i p_i \cdot \delta(s_i, q_i) \quad \textit{and} \quad \mu' = \sum_i p_i \cdot \delta(s_i, q_i')$$

i.e., μ and μ' differ only for the continuation states.

Proof. Because of Assumption 4.5.1, there is only one component that can generate a secret choice, and it generates only one such choice. Due to

the different possible interleavings, this choice can appear as an outgoing transition in more than one state of the TPA, but the probabilities are always the same, because the interleaving rule does not change them. □

Given a system, each scheduler ζ determines a fully probabilistic automaton, and, as a consequence, the probabilities

$$\mathbb{P}_\zeta(s,o) \triangleq \mathbb{P}_\zeta\left(\bigcup\left\{\langle\sigma\rangle \mid \ \sigma \in \mathsf{Paths}^\star(S), t_{\mathrm{x}}(\sigma) = o, secr(\sigma) = s\ \right\}\right)$$

for each secret $s \in \mathcal{S}$ and observable $o \in \mathcal{O}_{\mathrm{x}}$, where $\mathrm{x} \in \{\mathrm{e},\mathrm{i},\mathrm{s}\}$. Here *secr* is the map from paths to their secret action. From these we can derive, in standard ways, the marginal probabilities $\mathbb{P}_\zeta(s)$, $\mathbb{P}_\zeta(o)$, and the conditional probabilities $\mathbb{P}_\zeta(o \mid s)$.

Every scheduler leads to a (generally different) noisy channel, whose matrix is determined by the conditional probabilities as follows:

Definition 4.5.3. Let $x \in \{\mathrm{e},\mathrm{i},\mathrm{s}\}$. Given a system and a scheduler ζ, the corresponding channel matrix $\mathcal{C}^{\mathrm{x}}_\zeta$ has rows indexed by $s \in \mathcal{S}$ and columns indexed by $o \in \mathcal{O}_{\mathrm{x}}$. The value in (s,o) is given by

$$\mathbb{P}_\zeta(o \mid s) \triangleq \frac{\mathbb{P}_\zeta(s,o)}{\mathbb{P}_\zeta(s)}.$$

Given a scheduler ζ, the multiplicative leakage can be defined as $\mathcal{L}_\times(\mathcal{C}^{\mathrm{x}}_\zeta, P_\zeta)$, while the additive leakage can be defined as $\mathcal{L}_+(\mathcal{C}^{\mathrm{x}}_\zeta, P_\zeta)$ where P_ζ is the a priori distribution on the set of secrets (see preliminaries, Section 4.2). However, we want a notion of leakage independent from the scheduler, and therefore it is natural to consider the worst case over all possible admissible schedulers.

Definition 4.5.4 (x-leakage)**.** Let $\mathrm{x} \in \{\mathrm{e},\mathrm{i},\mathrm{s}\}$. Given a system, the multiplicative leakage is defined as

$$\mathcal{ML}^{\mathrm{x}}_\times \triangleq \max_{\zeta \in Adm} \mathcal{L}_\times(\mathcal{C}^{\mathrm{x}}_\zeta, P_\zeta),$$

while the additive leakage is defined as

$$\mathcal{ML}^{\mathrm{x}}_{+} \triangleq \max_{\zeta \in Adm} \mathcal{L}_{+}(\mathcal{C}^{\mathrm{x}}_{\zeta}, P_{\zeta}),$$

where *Adm* is the class of admissible schedulers defined in the previous section.

We have that the classes of observables e, i, and s determine an increasing degree of leakage:

Proposition 4.5.5. *Given a system, for the multiplicative leakage we have*

1. *For every scheduler* ζ, $\mathcal{L}_{\times}(\mathcal{C}^{\mathrm{e}}_{\zeta}, P_{\zeta}) \leq \mathcal{L}_{\times}(\mathcal{C}^{\mathrm{i}}_{\zeta}, P_{\zeta}) \leq \mathcal{L}_{\times}(\mathcal{C}^{\mathrm{s}}_{\zeta}, P_{\zeta})$
2. $\mathcal{ML}^{\mathrm{e}}_{\times} \leq \mathcal{ML}^{\mathrm{i}}_{\times} \leq \mathcal{ML}^{\mathrm{s}}_{\times}$

Similarly for the additive leakage.

Proof.

1. The property follows immediately from the fact that the domain $\mathcal{O}_{\mathrm{e}}$ is an abstraction of $\mathcal{O}_{\mathrm{i}}$, and $\mathcal{O}_{\mathrm{i}}$ is an abstraction of $\mathcal{O}_{\mathrm{s}}$.
2. Immediate from previous point and from the definition of $\mathcal{ML}^{\mathrm{x}}_{\times}$ and $\mathcal{ML}^{\mathrm{x}}_{+}$. □

4.5.3 Strong anonymity (revised)

We consider now the situation in which the leakage is the minimum for all possible admissible schedules. In the purely probabilistic case, we know that the minimum possible multiplicative leakage is 1, and the minimum possible additive one is 0. We also know that this is the case for all possible input distributions if and only if the capacity of the channel matrix is 0, which corresponds to the case in which the rows of the matrix are all the same. This corresponds to the notion of strong probabilistic anonymity defined in [BP05]. In the framework of information flow, it would correspond to probabilistic non-interference. Still in [BP05], the authors considered also the extension of this notion in presence of nondeterminism, and required

the condition to hold under all possible schedulers. This is too strong in practice, as we have argued in the introduction: in most cases we can build a scheduler that leaks the secret by changing the interleaving order. We therefore tune this notion by requiring the condition to hold only under the admissible schedulers.

Definition 4.5.6 (x-strongly anonymous)**.** Let $x \in \{\mathrm{e}, \mathrm{i}, \mathrm{s}\}$. We say that a system is *x-strongly-anonymous* if for all admissible schedulers ζ we have

$$\mathbb{P}_\zeta\left(o \mid s_1\right) = \mathbb{P}_\zeta\left(o \mid s_2\right)$$

for all $s_1, s_2 \in \Sigma_S$, and $o \in \mathcal{O}_{\mathrm{x}}$.

The following corollary is an immediate consequence of Proposition 4.5.5.

Corollary 4.5.7.

1. If a system is s-strongly-anonymous, then it is also i-strongly-anonymous.
2. If a system is i-strongly-anonymous, then it is also e-strongly-anonymous.

The converse of point (2), in the previous corollary, does not hold, as shown by the following example:

Example 4.5.8. Consider the system $S \triangleq (\{c_1, c_2\})\ P \,||\, Q \,||\, T$ where

$$P \triangleq (0.5 : s_1 \,.\, \bar{c}_1) + (0.5 : s_2 \,.\, \bar{c}_2) \quad Q \triangleq c_1 \,.\, o \quad T \triangleq c_2 \,.\, o$$

It is easy to check that S is *e*-strongly anonymous but not *i*-strongly anonymous, showing that (as expected) internal adversaries can "distinguish more" than external adversaries.

On the contrary, for point (1) of Corollary 4.5.7, also the other direction holds:

Proposition 4.5.9. *A system is s-strongly-anonymous if and only if it is i-strongly-anonymous.*

Proof. Corollary 4.5.7 ensures the only-if part. For the if part, we proceed by contradiction. Assume that the system is i-strongly-anonymous but that $\mathbb{P}_\zeta(o \mid s_1) \neq \mathbb{P}_\zeta(o \mid s_2)$ for some admissible scheduler ζ and observable $o \in \mathcal{O}_\mathrm{s}$. Let $o = (enab(\hat{q}), \ell_1, sift(\alpha_1)) \cdots (enab(q_n), \ell_n, sift(\alpha_n))$ and let o' be the projection of o on $\mathcal{O}_\mathrm{i}$, i.e. $o' = (\ell_1, sift(\alpha_1)) \cdots (\ell_n, sift(\alpha_n))$. Since the system is i-strongly-anonymous, $\mathbb{P}_\zeta(o' \mid s_1) = \mathbb{P}_\zeta(o' \mid s_2)$, which means that the difference in probability with respect to o must be due to at least one of the sets of enabled processes. Let us consider the first set L in o which exhibits a difference in the probabilities, and let o'' be the prefix of o up to the tuple containing L. Since the probabilities are determined by the distributions on the probabilistic choices which occur in the individual components, the probability of each $\ell \in L$ to be available (given the trace o'') is independent of the other labels in L. At least one such ℓ must therefore have a different probability, given the trace o'', depending on whether the secret choice was s_1 or s_2. And, because of the assumption on L, we can replace the conditioning on trace o'' with the conditioning on the projection o''' of o'' on $\mathcal{O}_\mathrm{i}$. Consider now an admissible scheduler ζ' that acts like ζ up to o'', and then selects ℓ if and only if it is available. Since the probability that ℓ is not available depends on the choice of s_1 or s_2, we have $\mathbb{P}_\zeta(o''' \mid s_1) \neq \mathbb{P}_\zeta(o''' \mid s_2)$, which contradicts the hypothesis that the system is i-strongly-anonymous. □

Intuitively, this result means that an s-adversary can leak information if and only if an i-adversary can leak information or, in other words, s-adversaries are as powerful as i-adversaries (even when the former can observe more information).

4.6 Verifying strong anonymity: a proof technique based on automorphisms

As mentioned in the introduction, several problems involving restricted schedulers have been shown undecidable (including computing maximum / minimum probabilities for the case of standard model checking [GD07,

Gir09]). These results are discouraging in the aim to find algorithms for verifying strong anonymity/non-interference using our notion of admissible schedulers (and most definitions based on restricted schedulers). Despite the fact that the problem seems to be undecidable in general, in this section we present a sufficient (but not necessary) anonymity proof technique: we show that the existence of automorphisms between each pair of secrets implies strong anonymity. We conclude this section illustrating the applicability of our proof technique by means of the DC protocol, i.e., we prove that the protocol does not leak information by constructing automorphisms between pairs of cryptographers. It is worth mentioning that our proof technique is general enough to be used for the analysis of information leakage of a broad family of protocols, namely any protocol that can be modeled in our framework.

4.6.1 The proof technique

In practice proving anonymity often happens in the following way. Given a trace in which user A is the 'culprit', we construct an observationally equivalent trace in which user B is the 'culprit' [HO05, GHvRP05, MVdV04, HK07c]. This new trace is typically obtained by 'switching' the behavior of users A and B. We formalize this idea by using the notion of automorphism, cf. e.g. [Rut00].

Definition 4.6.1 (Automorphism)**.** Given a TPA $(Q, L, \Sigma, \hat{q}, \theta)$ we say that a bijection $f : Q \to Q$ is an *automorphism* if it satisfies $f(\hat{q}) = \hat{q}$ and

$$q \xrightarrow{\ell} \sum_i p_i \cdot \delta(\alpha_i, q_i) \iff f(q) \xrightarrow{\ell} \sum_i p_i \cdot \delta(\alpha_i, f(q_i)).$$

In order to prove anonymity it is sufficient to prove that the behaviors of any two 'culprits' can be exchanged without the adversary noticing. We will express this by means of the existence of automorphisms that exchange a given pair of secret s_i and s_j.

Before presenting the main theorem of this section we need to introduce one last definition. Let $S = (C)\ q_1 || \cdots || q_n$ be a system and M its corresponding TPA. We define M_τ as the automaton obtained after "hiding" all

the secret actions of M. The idea is to replace every occurrence of a secret s in M by the silent action τ. Note that this can be formalized by replacing the secret choice by a blind choice in the corresponding component q_i of the system S.

We now formalize the relation between automorphisms and strong anonymity. We will first show that the existence of automorphisms exchanging pairs of secrets implies s-strong anonymity (Theorem 4.6.2). Then, we will show that the converse does not hold, i.e. s-strongly-anonymous systems are not necessarily automorphic (Example 4.6.3).

Theorem 4.6.2. *Let S be a system satisfying Assumption 4.5.1 and M its tagged probabilistic automaton. If for every pair of secrets $s_i, s_j \in \Sigma_S$ there exists an automorphism f of M_τ such that for any state q we have*

$$q \xrightarrow{\ell,s_i}_M q' \implies f(q) \xrightarrow{\ell,s_j}_M f(q'), \tag{4.2}$$

then S is s-strongly-anonymous.

Proof. Assume that for every pair of secrets s_i, s_j we have an automorphism f satisfying the hypothesis of the theorem. We have to show that, for every admissible scheduler ζ we have:

$$\forall\, o \in \mathcal{O}_s :\ \mathbb{P}_\zeta\,(o \mid s_1) = \mathbb{P}_\zeta\,(o \mid s_2)\,.$$

We start by observing that for s_i, by Proposition 4.5.2, there exists a unique p_i such that, for all transitions $q \xrightarrow{l} \mu$, if μ is a (probabilistic) secret choice, then $\mu(s_i, -) = p_i$. Similarly for s_j, there exists a unique p_j such that $\mu(s_j, -) = p_j$ for all secret choices μ.

Let us now recall the definition of $\mathbb{P}_\zeta\,(o \mid s)$:

$$\mathbb{P}_\zeta\,(o \mid s) \triangleq \frac{\mathbb{P}_\zeta\,(o \wedge s)}{\mathbb{P}_\zeta\,(s)}$$

where $\mathbb{P}_\zeta\,(o \wedge s) \triangleq \mathbb{P}_\zeta\,(\{\pi \in \mathsf{CPaths} \mid t_s(\pi) = o \wedge \mathit{secr}(\pi) = s\})$ with $\mathit{secr}(\pi)$ being the (either empty or singleton) sequence of secret actions of π, and $\mathbb{P}_\zeta\,(s) \triangleq \mathbb{P}_\zeta\,(\{\pi \in \mathsf{CPaths} \mid \mathit{secr}(\pi) = s\})$.

Note that, since a secret appears at most once on a complete path, we have:

$$\begin{aligned}\mathbb{P}_\zeta(s_i) &= \mathbb{P}_\zeta\left(\{\pi \xrightarrow{\ell,s_i} \sigma \in \mathsf{CPaths} \mid \pi,\sigma\}\right)\\ &= \sum_{\pi \xrightarrow{\ell,s_i} q_i \in \mathsf{Paths}^\star} \mathbb{P}_\zeta\left(\langle \pi \xrightarrow{\ell,s_i} q_i\rangle\right) = \sum_{\substack{\mathsf{last}(\pi)\xrightarrow{\ell}\mu\\ \mu \text{ secret choice}}} \mathbb{P}_\zeta(\langle\pi\rangle)\cdot p_i\end{aligned}$$

and analogously

$$\begin{aligned}\mathbb{P}_\zeta(s_j) &= \mathbb{P}_\zeta\left(\{\pi \xrightarrow{\ell,s_j} \sigma \in \mathsf{CPaths} \mid \pi,\sigma\}\right)\\ &= \sum_{\pi \xrightarrow{\ell,s_j} q_j \in \mathsf{Paths}^\star} \mathbb{P}_\zeta\left(\langle \pi \xrightarrow{\ell,s_j} q_j\rangle\right) = \sum_{\substack{\mathsf{last}(\pi)\xrightarrow{\ell}\mu\\ \mu \text{ secret choice}}} \mathbb{P}_\zeta(\langle\pi\rangle)\cdot p_j\end{aligned}$$

Let us now consider $\mathbb{P}_\zeta(o \mid s_i)$ and $\mathbb{P}_\zeta(o \mid s_j)$. We have:

$$\begin{aligned}\mathbb{P}_\zeta(o \wedge s_i) &= \mathbb{P}_\zeta\left(\left\{\pi \xrightarrow{\ell,s_i} \sigma \in \mathsf{CPaths} \mid t_s(\pi \xrightarrow{\ell,s_i} \sigma) = o\right\}\right)\\ &= \sum_{\substack{\pi\\ \mathsf{last}(\pi)\xrightarrow{\ell}\mu\\ \mu \text{ secret choice}}} \mathbb{P}_\zeta(\langle\pi\rangle)\cdot p_i \cdot \sum_{\substack{\sigma\\ \pi\xrightarrow{\ell,s_i}\sigma\in\mathsf{Paths}^\star\\ t_s(\pi\xrightarrow{\ell,s_i}\sigma)=o\wedge\mathsf{last}(t_s(\sigma))\neq\tau}} \mathbb{P}_\zeta(\langle\sigma\rangle)\end{aligned}$$

again using that a secret appears at most once on a complete path. Moreover, note that we have overloaded the notation $\mathbb{P}_\zeta$ by using it for different measures when writing $\mathbb{P}_\zeta(\sigma)$, since σ need not start in the initial state $\hat{q}$. Analogously we have:

$$\begin{aligned}\mathbb{P}_\zeta(o \wedge s_j) &= \mathbb{P}_\zeta\left(\left\{\pi \xrightarrow{\ell,s_j} \sigma \in \mathsf{CPaths} \mid t_s(\pi \xrightarrow{\ell,s_j} \sigma) = o\right\}\right)\\ &= \sum_{\substack{\pi\\ \mathsf{last}(\pi)\xrightarrow{\ell}\mu\\ \mu \text{ secret choice}}} \mathbb{P}_\zeta(\langle\pi\rangle)\cdot p_j \cdot \sum_{\substack{\sigma\\ \pi\xrightarrow{\ell,s_j}\sigma\in\mathsf{Paths}^\star\\ t_s(\pi\xrightarrow{\ell,s_j}\sigma)=o\wedge\mathsf{last}(t_s(\sigma))\neq\tau}} \mathbb{P}_\zeta(\langle\sigma\rangle)\end{aligned}$$

Therefore, we derive

$$\mathbb{P}_\zeta\left(o \mid s_i\right) = \frac{\displaystyle\sum_{\substack{\pi \\ \mathsf{last}(\pi)\xrightarrow{\ell}\mu \\ \mu \text{ secret choice}}} \sum_{\substack{\sigma \\ \pi \xrightarrow{\ell,s_i} \sigma \in \mathsf{Paths}^\star \\ t_s(\pi \xrightarrow{\ell,s_i} \sigma)=o \wedge \mathsf{last}(t_s(\sigma))\neq\tau}} \mathbb{P}_\zeta\left(\langle\pi\rangle\right) \cdot \mathbb{P}_\zeta\left(\langle\sigma\rangle\right)}{\displaystyle\sum_{\substack{\mathsf{last}(\pi)\xrightarrow{\ell}\mu \\ \mu \text{ secret choice}}} \mathbb{P}_\zeta\left(\langle\pi\rangle\right)} \tag{4.3}$$

$$\mathbb{P}_\zeta\left(o \mid s_j\right) = \frac{\displaystyle\sum_{\substack{\pi \\ \mathsf{last}(\pi)\xrightarrow{\ell}\mu \\ \mu \text{ secret choice}}} \sum_{\substack{\sigma \\ \pi \xrightarrow{\ell,s_j} \sigma \in \mathsf{Paths}^\star \\ t_s(\pi \xrightarrow{\ell,s_j} \sigma)=o \wedge \mathsf{last}(t_s(\sigma))\neq\tau}} \mathbb{P}_\zeta\left(\langle\pi\rangle\right) \cdot \mathbb{P}_\zeta\left(\langle\sigma\rangle\right)}{\displaystyle\sum_{\substack{\mathsf{last}(\pi)\xrightarrow{\ell}\mu \\ \mu \text{ secret choice}}} \mathbb{P}_\zeta\left(\langle\pi\rangle\right)} \tag{4.4}$$

Observe that the denominators of both formulae (4.3) and (4.4) are the same. Also note that, since f is an automorphism, for every path π, $f(\pi)$ obtained by replacing each state in π with its image under f is also a path. Moreover, since f satisfies (4.2), for every path $\pi \xrightarrow{\ell,s_i} \sigma$ we have that $f(\pi) \xrightarrow{\ell,s_j} f(\sigma)$ is also a path. Furthermore f induces a bijection between the sets

$$\begin{aligned}\{(\pi,\sigma) \mid\ & \mathsf{last}(\pi) \xrightarrow{\ell'} \mu \text{ s.t. } \mu \text{ secret choice}, \pi \xrightarrow{\ell,s_i} \sigma \in \mathsf{Paths}^\star \\ & t_s(\pi \xrightarrow{\ell,s_i} \sigma) = o, \mathsf{last}(t_e(\sigma)) \neq \tau \quad \}, \text{ and}\end{aligned}$$

$$\begin{aligned}\{(\pi,\sigma) \mid\ & \mathsf{last}(\pi) \xrightarrow{\ell'} \mu \text{ s.t. } \mu \text{ secret choice}, \pi \xrightarrow{\ell,s_j} \sigma \in \mathsf{Paths}^\star \\ & t_s(\pi \xrightarrow{\ell,s_j} \sigma) = o, \mathsf{last}(t_e(\sigma)) \neq \tau \quad \}\end{aligned}$$

given by $(\pi,\sigma) \leftrightarrow (f(\pi), f(\sigma))$.

Finally, since ζ is admissible, $t_s(\pi) = t_s(f(\pi))$, and f is an automorphism, it is easy to prove by induction that $\mathbb{P}_\zeta\left(\langle\pi\rangle\right) = \mathbb{P}_\zeta\left(\langle f(\pi)\rangle\right)$. Similarly, $\mathbb{P}_\zeta\left(\langle\sigma\rangle\right) = \mathbb{P}_\zeta\left(\langle f(\sigma)\rangle\right)$. Hence the numerators of (4.3) and (4.4) coincide which concludes the proof. □

Note that, since s-strong anonymity implies i-strong anonymity and e-strong anonymity, the existence of such an automorphism implies all the notions of strong anonymity presented in this work. We now proceed to show that the converse does not hold, i.e. strongly anonymous systems are not necessarily automorphic.

Example 4.6.3. Consider the following (single component) system

$$\begin{array}{c} 0.5 : s_1.(0.5 : (p : a + (1-p) : b) + 0.5 : ((1-p) : a + p : b)) \\ + \\ 0.5 : s_2.(0.5 : (q : a + (1-q) : b) + 0.5 : ((1-q) : a + q : b)) \end{array}$$

It is easy to see that such system is s-strongly-anonymous, however if $p \neq q$ and $p \neq 1 - q$ there does not exist an automorphism for the pair of secrets (s_1, s_2).

The following example demonstrates that our proof technique does not carry over to systems whose components admit internal parallelism.

Example 4.6.4. Consider $S \triangleq (\{c_1, c_2\})\ r \,\|\, q \,\|\, t$, where

$$r \triangleq 0.5 : s_1.\overline{c}_1 + 0.5 : s_2.\overline{c}_2, \qquad q \triangleq c_1.(a \,|\, b), \qquad t \triangleq c_2.(a \,|\, b).$$

where $q_1|q_2$ represents the parallel composition of q_1 and q_2. It is easy to show that there exists an automorphism for s_1 and s_2. However, admissible schedulers are able to leak such secrets. This is due to the fact that component r synchronizes with q and t on different channels, thus a scheduler of S is not restricted to select the same transitions on the branches associated to s_1 and s_2 (remember that schedulers can observe synchronization).

We now show that the definition of x-strong-anonymity is independent of the particular distribution over secrets, i.e., if a system is x-strongly-anonymous for a particular distribution over secrets, then it is x-strongly-anonymous for all distributions over secrets. This result is useful because it allows us to prove systems to be strongly anonymous even when their distribution over secrets is not known.

Theorem 4.6.5. *Consider a system* $S = (C)\ q_1 \parallel \cdots \parallel q_i \parallel \cdots \parallel q_n$. *Let* q_i *be the component which contains the secret choice, and assume that it is of the form* $\sum_j p_j : s_j \,.\, q_j$. *Consider now the system* $S' = (C)\ q_1 \parallel \cdots \parallel q_i' \parallel \cdots \parallel q_n$, *where* q_i' *is identical to* q_i *except for the secret choice, which is replaced by* $\sum_j p_j' : s_j \,.\, q_j$. *Then we have that:*

1. *For every* s_i, s_j *there is an automorphism on* S *satisfying the assumption of Theorem 4.6.2 if and only if the same holds for* S'.
2. S *is* x*-strongly-anonymous if and only if* S' *is* x*-strongly-anonymous.*

Note: 1) does not imply 2), because in principle neither S not S' may have the automorphism, and still one of the two could be strongly anonymous.

Proof. We note that the PAs generated by S and S' coincide except for the probability distribution on the secret choices. Since the definition of automorphism and the assumption of Theorem 4.6.2 do not depend on these probability distributions, (1) is immediate. As for (2), we observe that x-strong anonymity only depends on the conditional probabilities $\mathbb{P}_\zeta(o \mid s)$. By looking at the proof of Theorem 4.6.2, we can see that in the computation of $\mathbb{P}_\zeta(o \mid s)$ the probabilities on the secret choices (i.e. the p_j's) are eliminated. Namely $\mathbb{P}_\zeta(o \mid s)$ does not depend on the p_j's, which means that the value of the p_j's has no influence on whether the system is x-strong anonymous or not. □

4.6.2 An Application: Dining Cryptographers

Now we show how to apply the proof technique presented in this section to the Dining Cryptographers protocol. Concretely, we show that there exists an automorphism f exchanging the behavior of the Crypt_0 and Crypt_1; by symmetry, the same holds for the other two combinations.

Consider the automorphisms of Master and Coin_1 indicated in Figure 4.5. The states that are not explicitly mapped (by a dotted arrow) are mapped to themselves.

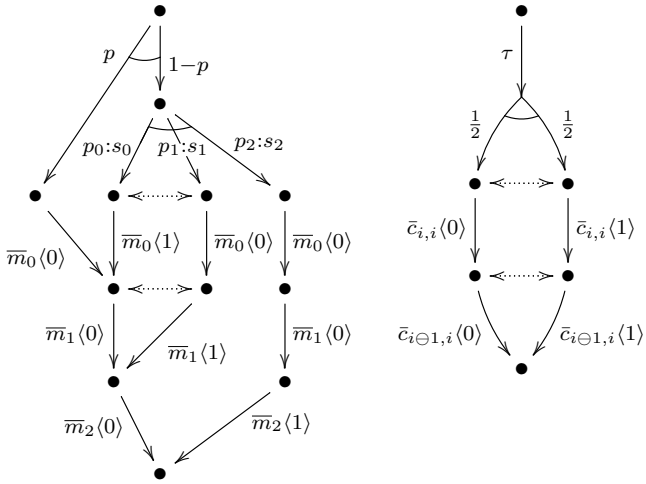


Figure 4.5: Automorphism between Crypt_0 and Crypt_1

Also consider the identity automorphism on Crypt_i (for $i = 0, 1, 2$) and on Coin_i (for $i = 0, 2$). It is easy to check that the product of these seven automorphisms is an automorphism for Crypt_0 and Crypt_1.

4.7 Related Work

The problem of the full-information scheduler has already been extensively investigated in literature. The works [CCK+06a] and [CCK+06b] consider probabilistic automata and introduce a restriction on the scheduler to the purpose of making them suitable to applications in security. Their approach is based on dividing the actions of each component of the system in equivalence classes (*tasks*). The order of execution of different tasks is decided in advance by a so-called *task scheduler*. The remaining nondeterminism within a task is resolved by a second scheduler, which models the standard *adversarial scheduler* of the cryptographic community. This second entity has limited knowledge about the other components: it sees only the information that they communicate during execution. Their notion of task scheduler is similar to our notion of admissible scheduler, but more restricted since the strategy of the task scheduler is decided entirely before

the execution of the system.

Another work along these lines is [dAHJ01], which uses partitions on the state-space to obtain partial-information schedulers. However that work considers a synchronous parallel composition, so the setting is rather different from ours.

The works in [CP10, CNP09] are similar to ours in spirit, but in a sense *dual* from a technical point of view. Instead of defining a restriction on the class of schedulers, they provide a way to specify that a choice is transparent to the scheduler. They achieve this by introducing labels in process terms, used to represent both the states of the execution tree and the next action or step to be scheduled. They make two states indistinguishable to schedulers, and hence the choice between them private, by associating to them the same label. Furthermore, their "equivalence classes" (schedulable actions with the same label) can change dynamically, because the same action can be associated to different labels during the execution.

In [AAPvR10] we have extended the framework presented in this work (by allowing internal nondeterminism and adding a second type of scheduler to resolve it) with the aim of investigating angelic vs demonic nondeterminism in equivalence-based properties.

The fact that full-information schedulers are unrealistic has also been observed in fields other than security. With the aim to cope with general properties (not only those concerning security), first attempts used restricted schedulers in order to obtain rules for compositional reasoning [dAHJ01]. The justification for those restricted schedulers is the same as for ours, namely, that not all information is available to all entities in the system. Later on, it was shown that model checking is undecidable in its general form for the kind of restricted schedulers presented in [dAHJ01]. See [GD07] and, more recently, [Gir09].

Finally, to the best of our knowledge, this is the first work using automorphisms as a sound proof technique (in our case to prove strong anonymity and non-interference). The closest line of work we are aware of is in the field of model checking. There, isomorphisms can be used to identify symmetries in the system, and such symmetries can then be exploited to alleviate the state space explosion (see for instance [KNP06]).

Chapter 5

Significant Diagnostic Counterexample Generation

In this chapter, we present a novel technique for counterexample generation in probabilistic model checking of Markov Chains and Markov Decision Processes. (Finite) paths in counterexamples are grouped together in witnesses that are likely to provide similar debugging information to the user. We list five properties that witnesses should satisfy in order to be useful as debugging aid: similarity, accuracy, originality, significance, and finiteness. Our witnesses contain paths that behave similarly outside strongly connected components. We then proceed to show how to compute these witnesses by reducing the problem of generating counterexamples for general properties over Markov Decision Processes, in several steps, to the easy problem of generating counterexamples for reachability properties over acyclic Markov Chains.

5.1 Introduction

Model checking is an automated technique that, given a finite-state model of a system and a property stated in an appropriate logical formalism, systematically checks the validity of this property. Model checking is a general approach and is applied in areas like hardware verification and software engineering.

Nowadays, the interaction geometry of distributed systems and network protocols calls for probabilistic, or more generally, quantitative estimates of, e.g., performance and cost measures. Randomized algorithms are increasingly utilized to achieve high performance at the cost of obtaining correct answers only with high probability. For all this, there is a wide range of models and applications in computer science requiring quantitative analysis. Probabilistic model checking allows to check whether or not a probabilistic property is satisfied in a given model, e.g., "Is every message sent successfully received with probability greater or equal than 0.99?".

A major strength of model checking is the possibility of generating diagnostic information in case the property is violated. This diagnostic information is provided through a *counterexample* showing an execution of the model that invalidates the property under verification. Besides the immediate feedback in model checking, counterexamples are also used in abstraction-refinement techniques [CGJ+00], and provide the foundations for schedule derivation (see, e.g., [BLR05, Feh02]).

Although counterexample generation was studied from the very beginning in most model checking techniques, this has not been the case for probabilistic model checking. Only recently [AHL05, And06, AL06, HK07a, HK07b, AL09] attention was drawn to this subject, fifteen years after the first studies on probabilistic model checking. Contrarily to other model checking techniques, counterexamples in this setting are *not* given by a single execution path. Instead, they are a *set of executions* of the system satisfying a certain undesired property whose probability mass is higher than a given bound. Since counterexamples are used as a diagnostic tool, previous works on counterexamples have presented them as a set of *finite* paths with probability large enough. We refer to these sets as *represen-*

tative counterexamples. Elements of representative counterexamples with high probability have been considered the most informative since they contribute mostly to the property refutation.

A challenge in counterexample generation for probabilistic model checking is that (1) representative counterexamples are very large (often infinite), (2) many of its elements have very low probability (which implies that they are very distant from the counterexample), and (3) that elements can be extremely similar to each other (consequently providing similar diagnostic information). Even worse, (4) sometimes the finite paths with highest probability do not indicate the most likely violation of the property under consideration.

For example, look at the Markov Chain $\mathcal{D}$ in Figure 5.1. The property $\mathcal{D} \models_{\leq 0.5} \Diamond\psi$ stating that execution reaches a state satisfying ψ (i.e., reaches s_3 or s_4) with probability lower or equal than 0.5 is violated (since the probability of reaching a state satisfying ψ is 1). The left hand side of table in Figure 5.2 lists finite paths reaching ψ ranked according to their probability. Note that finite paths with highest probability take the left branch of the system, whereas the right branch in itself has higher probability, illustrating Problem 4. To adjust the model so that it does satisfy the property (bug fixing), it is not sufficient to modify the left hand side of the system alone; no matter how one changes the left hand side, the probability of reaching ψ remains at least 0.6. Furthermore, the first six finite paths provide similar diagnostic information: they just make extra loops in s_1. This is an example of Problem 3. Additionally, the probability of every single finite path is far below the bound 0.5, making it unclear if a particular path is important; see Problem 2 above. Finally, the (unique) counterexample for the property $\mathcal{D} \models_{<1} \Diamond\psi$ consists of infinitely many finite paths (namely all finite paths of $\mathcal{D}$); see Problem 1. To overcome these problems, we partition a representative counterexample into sets of finite paths that follow a similar pattern. We call these sets *witnesses*. To ensure that witnesses provide valuable diagnostic information, we desire that the set of witnesses that form a counterexample satisfies several properties: two different witnesses should provide different diagnostic information (solving Problem 3) and elements of a single witness should provide similar diagnos-

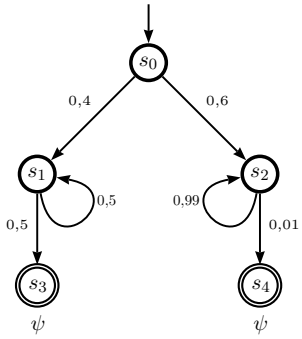


Figure 5.1: Markov Chain

	Single paths		Witnesses	
Rank	F. Path	Prob	Witness	Mass
1	$s_0(s_1)^1s_3$	0.2	$[s_0s_2s_4]$	0.6
2	$s_0(s_1)^2s_3$	0.1	$[s_0s_1s_3]$	0.4
3	$s_0(s_1)^3s_3$	0.05		
4	$s_0(s_1)^4s_3$	0.025		
5	$s_0(s_1)^5s_3$	0.0125		
6	$s_0(s_1)^6s_3$	0.00625		
7	$s_0(s_2)^1s_4$	0.006		
8	$s_0(s_2)^2s_4$	0.0059		
9	$s_0(s_2)^3s_4$	0.0058		
⋮	⋮	⋮		

Figure 5.2: Comparison Table

tic information, as a consequence witnesses have a high probability mass (solving Problems 2 and 4), and the number of witnesses of a representative counterexample should be finite (solving Problem 1).

In our setting, witnesses consist of paths that behave the same outside strongly connected components. In the example of Figure 5.1, there are two witnesses: the set of all finite paths going right, represented by $[s_0s_2s_4]$ whose probability (mass) is 0.6, and the set of all finite paths going left, represented by $[s_0s_1s_3]$ with probability (mass) 0.4.

In this chapter, we show how to obtain such sets of witnesses for bounded probabilistic LTL properties on Markov Decision Processes (MDP). In fact, we first show how to reduce this problem to finding witnesses for upper bounded probabilistic reachability properties on discrete time Markov Chains (MCs). The major technical matters lie on this last problem to which most of the chapter is devoted.

In a nutshell, the process to find witnesses for the violation of $\mathcal{D} \models_{\leq p} \Diamond\psi$, with $\mathcal{D}$ being an MC, is as follows. We first eliminate from the original MC all the "uninteresting" parts. This proceeds as the first steps of the model checking process: make absorbing all states satisfying ψ, and all states that cannot reach ψ, obtaining a new MC $\mathcal{D}_\psi$. Next, we reduce

this last MC to an acyclic MC $\mathsf{Ac}(\mathcal{D}_\psi)$ in which all strongly connected components have been conveniently abstracted with a single probabilistic transition. The original and the acyclic MCs are related by a mapping that, to each finite path in $\mathsf{Ac}(\mathcal{D}_\psi)$ (that we call *rail*), assigns a set of finite paths behaving similarly in $\mathcal{D}$ (that we call *torrent*). This map preserves the probability of reaching ψ and hence relates counterexamples in $\mathsf{Ac}(\mathcal{D}_\psi)$ to counterexamples in $\mathcal{D}$. Finally, counterexamples in $\mathsf{Ac}(\mathcal{D}_\psi)$ are computed by reducing the problem to a k shortest path problem, as in [HK07a]. Because $\mathsf{Ac}(\mathcal{D}_\psi)$ is acyclic, the complexity is lower than the corresponding problem in [HK07a].

It is worth mentioning that our technique can also be applied to pCTL formulas without nested path quantifiers.

Looking ahead, Section 5.2 presents the necessary background on Markov Chains (MC), Markov Decision Processes (MDP), and Linear Temporal Logic (LTL). Section 5.3 presents the definition of counterexamples and discusses the reduction from general LTL formulas to upper bounded probabilistic reachability properties, and the extraction of the maximizing MC in an MDP. Section 5.4 discusses desired properties of counterexamples. In Sections 5.5 and 5.6 we introduce the fundamentals on rails and torrents, the reduction of the original MC to the acyclic one, and our notion of significant diagnostic counterexamples. Section 5.7 then presents the techniques to actually compute counterexamples. Finally, in Section 5.8 we discuss related work..

5.2 Preliminaries

We now recall the notions of Markov Decision Processes, Markov Chains, and Linear Temporal Logic.

5.2.1 Markov Decision Processes

Markov Decision Processes (MDPs) constitute a formalism that combines nondeterministic and probabilistic choices. They are an important model in corporate finance, supply chain optimization, system verification and

optimization. There are many slightly different variants of this formalism such as action-labeled MDPs [Bel57, FV97], probabilistic automata [SL95, SdV04]; we work with the state-labeled MDPs from [BdA95].

Definition 5.2.1. Let S be a finite set. A *probability distribution* on S is a function $p\colon S \to [0,1]$ such that $\sum_{s\in S} p(s) = 1$. We denote the set of all probability distributions on S by $\mathsf{Distr}(S)$. Additionally, we define the *Dirac distribution* on an element $s \in S$ as 1_s, i.e., $1_s(s) = 1$ and $1_s(t) = 0$ for all $t \in S \setminus \{s\}$.

Definition 5.2.2. A *Markov Decision Process* (MDP) is a quadruple $\mathcal{M} = (S, s_0, L, \tau)$, where

- S is the finite state space;
- $s_0 \in S$ is the initial state;
- L is a labeling function that associates to each state $s \in S$ a set $L(s)$ of propositional variables that are *valid* in s;
- $\tau\colon S \to \wp(\mathsf{Distr}(S))$ is a function that associates to each $s \in S$ a non-empty and finite subset of $\mathsf{Distr}(S)$ of probability distributions.

Definition 5.2.3. Let $\mathcal{M} = (S, s_0, \tau, L)$ be an MDP. We define a *successor* relation $\delta \subseteq S \times S$ by $\delta \triangleq \{(s,t) | \exists\, \pi \in \tau(s)\,.\,\pi(t) > 0\}$ and for each state $s \in S$ we define the sets

$$\mathsf{Paths}(\mathcal{M}, s) \triangleq \{t_0 t_1 t_2 \ldots \in S^\omega | t_0 = s \wedge \forall n \in \mathbb{N}\,.\,\delta(t_n, t_{n+1})\} \text{ and}$$
$$\mathsf{Paths}^\star(\mathcal{M}, s) \triangleq \{t_0 t_1 \ldots t_n \in S^\star | t_0 = s \wedge \forall\, 0 \leq i < n\,.\,\delta(t_n, t_{n+1})\}$$

of paths of $\mathcal{D}$ and finite paths of $\mathcal{D}$ respectively beginning at s. We usually omit $\mathcal{M}$ from the notation; we also abbreviate $\mathsf{Paths}(\mathcal{M}, s_0)$ as $\mathsf{Paths}(\mathcal{M})$ and $\mathsf{Paths}^\star(\mathcal{M}, s_0)$ as $\mathsf{Paths}^\star(\mathcal{M})$. For $\omega \in \mathsf{Paths}(s)$, we write the $(n+1)$-st state of ω as ω_n. As usual, we let $\mathcal{B}_s \subseteq \wp(\mathsf{Paths}(s))$ be the Borel σ-algebra on the cones $\langle t_0 \ldots t_n \rangle \triangleq \{\omega \in \mathsf{Paths}(s) | \omega_0 = t_0 \wedge \ldots \wedge \omega_n = t_n\}$. Additionally, for a set of finite paths $\Lambda \subseteq \mathsf{Paths}^\star(s)$, we define $\langle \Lambda \rangle \triangleq \bigcup_{\sigma\in\Lambda} \langle \sigma \rangle$.

Figure 5.3 shows an MDP. Absorbing states (i.e., states s with $\tau(s) = \{1_s\}$) are represented by double lines. This MDP features a single nondeterministic decision, to be made in state s_0, namely π_1 and π_2.

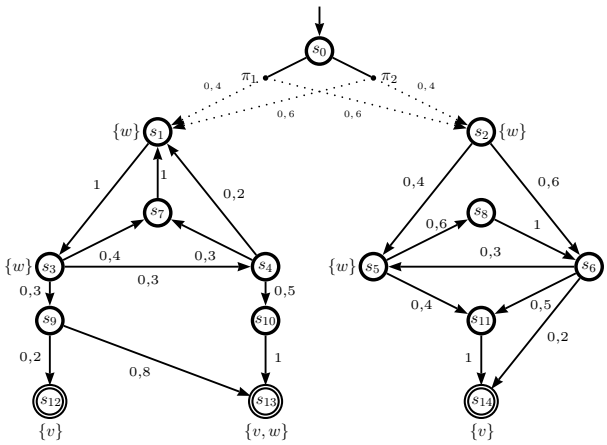


Figure 5.3: Markov Decision Process

Definition 5.2.4. Let $\mathcal{M} = (S, s_0, \tau, L)$ be an MDP, $s \in S$ and $\mathcal{A} \subseteq S$. We define the sets of paths and finite paths reaching $\mathcal{A}$ from s as

$$\mathsf{Reach}(\mathcal{M}, s, \mathcal{A}) \triangleq \{\omega \in \mathsf{Paths}(\mathcal{M}, s) \mid \exists_{i \geq 0}.\omega_i \in \mathcal{A}\} \text{ and}$$
$$\mathsf{Reach}^\star(\mathcal{M}, s, \mathcal{A}) \triangleq \{\sigma \in \mathsf{Paths}^\star(\mathcal{M}, s) \mid \mathsf{last}(\sigma) \in \mathcal{A} \wedge \forall_{i \leq |\sigma|-1}.\sigma_i \notin \mathcal{A}\}$$

respectively. Note that $\mathsf{Reach}^\star(\mathcal{M}, s, \mathcal{A})$ consists of those finite paths σ starting on s reaching $\mathcal{A}$ exactly once, at the end of the execution. It is easy to check that these sets are *prefix free*, i.e. contain finite paths such that none of them is a prefix of another one.

5.2.2 Schedulers

Schedulers (also called strategies, adversaries, or policies) resolve the non-deterministic choices in an MDP [PZ93, Var85, BdA95].

Definition 5.2.5. Let $\mathcal{M} = (S, s_0, \tau, L)$ be an MDP. A *scheduler* η on $\mathcal{M}$ is a function from $\mathsf{Paths}^\star(\mathcal{M})$ to $\mathsf{Distr}(\wp(\mathsf{Distr}(S)))$ such that for all $\sigma \in \mathsf{Paths}^\star(\mathcal{M})$ we have $\eta(\sigma) \in \mathsf{Distr}(\tau(\mathsf{last}(\sigma)))$. We denote the set of all schedulers on $\mathcal{M}$ by $\mathsf{Sch}(\mathcal{M})$.

Note that our schedulers are randomized, i.e., in a finite path σ a scheduler chooses an element of $\tau(\mathsf{last}(\sigma))$ probabilistically. Under a scheduler η, the probability that the next state reached after the path σ is t, equals $\sum_{\pi\in\tau(\mathsf{last}(\sigma))} \eta(\sigma)(\pi)\cdot\pi(t)$. In this way, a scheduler induces a probability measure on $\mathcal{B}_s$ as usual.

Definition 5.2.6. Let $\mathcal{M} = (S, s_0, \tau, L)$ be an MDP and η a scheduler on $\mathcal{M}$. We define the probability measure $\mathbb{P}_\eta$ as the unique measure on $\mathcal{B}_{s_0}$ such that for all $s_0 s_1 \dots s_n \in \mathsf{Paths}^\star(\mathcal{M})$

$$\mathbb{P}_\eta(\langle s_0 s_1 \dots s_n\rangle) = \prod_{i=0}^{n-1} \sum_{\pi\in\tau(s_i)} \eta(s_0 s_1 \dots s_i)(\pi)\cdot\pi(s_{i+1}).$$

We now recall the notions of deterministic and memoryless schedulers.

Definition 5.2.7. Let $\mathcal{M}$ be an MDP and η a scheduler on $\mathcal{M}$. We say that η is *deterministic* if $\eta(\sigma)(\pi_i)$ is either 0 or 1 for all $\pi_i \in \tau(\mathsf{last}(\sigma))$ and all $\sigma \in \mathsf{Paths}^\star(\mathcal{M})$. We say that a scheduler is *memoryless* if for all finite paths σ_1, σ_2 of $\mathcal{M}$ with $\mathsf{last}(\sigma_1) = \mathsf{last}(\sigma_2)$ we have $\eta(\sigma_1) = \eta(\sigma_2)$.

Definition 5.2.8. Let $\mathcal{M}$ be an MDP and $\Delta \in \mathcal{B}_{s_0}$. Then the *maximal probability* $\mathbb{P}^+$ and *minimal probability* $\mathbb{P}^-$ of Δ are defined by

$$\mathbb{P}^+(\Delta) \triangleq \sup_{\eta\in\mathsf{Sch}(\mathcal{M})} \mathbb{P}_\eta(\Delta) \quad \text{and} \quad \mathbb{P}^-(\Delta) \triangleq \inf_{\eta\in\mathsf{Sch}(\mathcal{M})} \mathbb{P}_\eta(\Delta).$$

A scheduler that attains $\mathbb{P}^+(\Delta)$ or $\mathbb{P}^-(\Delta)$ is called a *maximizing* or *minimizing* scheduler respectively.

5.2.3 Markov Chains

A (discrete time) *Markov Chain* is an MDP associating exactly one probability distribution to each state. In this way nondeterministic choices are no longer allowed.

Definition 5.2.9 (Markov Chain)**.** Let $\mathcal{M} = (S, s_0, \tau, L)$ be an MDP. If $|\tau(s)| = 1$ for all $s \in S$, then we say that $\mathcal{M}$ is a *Markov Chain* (MC).

In order to simplify notation we represent probabilistic transitions on MCs by means of a probabilistic matrix $\mathcal{P}$ instead of τ. Additionally, we denote by $\mathbb{P}_{\mathcal{D},s}$ the probability measure induced by a MC $\mathcal{D}$ with initial state s and we abbreviate $\mathbb{P}_{\mathcal{D},s_0}$ as $\mathbb{P}_{\mathcal{D}}$.

5.2.4 Linear Temporal Logic

Linear temporal logic (LTL) [MP91] is a modal temporal logic with modalities referring to time. In LTL is possible to encode formulas about the future of paths: a condition will eventually be true, a condition will be true until another fact becomes true, etc.

Definition 5.2.10. LTL is built up from the set of propositional variables $\mathcal{V}$, the logical connectives $\neg$, $\wedge$, and a temporal modal operator by the following grammar:

$$\phi ::= \mathcal{V} \mid \neg\phi \mid \phi \wedge \phi \mid \phi \mathcal{U} \phi.$$

Using these operators we define $\vee, \rightarrow, \Diamond$, and $\Box$ in the standard way.

Definition 5.2.11. Let $\mathcal{M} = (S, s_0, \tau, L)$ be an MDP. We define satisfiability for paths ω in $\mathcal{M}$, propositional variables $v \in \mathcal{V}$, and LTL formulas ϕ, γ inductively by

$$\begin{array}{lcl} \omega \models_{\mathcal{M}} v & \Leftrightarrow & v \in L(\omega_0) \\ \omega \models_{\mathcal{M}} \neg\phi & \Leftrightarrow & \text{not}(\omega \models_{\mathcal{M}} \phi) \end{array} \qquad \begin{array}{lcl} \omega \models_{\mathcal{M}} \phi \wedge \gamma & \Leftrightarrow & \omega \models_{\mathcal{M}} \phi \text{ and } \omega \models_{\mathcal{M}} \gamma \\ \omega \models_{\mathcal{M}} \phi \mathcal{U} \gamma & \Leftrightarrow & \exists_{i \geq 0}. \omega_{\downarrow i} \models_{\mathcal{M}} \gamma \text{ and } \\ & & \forall_{0 \leq j < i}. \omega_{\downarrow j} \models_{\mathcal{M}} \phi \end{array}$$

where $\omega_{\downarrow i}$ is the i-th suffix of ω. When confusion is unlikely, we omit the subscript $\mathcal{M}$ on the satisfiability relation.

Definition 5.2.12. Let $\mathcal{M}$ be an MDP. We define the language $\text{Sat}_{\mathcal{M}}(\phi)$ associated to an LTL formula ϕ as the set of paths satisfying ϕ, i.e. $\text{Sat}_{\mathcal{M}}(\phi) \triangleq \{\omega \in \mathsf{Paths}(\mathcal{M}) \mid \omega \models \phi\}$. Here we also generally omit the subscript $\mathcal{M}$.

We now define satisfiability of an LTL formula ϕ on an MDP $\mathcal{M}$. We say that $\mathcal{M}$ satisfies ϕ with probability at most p ($\mathcal{M} \models_{\leq p} \phi$) if the probability of getting an execution satisfying ϕ is at most p.

Definition 5.2.13. Let $\mathcal{M}$ be an MDP, ϕ an LTL formula and $p \in [0,1]$. We define $\models_{\leq p}$ and $\models_{\geq p}$ by

$$\mathcal{M} \models_{\leq p} \phi \Leftrightarrow \mathbb{P}^{+}(\mathsf{Sat}(\phi)) \leq p,$$
$$\mathcal{M} \models_{\geq p} \phi \Leftrightarrow \mathbb{P}^{-}(\mathsf{Sat}(\phi)) \geq p.$$

We define $\mathcal{M} \models_{<p} \phi$ and $\mathcal{M} \models_{>p} \phi$ in a similar way. In case the MDP is fully probabilistic, i.e., an MC, the satisfiability problem is reduced to $\mathcal{M} \models_{\bowtie p} \phi \Leftrightarrow \mathbb{P}_{\mathcal{M}}(\mathsf{Sat}(\phi)) \bowtie p$, where $\bowtie \in \{<, \leq, >, \geq\}$.

5.3 Counterexamples

In this section, we define what counterexamples are and how the problem of finding counterexamples for a general LTL property over Markov Decision Processes reduces to finding counterexamples to reachability problems over Markov Chains.

Definition 5.3.1 (Counterexamples)**.** Let $\mathcal{M}$ be an MDP and ϕ an LTL formula. A *counterexample* to $\mathcal{M} \models_{\leq p} \phi$ is a measurable set $\mathcal{C} \subseteq \mathsf{Sat}(\phi)$ such that $\mathbb{P}^{+}(\mathcal{C}) > p$. Counterexamples to $\mathcal{M} \models_{<p} \phi$ are defined similarly.

Counterexamples to $\mathcal{M} \models_{>p} \phi$ and $\mathcal{M} \models_{\geq p} \phi$ cannot be defined straightforwardly as it is always possible to find a set $\mathcal{C} \subseteq \mathsf{Sat}(\phi)$ such that $\mathbb{P}^{-}(\mathcal{C}) \leq p$ or $\mathbb{P}^{-}(\mathcal{C}) < p$, note that the empty set trivially satisfies it. Therefore, the best way to find counterexamples to lower bounded probabilities is to find counterexamples to the dual properties $\mathcal{M} \models_{<1-p} \neg\phi$ and $\mathcal{M} \models_{\leq 1-p} \neg\phi$. That is, while for upper bounded probabilities, a counterexample is a set of paths satisfying the property with mass probability beyond the bound, for lower bounded probabilities the counterexample is a set of paths that *does not* satisfy the property with sufficient probability.

Example 5.3.1. Consider the MDP $\mathcal{M}$ of Figure 5.4 and the LTL formula $\Diamond v$. It is easy to check that $\mathcal{M} \not\models_{<1} \Diamond v$. The set $\mathcal{C} = \mathsf{Sat}(\Diamond v) = \{\rho \in \mathsf{Paths}(s_0) | \exists_{i \geq 0}. \rho = s_0 (s_1)^i (s_4)^\omega\} \cup \{\rho \in \mathsf{Paths}(s_0) | \exists_{i \geq 0}. \rho = s_0 (s_3)^i (s_5)^\omega\}$ is a counterexample. Note that $\mathbb{P}_{\eta}(\mathcal{C}) = 1$ where η is any deterministic scheduler on $\mathcal{M}$ satisfying $\eta(s_0) = \pi_1$.

LTL formulas are actually checked by reducing the model checking problem to a reachability problem [dAKM97]. For checking upper bounded probabilities, the LTL formula is translated into an equivalent deterministic Rabin automaton and composed with the MDP under verification. On the obtained MDP, the set of states forming accepting end components (SCC that traps accepting conditions with probability 1) are identified. The maximum probability of the LTL property on the original MDP is the same as the maximum probability of reaching a state of an accepting end component in the final MDP. Hence, from now on we will focus on counterexamples to properties of the form $\mathcal{M} \models_{\leq p} \Diamond\psi$ or $\mathcal{M} \models_{<p} \Diamond\psi$, where ψ is a propositional formula, i.e., a formula without temporal operators.

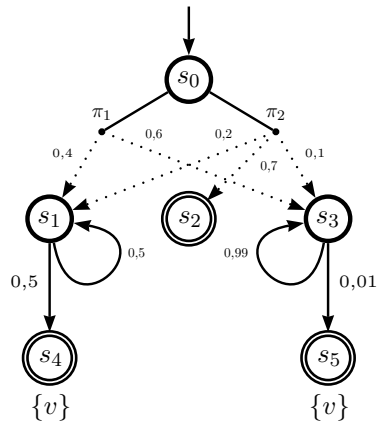


Figure 5.4:

In the following, it will be useful to identify the set of states in which a propositional property is valid.

Definition 5.3.2. Let $\mathcal{M}$ be an MDP. We define the state language $\mathrm{Sat}_{\mathcal{M}}(\psi)$ associated to a propositional formula ψ as the set of states satisfying ψ, i.e., $\mathrm{Sat}_{\mathcal{M}}(\psi) \triangleq \{s \in S \mid s \models \psi\}$, where $\models$ has the obvious satisfaction meaning for states. As usual, we generally omit the subscript $\mathcal{M}$.

We will show now that, in order to find a counterexample to a property in an MDP with respect to an upper bound, it suffices to find a counterexample for the MC *induced* by the maximizing scheduler. The maximizing scheduler turns out to be deterministic and memoryless [BdA95]; consequently the induced Markov Chain can be easily extracted from the MDP as follows.

Definition 5.3.3. Let $\mathcal{M} = (S, s_0, \tau, L)$ be an MDP and η a deterministic

memoryless scheduler. Then we define the MC induced by η as $\mathcal{M}_\eta = (S, s_0, \mathcal{P}_\eta, L)$ where $\mathcal{P}_\eta(s,t) = (\eta(s))(t)$ for all $s, t \in S$.

Now we state that finding counterexamples to upper bounded probabilistic reachability LTL properties on MDPs can be reduced to finding counterexamples to upper bounded probabilistic reachability LTL properties on MCs.

Observation 5.3.2. Let $\mathcal{M}$ be an MDP, ψ a propositional formula and $p \in [0,1]$. Then, there is a maximizing (deterministic memoryless) scheduler η such that $\mathcal{M} \models_{\leq p} \Diamond\psi \Leftrightarrow \mathcal{M}_\eta \models_{\leq p} \Diamond\psi$. Moreover, if $\mathcal{C}$ is a counterexample to $\mathcal{M}_\eta \models_{\leq p} \Diamond\psi$ then $\mathcal{C}$ is also a counterexample to $\mathcal{M} \models_{\leq p} \Diamond\psi$.

Note that η can be computed by solving a linear minimization problem [BdA95] (see Section 5.7.1).

5.4 Representative Counterexamples, Partitions and Witnesses

The notion of counterexample from Definition 5.3.1 is very broad: just an arbitrary (measurable) set of paths with high enough mass probability. To be useful as a debugging tool (and in fact to be able to present the counterexample to a user), we need counterexamples with specific properties. We will partition counterexamples (or rather, representative counterexamples) in witnesses and list five informal properties that we consider valuable in order to increase the quality of witnesses as a debugging tool.

We first note that for reachability properties it is sufficient to consider counterexamples that consist of finite paths.

Definition 5.4.1 (Representative counterexamples)**.** Let $\mathcal{M}$ be an MDP, ψ a propositional formula and $p \in [0,1]$. A *representative counterexample* to $\mathcal{M} \models_{\leq p} \Diamond\psi$ is a set $\mathcal{C} \subseteq \mathsf{Reach}^\star(\mathcal{M}, \mathsf{Sat}(\psi))$ such that $\mathbb{P}^+(\langle\mathcal{C}\rangle) > p$. We denote the set of all representative counterexamples to $\mathcal{M} \models_{\leq p} \Diamond\psi$ by $\mathcal{R}(\mathcal{M}, p, \psi)$.

Observation 5.4.1. Let $\mathcal{M}$ be an MDP, ψ a propositional formula and $p \in [0,1]$. If $\mathcal{C}$ is a representative counterexample to $\mathcal{M} \models_{\leq p} \Diamond\psi$, then $\langle\mathcal{C}\rangle$ is a counterexample to $\mathcal{M} \models_{\leq p} \Diamond\psi$. Furthermore, there exists a counterexample to $\mathcal{M} \models_{\leq p} \Diamond\psi$ if and only if there exists a representative counterexample to $\mathcal{M} \models_{\leq p} \Diamond\psi$.

Following [HK07a], we present the notions of *minimum counterexample*, *strongest evidence* and *most indicative counterexamples*.

Definition 5.4.2 (Minimum counterexample)**.** Let $\mathcal{D}$ be an MC, ψ a propositional formula and $p \in [0,1]$. We say that $\mathcal{C} \in \mathcal{R}(\mathcal{D}, p, \psi)$ is a *minimum counterexample* if $|\mathcal{C}| \leq |\mathcal{C}'|$, for all $\mathcal{C}' \in \mathcal{R}(\mathcal{D}, p, \psi)$.

Definition 5.4.3 (Strongest evidence)**.** Let $\mathcal{D}$ be an MC, ψ a propositional formula and $p \in [0,1]$. A *strongest evidence* to $\mathcal{D} \not\models_{\leq p} \Diamond\psi$ is a finite path $\sigma \in \mathsf{Reach}^{\star}(\mathcal{D}, \mathsf{Sat}(\psi))$ such that $\mathbb{P}_{\mathcal{D}}(\langle\sigma\rangle) \geq \mathbb{P}_{\mathcal{D}}(\langle\rho\rangle)$, for all $\rho \in \mathsf{Reach}^{\star}(\mathcal{D}, \mathsf{Sat}(\psi))$.

Definition 5.4.4 (Most indicative counterexample)**.** Let $\mathcal{D}$ be an MC, ψ a propositional formula and $p \in [0,1]$. We call $\mathcal{C} \in \mathcal{R}(\mathcal{D}, p, \psi)$ a *most indicative counterexample* if it is minimum and $\mathbb{P}_{\mathcal{D}}(\langle\mathcal{C}\rangle) \geq \mathbb{P}_{\mathcal{D}}(\langle\mathcal{C}'\rangle)$, for all minimum counterexamples $\mathcal{C}' \in \mathcal{R}(\mathcal{D}, p, \psi)$.

As pointed out in the Introduction of this chapter, very often most indicative counterexamples are very large (even infinite), many of its elements have insignificant measure and elements can be extremely similar to each other (consequently providing the same diagnostic information). Even worse, sometimes the finite paths with highest probability do not exhibit the way in which the system accumulates higher probability to reach the undesired property (and consequently where an error occurs with higher probability). For these reasons, we are of the opinion that representative counterexamples are still too general in order to be useful as feedback information. We approach this problem by refining a representative counterexample into sets of finite paths following a "similarity" criteria (introduced in Section 5.5). These sets are called *witnesses* of the counterexample.

Recall that a set Y of nonempty sets is a partition of X if the elements of Y cover X and are pairwise disjoint. We define counterexample partitions in the following way.

Definition 5.4.5 (Counterexample partitions and witnesses)**.** Let $\mathcal{M}$ be an MDP, ψ a propositional formula, $p \in [0, 1]$, and $\mathcal{C}$ a representative counterexample to $\mathcal{M} \models_{\leq p} \Diamond\psi$. A *counterexample partition* $W_{\mathcal{C}}$ is a partition of $\mathcal{C}$. We call the elements of $W_{\mathcal{C}}$ *witnesses*.

Since not every partition generates useful witnesses (from the debugging perspective), we now state five informal properties that we consider valuable in order to improve the diagnostic information provided by witnesses. In Section 5.7 we show how to partition the representative counterexample in order to obtain witnesses satisfying most of these properties.

Similarity: Elements of a witness should provide similar debugging information.

Accuracy: Witnesses with higher probability should exhibit evolutions of the system with higher probability of containing errors.

Originality: Different witnesses should provide different debugging information.

Significance: Witnesses should be as closed to the counterexample as possible (their mass probability should be as closed as possible to the bound p).

Finiteness: The number of witnesses of a counterexample partition should be finite.

5.5 Rails and Torrents

As argued before we consider that representative counterexamples are excessively general to be useful as feedback information. Therefore, we group

finite paths of a representative counterexample in witnesses if they are "similar enough". We will consider finite paths that behave the same outside SCCs of the system as providing similar feedback information.

In order to formalize this idea, we first reduce the original MC $\mathcal{D}$ to an acyclic MC preserving reachability probabilities. We do so by removing all SCCs K of $\mathcal{D}$ keeping just *input states* of K. In this way, we get a new acyclic MC denoted by $\mathsf{Ac}(\mathcal{D})$. The probability matrix of the Markov Chain relates input states of each SCC to its *output states* with the reachability probability between these states in $\mathcal{D}$. Secondly, we establish a map between finite paths σ in $\mathsf{Ac}(\mathcal{D})$ (*rails*) and sets of paths W_σ in $\mathcal{D}$ (*torrents*). Each torrent contains finite paths that are similar, i.e., behave the same outside SCCs. We conclude the section showing that the probability of σ is equal to the mass probability of W_σ.

Reduction to Acyclic Markov Chains

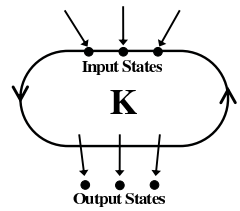
Consider an MC $\mathcal{D} = (S, s_0, \mathcal{P}, L)$. Recall that a subset $\mathsf{K} \subseteq S$ is called *strongly connected* if for every $s, t \in \mathsf{K}$ there is a finite path from s to t. Additionally K is called a *strongly connected component* (SCC) if it is a maximally (with respect to $\subseteq$) strongly connected subset of S.

Note that every state is a member of exactly one SCC of $\mathcal{D}$; even those states that are not involved in cycles, since the trivial finite path s connects s to itself. We call *trivial strongly connected components* to the SCCs containing absorbing states or states not involved in cycles (note that trivial SCCs are composed by a single state). From now on we let $\mathsf{SCC}^\star$ be the set of non trivial strongly connected components of an MC.

A Markov Chain is called *acyclic* if it contains only trivial SCCs. Note that an acyclic Markov Chain still has absorbing states.

Definition 5.5.1 (Input and Output states)**.** Let $\mathcal{D} = (S, s_0, \mathcal{P}, L)$ be an MC. Then, for each $\mathsf{SCC}^\star$ K of $\mathcal{D}$, we define the sets $\mathsf{Inp}_\mathsf{K} \subseteq S$ of all states in K that have an incoming transition from a state outside of K and $\mathsf{Out}_\mathsf{K} \subseteq S$ of all states outside of K that have an incoming transition from a state of K in the following way

$$\mathsf{Inp}_{\mathsf{K}} \triangleq \{t \in \mathsf{K} \mid \exists\, s \in S \setminus \mathsf{K}\,.\mathcal{P}(s,t) > 0\},$$
$$\mathsf{Out}_{\mathsf{K}} \triangleq \{s \in S \setminus \mathsf{K} \mid \exists\, t \in \mathsf{K}\,.\mathcal{P}(t,s) > 0\}.$$



We also define for each $\mathsf{SCC}^\star$ K an MC related to K as $\mathcal{D}_{\mathsf{K}} \triangleq (\mathsf{K} \cup \mathsf{Out}_{\mathsf{K}}, s_{\mathsf{K}}, \mathcal{P}_{\mathsf{K}}, L_{\mathsf{K}})$ where s_{K} is any state in $\mathsf{Inp}_{\mathsf{K}}$, $L_{\mathsf{K}}(s) \triangleq L(s)$, and $\mathcal{P}_{\mathsf{K}}(s,t)$ is equal to $\mathcal{P}(s,t)$ if $s \in \mathsf{K}$ and equal to 1_s otherwise. Additionally, for every state s involved in non trivial SCCs we define SCC_s^+ as $\mathcal{D}_{\mathsf{K}}$, where K is the $\mathsf{SCC}^\star$ of $\mathcal{D}$ such that $s \in \mathsf{K}$.

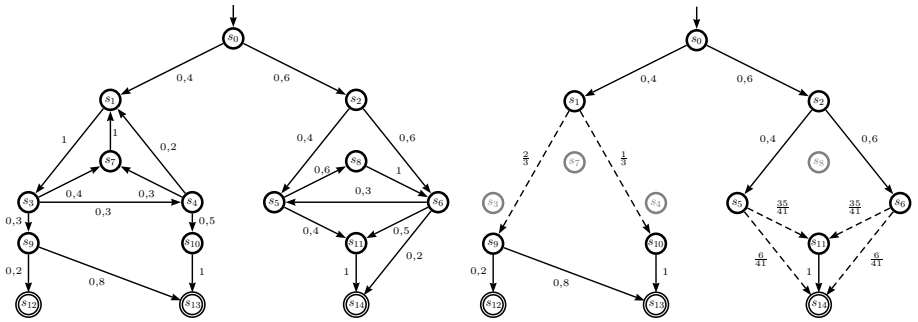
Now we are able to define an acyclic MC $\mathsf{Ac}(\mathcal{D})$ related to $\mathcal{D}$.

Definition 5.5.2. Let $\mathcal{D} = (S, s_0, \mathcal{P}, L)$ be a MC. We define $\mathsf{Ac}(\mathcal{D}) \triangleq (S', s_0, \mathcal{P}', L')$ where

- $S' \triangleq S \setminus \overbrace{\bigcup_{\mathsf{K} \in \mathsf{SCC}^\star} \mathsf{K}}^{S_{\text{com}}} \cup \overbrace{\bigcup_{\mathsf{K} \in \mathsf{SCC}^\star} \mathsf{Inp}_{\mathsf{K}}}^{S_{\text{inp}}}$,
- $L' \triangleq L_{|_{S'}}$,
- $\mathcal{P}'(s,t) \triangleq \begin{cases} \mathcal{P}(s,t) & \text{if } s \in S_{com}, \\ \mathbb{P}_{\mathcal{D},s}(\mathsf{Reach}(\mathsf{SCC}_s^+, s, \{t\})) & \text{if } s \in S_{inp} \wedge t \in \mathsf{Out}_{\mathsf{SCC}_s^+}, \\ 1_s & \text{if } s \in S_{inp} \wedge \mathsf{Out}_{\mathsf{SCC}_s^+} = \emptyset, \\ 0 & \text{otherwise.} \end{cases}$

Note that $\mathsf{Ac}(\mathcal{D})$ is indeed acyclic.

Example 5.5.1. Consider the MC $\mathcal{D}$ of Figure 5.5(a). The strongly connected components of $\mathcal{D}$ are $\mathsf{K}_1 \triangleq \{s_1, s_3, s_4, s_7\}$, $\mathsf{K}_2 \triangleq \{s_5, s_6, s_8\}$ and the singletons $\{s_0\}$, $\{s_2\}$, $\{s_9\}$, $\{s_{10}\}$, $\{s_{11}\}$, $\{s_{12}\}$, $\{s_{13}\}$, and $\{s_{14}\}$. The input states of K_1 are $\mathsf{Inp}_{\mathsf{K}_1} = \{s_1\}$ and its output states are $\mathsf{Out}_{\mathsf{K}_1} = \{s_9, s_{10}\}$. For K_2, $\mathsf{Inp}_{\mathsf{K}_2} = \{s_5, s_6\}$ and $\mathsf{Out}_{\mathsf{K}_2} = \{s_{11}, s_{14}\}$. The reduced acyclic MC of $\mathcal{D}$ is shown in Figure 5.5(b).



(a) Original MC (b) Derived Acyclic MC

Figure 5.5:

Rails and Torrents

We now relate (finite) paths in $\mathsf{Ac}(\mathcal{D})$ (rails) to sets of paths in $\mathcal{D}$ (torrents).

Definition 5.5.3 (Rails)**.** Let $\mathcal{D}$ be an MC. A finite path $\sigma \in \mathsf{Paths}^\star(\mathsf{Ac}(\mathcal{D}))$ will be called a *rail of* $\mathcal{D}$.

Consider a rail σ, i.e., a finite path of $\mathsf{Ac}(\mathcal{D})$. We will use σ to represent those paths ω of $\mathcal{D}$ that behave "similar to" σ outside SCCs of $\mathcal{D}$. Naively, this means that σ is a subsequence of ω. There are two technical subtleties to deal with: every input state in σ must be the first state in its SCC in ω (freshness) and every SCC visited by ω must be also visited by σ (inertia) (see Definition 5.5.5). We need these extra conditions to make sure that no path ω behaves "similar to" two distinct rails (see Lemma 5.5.7).

Recall that given a finite sequence σ and a (possible infinite) sequence ω, we say that σ is a *subsequence* of ω, denoted by $\sigma \sqsubseteq \omega$, if and only if there exists a strictly increasing function $f : \{0, 1, \ldots, |\sigma|-1\} \to \{0, 1, \ldots, |\omega|-1\}$ such that $\forall_{0 \leq i < |\sigma|}.\sigma_i = \omega_{f(i)}$. If ω is an infinite sequence, we interpret the codomain of f as $\mathbb{N}$. In case f is such a function we write $\sigma \sqsubseteq_f \omega$.

Definition 5.5.4. Let $\mathcal{D} = (S, s_0, \mathcal{P}, L)$ be an MC. On S we consider the equivalence relation $\asymp_{\mathcal{D}}$ satisfying $s \asymp_{\mathcal{D}} t$ if and only if s and t are in the same strongly connected component. Again, we usually omit the subscript $\mathcal{D}$ from the notation.

The following definition refines the notion of subsequence, taking care of the two technical subtleties noted above.

Definition 5.5.5. Let $\mathcal{D} = (S, s_0, \mathcal{P}, L)$ be an MC, ω a (finite) path of $\mathcal{D}$, and $\sigma \in \mathsf{Paths}^\star(\mathsf{Ac}(\mathcal{D}))$ a finite path of $\mathsf{Ac}(\mathcal{D})$. Then we write $\sigma \preceq \omega$ if there exists $f : \{0, 1, \ldots, |\sigma| - 1\} \to \mathbb{N}$ such that $\sigma \sqsubseteq_f \omega$ and

$$\forall_{0 \leq j < f(i)} : \omega_{f(i)} \not\sim \omega_j; \text{ for all } i = 0, 1, \ldots |\sigma| - 1, \quad \{\textit{Freshness property}\}$$
$$\forall_{f(i) < j < f(i+1)} : \omega_{f(i)} \sim \omega_j; \text{ for all } i = 0, 1, \ldots |\sigma| - 2. \qquad \{\textit{Inertia property}\}$$

In case f is such a function we write $\sigma \preceq_f \omega$.

Example 5.5.2. Let $\mathcal{D} = (S, s_0, \mathcal{P}, L)$ be the MC of Figure 5.5(a) and take $\sigma = s_0 s_2 s_6 s_{14}$. Then for all $i \in \mathbb{N}$ we have $\sigma \preceq_{f_i} \omega_i$ where $\omega_i = s_0 s_2 s_6 (s_5 s_8 s_6)^i s_{14}$ and $f_i(0) \triangleq 0$, $f_i(1) \triangleq 1$, $f_i(2) \triangleq 2$, and $f_i(3) \triangleq 3 + 3i$. Additionally, $\sigma \not\preceq s_0 s_2 s_5 s_8 s_6 s_{14}$ since for all f satisfying $\sigma \sqsubseteq_f s_0 s_2 s_5 s_8 s_6 s_{14}$ we must have $f(2) = 4$; this implies that f does not satisfy the freshness property. Finally, note that $\sigma \not\preceq s_0 s_2 s_6 s_{11} s_{14}$ since for all f satisfying $\sigma \sqsubseteq_f s_0 s_2 s_6 s_{11} s_{14}$ we must have $f(2) = 2$; this implies that f does not satisfy the inertia property.

We now give the formal definition of torrents.

Definition 5.5.6 (Torrents)**.** Let $\mathcal{D} = (S, s_0, \mathcal{P}, L)$ be an MC and σ a sequence of states in S. We define the function Torr by

$$\mathsf{Torr}(\mathcal{D}, \sigma) \triangleq \{\omega \in \mathsf{Paths}(\mathcal{D}) \mid \sigma \preceq \omega\}.$$

We call $\mathsf{Torr}(\mathcal{D}, \sigma)$ the *torrent* associated to σ.

We now show that torrents are disjoint (Lemma 5.5.7) and that the probability of a rail is equal to the probability of its associated torrent (Theorem 5.5.10). For this last result, we first show that torrents can be represented as the disjoint union of cones of finite paths. We call these finite paths *generators* of the torrent (Definition 5.5.8).

Lemma 5.5.7. Let $\mathcal{D}$ be an MC. For every $\sigma, \rho \in \mathsf{Paths}^\star(\mathsf{Ac}(\mathcal{D}))$ we have

$$\sigma \neq \rho \Rightarrow \mathsf{Torr}(\mathcal{D}, \sigma) \cap \mathsf{Torr}(\mathcal{D}, \rho) = \emptyset.$$

Definition 5.5.8 (Torrent Generators)**.** Let $\mathcal{D}$ be an MC. Then we define for every rail $\sigma \in \mathsf{Paths}^\star(\mathsf{Ac}(\mathcal{D}))$ the set

$$\mathsf{TorrGen}(\mathcal{D}, \sigma) \triangleq \{\rho \in \mathsf{Paths}^\star(\mathcal{D}) \mid \exists f : \sigma \preceq_f \rho \wedge f(|\sigma| - 1) = |\rho| - 1\}.$$

In the example from the Introduction (see Figure 5.1), $s_0 s_1 s_3$ and $s_0 s_2 s_4$ are rails. Their associated torrents are, respectively, $\{s_0 s_1^n s_3^\omega \mid n \in \mathbb{N}^*\}$ and $\{s_0 s_2^n s_4^\omega \mid n \in \mathbb{N}^*\}$ (note that s_3 and s_4 are absorbing states), i.e. the paths going left and the paths going right. The generators of the first torrent are $\{s_0 s_1^n s_3 \mid n \in \mathbb{N}^*\}$ and similarly for the second torrent.

Lemma 5.5.9. Let $\mathcal{D}$ be an MC and $\sigma \in \mathsf{Paths}^\star(\mathsf{Ac}(\mathcal{D}))$ a rail of $\mathcal{D}$. Then we have

$$\mathsf{Torr}(\mathcal{D}, \sigma) = \biguplus_{\rho \in \mathsf{TorrGen}(\mathcal{D}, \sigma)} \langle \rho \rangle.$$

Proof. The proof is by cases on the length of σ. We prove the result for the cases on which σ is of the form $\sigma t s$, with t an input state and s an output state, the other cases are simpler. In order to prove this lemma, we define (for each $\sigma s t$ of the above form) the following set of finite paths

$$\Delta_{\sigma t s} \triangleq \{\rho\, \mathsf{tail}(\pi) \mid \rho \in \mathsf{TorrGen}(\sigma t) \text{ and } \pi \in \mathsf{Paths}^\star(\mathsf{SCC}_t^+, t, \{s\})\} \quad (5.1)$$

Checking that $\mathsf{Torr}(\mathcal{D}, \sigma) = \biguplus_{\rho \in \Delta_{\sigma t s}} \langle \rho \rangle$ is straightforward. We now focus on proving that

$$\Delta_{\sigma t s} = \mathsf{TorrGen}(\mathcal{D}, \sigma). \quad (5.2)$$

For that purpose we need the following two observations.

Observation 5.5.3. Let $\mathcal{D}$ be a MC. Since $\mathsf{Ac}(\mathcal{D})$ is acyclic we have $\sigma_i \not\sim \sigma_j$ for every $\sigma \in \mathsf{Paths}^\star(\mathsf{Ac}(\mathcal{D}))$ and $i \neq j$ (with the exception of absorbing states).

Observation 5.5.4. Let σ, ω and f be such that $\sigma \preceq_f \omega$. Then $\forall i : \exists j : \omega_i \sim \sigma_j$. This follows from $\sigma \sqsubseteq_f \omega$ and the inertia property.

We now proceed to prove that $\Delta_{\sigma ts} = \mathsf{TorrGen}(\mathcal{D}, \sigma)$.
($\supseteq$) Let $\rho_0\rho_1 \cdots \rho_k \in \mathsf{TorrGen}(\sigma ts)$ and n_t the lowest subindex of ρ such that $\rho_{n_t} = t$. Take $\rho \triangleq \rho_0\rho_1 \cdots \rho_{n_t}$ and $\pi \triangleq \rho_{n_t} \cdots \rho_k$ (Note that $\rho_0\rho_1 \cdots \rho_k = \rho\,\mathsf{tail}(\pi)$). In order to prove that $\rho_0\rho_1 \cdots \rho_k \in \Delta_{\sigma ts}$ it is enough to show that

(1) $\rho \in \mathsf{TorrGen}(\sigma t)$, and

(2) $\pi \in \mathsf{Paths}^\star(\mathsf{SCC}_t^+, t, \{s\})$.

(1) Let f be such that $\sigma ts \preceq_f \rho_0\rho_1 \cdots \rho_k$ and $f(|\sigma ts| - 1) = k$. Take $g : \{0, 1, \ldots, |\sigma t| - 1\} \to \mathbb{N}$ be the restriction of f. It is easy to check that $\sigma t \preceq_g \rho$. Additionally $f(|\sigma t| - 1) = n_t$ (otherwise f would not satisfy the freshness property for $i = |\sigma t| - 1$). Then, by definition of g, we have $g(|\sigma t| - 1) = n_t$.

(2) It is clear that π is a path from t to s. Therefore we only have to show that every state of π is in SCC_t^+. By definition of SCC_t^+, $\pi_0 = t \in \mathsf{SCC}_t^+$ and $s \in \mathsf{SCC}_t^+$ since $s \in \mathsf{Out}_{\mathsf{SCC}_t^+}$. Additionally, since f satisfies inertia property we have that $\forall_{f(|\sigma t|-1)<j<f(|\sigma ts|-1)} : \rho_{f(|\sigma t|-1)} \sim \rho_j$, since $f(|\sigma t| - 1) = n_t$ and $\pi \triangleq \rho_{n_t} \cdots \rho_k$ we have $\forall_{0<j<|\pi|-1} : t \sim \pi_j$ proving that $\pi_j \in \mathsf{SCC}_t^+$ for $j \in \{1, \cdots, |\pi| - 2\}$.

($\subseteq$) Take $\rho \in \mathsf{TorrGen}(\sigma t)$ and $\mathsf{tail}(\pi) \in \mathsf{Paths}^\star(\mathsf{SCC}_t^+, t, \{s\})$. In order to prove that $\rho\,\mathsf{tail}(\pi) \in \mathsf{TorrGen}(\sigma ts)$ we need to show that there exists a function g such that:

(1) $\sigma ts \preceq_g \rho\,\mathsf{tail}(\pi)$,

(2) $g(|\sigma ts| - 1) = |\rho\,\mathsf{tail}(\pi)| - 1$.

Since $\rho \in \mathsf{TorrGen}(\sigma t)$ we know that there exists f be such that $\sigma t \preceq_f \rho$ and $f(|\sigma t| - 1) = |\rho| - 1$. We define $g : \{0, 1, \ldots, |\sigma ts| - 1\} \to \{0, 1, \ldots,$

$|\rho\,\mathsf{tail}(\pi)| - 1\}$ by

$$g(i) \triangleq \begin{cases} f(i) & \text{if } i < |\sigma ts| - 1, \\ |\rho\,\mathsf{tail}(\pi)| - 1 & \text{if } i = |\sigma ts| - 1. \end{cases}$$

(1) It is easy to check that $\sigma ts \sqsubseteq_g \rho\,\mathsf{tail}(\pi)$. Now we will show that g satisfies Freshness and Inertia properties.

Freshness property: We need to show that for all $0 \leq i < |\sigma ts|$ we have $\forall_{0\leq j<g(i)} : \rho\,\mathsf{tail}(\pi)_{g(i)} \not\sim \rho\,\mathsf{tail}(\pi)_j$. For the cases $i \in \{0, \ldots, |\sigma t| - 1\}$ this holds since $\sigma t \preceq_f \rho$ and definition of g.

Consider $i = |\sigma ts| - 1$, in this case we have to prove $\forall_{0\leq j<|\rho\,\mathsf{tail}(\pi)|-1} : \rho\,\mathsf{tail}(\pi)_{|\rho\,\mathsf{tail}(\pi)|-1)} \not\sim \rho\,\mathsf{tail}(\pi)_j$ or equivalently $\forall_{0\leq j<|\rho\,\mathsf{tail}(\pi)|-1} : s \not\sim \rho\,\mathsf{tail}(\pi)_j$.

Case $j \in \{|\rho|, \ldots |\rho\,\mathsf{tail}(\pi)| - 1\}$.
From $\pi \in \mathsf{Paths}^\star(\mathsf{SCC}_t^+, t, \{s\})$ and $s \in \mathsf{Out}^+_{\mathsf{SCC}_t^+}$ it is easy to see $\forall_{0\leq j<|\,\mathsf{tail}(\pi)|-1} : s \not\sim \mathsf{tail}(\pi)_j$

Case $j \in \{0, \ldots, |\rho| - 1\}$.
From $\sigma ts \in \mathsf{Paths}^\star(\mathsf{Ac}(\mathcal{D}))$ and Observation 5.5.3 we have that $\forall_{0\leq j<|\sigma t|-1} : s \not\sim \sigma t_j$. Additionally, $\sigma t \preceq_f \rho$ and definition of g and Observation 5.5.4 imply $\forall_{0\leq j<|\rho|} : s \not\sim \rho_j$ or equivalently $\forall_{0\leq j<|\rho|} : s \not\sim \rho\,\mathsf{tail}(\pi)_j$.

Inertia property: Since $\pi \in \mathsf{Paths}^\star(SCC_t^+, t, \{s\})$ we have $\forall_{0\leq j<|\pi|-1} : t \sim \pi_j$ which implies that $\forall_{|\rho|-1<j<|\rho\,\mathsf{tail}(\pi)|-1} : \rho\,\mathsf{tail}(\pi)_{|\rho|-1} \sim \rho\,\mathsf{tail}(\pi)_j$ or equivalently $\forall_{g(|\sigma|-1)<j<g(|\sigma s|-1)} : \rho\,\mathsf{tail}(\pi)_{g(|\rho|-1)} \sim \rho\,\mathsf{tail}(\pi)_j$ showing that g satisfies the inertia property.

(2) Follows from the definition of g. □

Theorem 5.5.10. Let $\mathcal{D}$ be an MC. Then for every rail $\sigma \in \mathsf{Paths}^\star(\mathsf{Ac}(\mathcal{D}))$ we have

$$\mathbb{P}_{\mathsf{Ac}(\mathcal{D})}(\langle\sigma\rangle) = \mathbb{P}_{\mathcal{D}}(\mathsf{Torr}(\mathcal{D}, \sigma)).$$

Proof. By induction on the structure of σ.

Base Case: Note that $\mathbb{P}_{\mathsf{Ac}(\mathcal{D})}(\langle s_0 \rangle) = \mathbb{P}_{\mathsf{Ac}(\mathcal{D})}(\mathsf{Paths}(\mathsf{Ac}(\mathcal{D}), s_0)) = 1$, and similarly $1 = \mathbb{P}_{\mathcal{D}}(\mathsf{Paths}(\mathcal{D}, s_0)) = \mathbb{P}_{\mathcal{D}}(\mathsf{Torr}(s_0))$.

Inductive Step: Let t be such that $\mathsf{last}(\sigma) = t$. Suppose that $t \in S_{\mathsf{Com}}$ and denote by $\mathsf{Ac}(\mathcal{P})$ to the probability matrix of $\mathsf{Ac}(\mathcal{D})$. Then

$$
\begin{aligned}
&\mathbb{P}_{\mathsf{Ac}(\mathcal{D})}(\langle \sigma s \rangle) \\
&\quad = \mathbb{P}_{\mathsf{Ac}(\mathcal{D})}(\langle \sigma \rangle) \cdot \mathsf{Ac}(\mathcal{P})(t, s) \\
&\qquad \{\text{Inductive Hypothesis and definition of } \mathcal{P}\} \\
&\quad = \mathbb{P}_{\mathcal{D}}(\mathsf{Torr}(\sigma)) \cdot \mathcal{P}(t, s) \\
&\quad = \mathbb{P}_{\mathcal{D}}(\biguplus_{\rho \in \mathsf{TorrGen}(\sigma)} \langle \rho \rangle) \cdot \mathcal{P}(t, s) \qquad \{\text{Lem. 5.5.9}\} \\
&\quad = \textstyle\sum_{\rho \in \mathsf{TorrGen}(\sigma)} \mathbb{P}_{\mathcal{D}}(\langle \rho \rangle) \cdot \mathbb{P}_{\mathcal{D}}(\langle ts \rangle) \\
&\quad = \textstyle\sum_{\rho \in \mathsf{TorrGen}(\sigma)} \mathbb{P}_{\mathcal{D}}(\langle \rho\, \mathsf{tail}(ts) \rangle) \\
&\quad = \textstyle\sum_{\rho \in \mathsf{TorrGen}(\sigma s)} \mathbb{P}_{\mathcal{D}}(\langle \rho \rangle) \\
&\quad = \mathbb{P}_{\mathcal{D}}(\biguplus_{\rho \in \mathsf{TorrGen}(\sigma s)} \langle \rho \rangle) \\
&\quad = \mathbb{P}_{\mathcal{D}}(\mathsf{Torr}(\sigma s)) \qquad \{\text{Lem. 5.5.9}\}
\end{aligned}
$$

Now suppose that $t \in S_{\mathsf{Inp}}$, then

$$
\begin{aligned}
&\mathbb{P}_{\mathsf{Ac}(\mathcal{D})}(\langle \sigma s \rangle) \\
&\quad = \mathbb{P}_{\mathsf{Ac}(\mathcal{D})}(\langle \sigma \rangle) \cdot \mathsf{Ac}(\mathcal{P})(t, s) \\
&\quad = \mathbb{P}_{\mathcal{D}}(\mathsf{Torr}(\sigma)) \cdot \mathsf{Ac}(\mathcal{P})(t, s) \qquad \{\text{HI}\} \\
&\quad = \mathbb{P}_{\mathcal{D}}(\biguplus_{\rho \in \mathsf{TorrGen}(\sigma)} \langle \rho \rangle) \cdot \mathsf{Ac}(\mathcal{P})(t, s) \qquad \{\text{Lem. 5.5.9}\} \\
&\quad = \left(\textstyle\sum_{\rho \in \mathsf{TorrGen}(\sigma)} \mathbb{P}_{\mathcal{D}}(\langle \rho \rangle) \right) \cdot \mathsf{Ac}(\mathcal{P})(t, s) \\
&\quad = \textstyle\sum_{\rho \in \mathsf{TorrGen}(\sigma)} \mathbb{P}_{\mathcal{D}}(\langle \rho \rangle) \cdot \mathbb{P}_{\mathcal{D},t}(\mathsf{Paths}(\mathsf{SCC}_t^+, t, \{s\})) \\
&\qquad \{\text{By definition of } \mathsf{Ac}(\mathcal{P}) \text{ and distributivity}\} \\
&\quad = \textstyle\sum_{\rho \in \mathsf{TorrGen}(\sigma)} \mathbb{P}_{\mathcal{D}}(\langle \rho \rangle) \cdot \sum_{\pi \in \mathsf{Paths}^\star(\mathsf{SCC}_t^+, t, \{s\})} \mathbb{P}_{\mathcal{D},t}(\langle \pi \rangle) \\
&\quad = \textstyle\sum_{\rho \in \mathsf{TorrGen}(\sigma), \pi \in \mathsf{Paths}^\star(\mathsf{SCC}_t^+, t, \{s\})} \mathbb{P}_{\mathcal{D}}(\langle \rho\, \mathsf{tail}(\pi) \rangle) \qquad \{\text{Dfn. } \mathbb{P}\} \\
&\quad = \textstyle\sum_{\rho \in \Delta_{\sigma s}} \mathbb{P}_{\mathcal{D}}(\langle \rho \rangle) \qquad \{(5.1)\} \\
&\quad = \textstyle\sum_{\rho \in \mathsf{TorrGen}(\sigma s)} \mathbb{P}_{\mathcal{D}}(\langle \rho \rangle) \qquad \{(5.2)\} \\
&\quad = \mathbb{P}_{\mathcal{D}}(\biguplus_{\rho \in \mathsf{TorrGen}(\sigma s)} \langle \rho \rangle) \\
&\quad = \mathbb{P}_{\mathcal{D}}(\mathsf{Torr}(\sigma s)) \qquad \square
\end{aligned}
$$

5.6 Significant Diagnostic Counterexamples

So far we have formalized the notion of paths behaving similarly (i.e., behaving the same outside SCCs) in an MC $\mathcal{D}$ by removing all SCC of $\mathcal{D}$, obtaining $\mathsf{Ac}(\mathcal{D})$. A representative counterexample to $\mathsf{Ac}(\mathcal{D}) \models_{\leq p} \Diamond\psi$ gives rise to a representative counterexample to $\mathcal{D} \models_{\leq p} \Diamond\psi$ in the following way: for every finite path σ in the representative counterexample to $\mathsf{Ac}(\mathcal{D}) \models_{\leq p} \Diamond\psi$ the set $\mathsf{TorrGen}(\mathcal{D}, \sigma)$ is a witness, then we obtain the desired representative counterexample to $\mathcal{D} \models_{\leq p} \Diamond\psi$ by taking the union of these witnesses.

Before giving a formal definition, there is still one technical issue to resolve: we need to be sure that by removing SCCs we are not discarding useful information. Because torrents are built from rails, we need to make sure that when we discard SCCs, we do not discard rails that reach ψ.

We achieve this by first making states satisfying ψ absorbing. Additionally, we make absorbing states from which it is not possible to reach ψ. Note that this does not affect counterexamples.

Definition 5.6.1. Let $\mathcal{D} = (S, s_0, \mathcal{P}, L)$ be an MC and ψ a propositional formula. We define the MC $\mathcal{D}_\psi \triangleq (S, s_0, \mathcal{P}_\psi, L)$, with

$$\mathcal{P}_\psi(s,t) \triangleq \begin{cases} 1 & \text{if } s \notin \mathsf{Sat}_\Diamond(\psi) \wedge s = t, \\ 1 & \text{if } s \in \mathsf{Sat}(\psi) \wedge s = t, \\ \mathcal{P}(s,t) & \text{if } s \in \mathsf{Sat}_\Diamond(\psi) - \mathsf{Sat}(\psi), \\ 0 & \text{otherwise,} \end{cases}$$

where $\mathsf{Sat}_\Diamond(\psi) \triangleq \{s \in S \mid \mathbb{P}_{\mathcal{D},s}(\mathsf{Reach}(\mathcal{D}, s, \mathsf{Sat}(\psi))) > 0\}$ is the set of states reaching ψ in $\mathcal{D}$.

The following corollary shows the relation between paths, finite paths, and probabilities of $\mathcal{D}$, $\mathcal{D}_\psi$, and $\mathsf{Ac}(\mathcal{D}_\psi)$. Most importantly, the probability of a rail σ (in $\mathsf{Ac}(\mathcal{D}_\psi)$) is equal to the probability of its associated torrent (in $\mathcal{D}$) (item 5 below) and the probability of $\Diamond\psi$ is not affected by reducing $\mathcal{D}$ to $\mathsf{Ac}(\mathcal{D}_\psi)$ (item 6 below).

Note that a rail σ is always a finite path in $\mathsf{Ac}(\mathcal{D}_\psi)$, but that we can talk about its associated torrent $\mathsf{Torr}(\mathcal{D}_\psi, \sigma)$ in $\mathcal{D}_\psi$ and about its associated

torrent $\mathsf{Torr}(\mathcal{D}, \sigma)$ in $\mathcal{D}$. The former exists for technical convenience; it is the latter that we are ultimately interested in. The following theorem also shows that for our purposes, viz. the definition of the generators of the torrent and the probability of the torrent, there is no difference (items 3 and 4 below).

Corollary 5.6.1. Let $\mathcal{D} = (S, s_0, \mathcal{P}, L)$ be an MC and ψ a propositional formula. Then for every $\sigma \in \mathsf{Paths}^\star(\mathcal{D}_\psi)$

1. $\mathsf{Reach}^\star(\mathcal{D}_\psi, s_0, \mathsf{Sat}(\psi)) = \mathsf{Reach}^\star(\mathcal{D}, s_0, \mathsf{Sat}(\psi))$,
2. $\mathbb{P}_{\mathcal{D}_\psi}(\langle\sigma\rangle) = \mathbb{P}_{\mathcal{D}}(\langle\sigma\rangle)$,
3. $\mathsf{TorrGen}(\mathcal{D}_\psi, \sigma) = \mathsf{TorrGen}(\mathcal{D}, \sigma)$,
4. $\mathbb{P}_{\mathcal{D}_\psi}(\mathsf{Torr}(\mathcal{D}_\psi, \sigma)) = \mathbb{P}_{\mathcal{D}}(\mathsf{Torr}(\mathcal{D}, \sigma))$,
5. $\mathbb{P}_{\mathsf{Ac}(\mathcal{D}_\psi)}(\langle\sigma\rangle) = \mathbb{P}_{\mathcal{D}}(\mathsf{Torr}(\mathcal{D}, \sigma))$,
6. $\mathsf{Ac}(\mathcal{D}_\psi) \models_{\leq p} \Diamond\psi$ if and only if $\mathcal{D} \models_{\leq p} \Diamond\psi$, for any $p \in [0, 1]$.

Definition 5.6.2 (Torrent-Counterexamples)**.** Let $\mathcal{D} = (S, s_0, \mathcal{P}, L)$ be an MC, ψ a propositional formula, and $p \in [0, 1]$ such that $\mathcal{D} \not\models_{\leq p} \Diamond\psi$. Let $\mathcal{C}$ be a representative counterexample to $\mathsf{Ac}(\mathcal{D}_\psi) \models_{\leq p} \Diamond\psi$. We define the set

$$\mathsf{TorRepCount}(\mathcal{C}) \triangleq \{\mathsf{TorrGen}(\mathcal{D}, \sigma) \mid \sigma \in \mathcal{C}\}.$$

We call the set $\mathsf{TorRepCount}(\mathcal{C})$ a *torrent-counterexample* of $\mathcal{C}$. Note that this set is a partition of a representative counterexample to $\mathcal{D} \models_{\leq p} \Diamond\psi$. Additionally, we denote by $\mathcal{R}_t(\mathcal{D}, p, \psi)$ to the set of all torrent-counterexamples to $\mathcal{D} \models_{\leq p} \Diamond\psi$, i.e., $\{\mathsf{TorRepCount}(\mathcal{C}) \mid \mathcal{C} \in \mathcal{R}(\mathsf{Ac}(\mathcal{D}), p, \psi)\}$.

Theorem 5.6.3. Let $\mathcal{D} = (S, s_0, \mathcal{P}, L)$ be an MC, ψ a propositional formula, and $p \in [0, 1]$ such that $\mathcal{D} \not\models_{\leq p} \Diamond\psi$. Take $\mathcal{C}$ a representative counterexample to $\mathsf{Ac}(\mathcal{D}_\psi) \models_{\leq p} \Diamond\psi$. Then the set of finite paths $\biguplus_{W \in \mathsf{TorRepCount}(\mathcal{C})} W$ is a representative counterexample to $\mathcal{D} \models_{\leq p} \Diamond\psi$.

Note that for each $\sigma \in \mathcal{C}$ we get a witness $\mathsf{TorrGen}(\mathcal{D}, \sigma)$. Also note that the number of rails is finite, so there are also only finitely many witnesses.

Following [HK07a], we extend the notions of *minimum counterexamples* and *strongest evidence*.

Definition 5.6.4 (Minimum torrent-counterexample)**.** Let $\mathcal{D}$ be an MC, ψ a propositional formula and $p \in [0, 1]$. We say that $\mathcal{C}_t \in \mathcal{R}_t(\mathcal{D}, p, \psi)$ is a *minimum torrent-counterexample* if $|\mathcal{C}_t| \leq |\mathcal{C}'_t|$, for all $\mathcal{C}'_t \in \mathcal{R}_t(\mathcal{D}, p, \psi)$.

Definition 5.6.5 (Strongest torrent-evidence)**.** Let $\mathcal{D}$ be an MC, ψ a propositional formula and $p \in [0, 1]$. A *strongest torrent-evidence* to $\mathcal{D} \not\models_{\leq_p} \Diamond\psi$ is a torrent $\mathsf{Torr}(\mathcal{D}, \sigma)$ such that $\sigma \in \mathsf{Paths}^\star(\mathsf{Ac}(\mathcal{D}_\psi))$ and $\mathbb{P}_{\mathcal{D}}(\mathsf{Torr}(\mathcal{D}, \sigma)) \geq \mathbb{P}_{\mathcal{D}}(\mathsf{Torr}(\mathcal{D}, \rho))$ for all $\rho \in \mathsf{Paths}^\star(\mathsf{Ac}(\mathcal{D}_\psi))$.

Now we define our notion of significant diagnostic counterexamples. It is the generalization of most indicative counterexample from [HK07a] to our setting.

Definition 5.6.6 (Most indicative torrent-counterexample)**.** Let $\mathcal{D}$ be an MC, ψ a propositional formula and $p \in [0, 1]$. We say that $\mathcal{C}_t \in \mathcal{R}_t(\mathcal{D}, p, \psi)$ is a *most indicative torrent-counterexample* if it is a minimum torrent-counterexample and $\mathbb{P}(\bigcup_{T \in \mathcal{C}_t} \langle T \rangle) \geq \mathbb{P}(\bigcup_{T \in \mathcal{C}'_t} \langle T \rangle)$ for all minimum torrent-counterexamples $\mathcal{C}'_t \in \mathcal{R}_t(\mathcal{D}, p, \psi)$.

Note that in our setting, as in [HK07a], a minimal torrent-counterexample $\mathcal{C}$ consists of the $|\mathcal{C}|$ strongest torrent-evidences.

By Theorem 5.6.3 it is possible to obtain strongest torrent-evidence and most indicative torrent-counterexamples of an MC $\mathcal{D}$ by obtaining strongest evidence and most indicative counterexamples of $\mathsf{Ac}(\mathcal{D}_\psi)$ respectively.

5.7 Computing Counterexamples

In this section we show how to compute most indicative torrent-counterexamples. We also discuss what information to present to the user: how to present witnesses and how to deal with overly large strongly connected components.

5.7.1 Maximizing Schedulers

The calculation of the maximal probability on a reachability problem can be performed by solving a linear minimization problem [BdA95, dA97]. This minimization problem is defined on a system of inequalities that has a variable x_i for each different state s_i and an inequality $\sum_j \pi(s_j) \cdot x_j \leq x_i$ for each distribution $\pi \in \tau(s_i)$. The maximizing (deterministic memoryless) scheduler η can be easily extracted out of such system of inequalities after obtaining the solution. If $p_0, \ldots, p_n$ are the values that minimize $\sum_i x_i$ in the previous system, then η is such that, for all s_i, $\eta(s_i) = \pi$ whenever $\sum_j \pi(s_j) \cdot p_j = p_i$. In the following we denote $\mathbb{P}_{s_i}[\Diamond\psi] \triangleq x_i$.

5.7.2 Computing most indicative torrent-counterexamples

We divide the computation of most indicative torrent-counterexamples to $\mathcal{M} \models_{\leq p} \Diamond\psi$ in three stages: *pre-processing*, SCC *analysis*, and *searching*.

Pre-processing stage. We first modify the original MC $\mathcal{D}$ by making all states in $\mathsf{Sat}(\psi) \cup S \setminus \mathsf{Sat}_{\Diamond}(\psi)$ absorbing. In this way we obtain the MC $\mathcal{D}_\psi$ from Definition 5.6.1. Note that we do not have to spend additional computational resources to compute this set, since $\mathsf{Sat}_{\Diamond}(\psi) = \{s \in S \mid \mathbb{P}_s[\Diamond\psi] > 0\}$ and hence all required data is already available from the LTL model checking phase.

SCC analysis stage. We remove all SCCs K of $\mathcal{D}_\psi$ keeping just *input states* of K, getting the acyclic MC $\mathsf{Ac}(\mathcal{D}_\psi)$ according to Definition 5.5.2.

To compute this, we first need to find the SCCs of $\mathcal{D}_\psi$. There exists several well known algorithms to achieve this: Kosaraju's, Tarjan's, Gabow's algorithms (among others). We also have to compute the reachability probability from input states to output states of every SCC. This can be done by using steady-state analysis techniques [Cas93].

Searching stage. To find most indicative torrent-counterexamples in $\mathcal{D}$, we find most indicative counterexamples in $\mathsf{Ac}(\mathcal{D}_\psi)$. For this we use the

same approach as [HK07a], turning the MC into a weighted digraph to replace the problem of finding the finite path with highest probability by a shortest path problem. The nodes of the digraph are the states of the MC and there is an edge between s and t if $\mathcal{P}(s,t) > 0$. The weight of such an edge is $-\log(\mathcal{P}(s,t))$.

Finding the most indicative counterexample in $\mathsf{Ac}(\mathcal{D}_\psi)$ is now reduced to finding k shortest paths. As explained in [HK07a], our algorithm has to compute k on the fly. Eppstein's algorithm [Epp98] produces the k shortest paths in general in $O(m+n\log n+k)$, where m is the number of nodes and n the number of edges. In our case, since $\mathsf{Ac}(\mathcal{D}_\psi)$ is acyclic, the complexity decreases to $O(m+k)$.

5.7.3 Debugging issues

Representative finite paths. What we have computed so far is a most indicative counterexample to $\mathsf{Ac}(\mathcal{D}_\psi) \models_{\leq p} \Diamond\psi$. This is a finite set of rails, i.e., a finite set of paths in $\mathsf{Ac}(\mathcal{D}_\psi)$. Each of these paths σ represents a witness $\mathsf{TorrGen}(\mathcal{D},\sigma)$. Note that this witness itself has usually infinitely many elements.

In practice, one has to display a witness to the user. The obvious way would be to show the user the rail σ. This, however, may be confusing to the user as σ is not a finite path of the original Markov Decision Process. Instead of presenting the user with σ, we therefore show the user the finite path of $\mathsf{TorrGen}(\mathcal{D},\sigma)$ with highest probability.

Definition 5.7.1. Let $\mathcal{D}$ be an MC, and $\sigma \in \mathsf{Paths}^\star(\mathsf{Ac}(\mathcal{D}_\psi))$ a rail of $\mathcal{D}$. We define the *representant of* $\mathsf{Torr}(\mathcal{D},\sigma)$ as

$$\mathsf{repTorr}\,(\mathcal{D},\sigma) = \mathsf{repTorr}\left(\biguplus_{\rho\in\mathsf{TorrGen}(\mathcal{D},\sigma)} \langle\rho\rangle\right) \triangleq \mathsf{arg}\max_{\rho\in\mathsf{TorrGen}(\mathcal{D},\sigma)} \mathbb{P}(\langle\rho\rangle)$$

Note that given $\mathsf{repTorr}\,(\mathcal{D},\sigma)$ one can easily recover σ. Therefore, no information is lost by presenting torrents as one of its generators instead of as a rail.

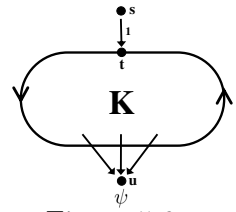


Figure 5.6:

Expanding SCC. Note that in the Preprocessing stage, we reduced the size of many SCCs of the system (and likely even completely removed some) by making states in $\mathsf{Sat}(\psi) \cup S \setminus \mathsf{Sat}_{\Diamond}(\psi)$ absorbing. However, It is possible that the system still contains some very large strongly connected components. In that case, a single witness could have a very large probability mass and one could argue that the information presented to the user is not detailed enough. For instance, consider the Markov Chain of Figure 5.6 in which there is a single large SCC with input state t and output state u.

The most indicative torrent-counterexample to the property $\mathcal{D} \models_{\leq 0.9} \Diamond\psi$ is simply $\{\mathsf{TorrGen}(stu)\}$, i.e., a single witness with probability mass 1 associated to the rail stu. Although this may seem uninformative, we argue that it is more informative than listing several paths of the form $st \cdots u$ with probability summing up to, say, 0.91. Our single witness counterexample suggests that the outgoing transition to a state not reaching ψ was simply forgotten in the design; the listing of paths still allows the possibility that one of the probabilities in the whole system is simply wrong.

Nevertheless, if the user needs more information to tackle bugs inside SCCs, note that there is more information available at this point. In particular, for every strongly connected component K, every input state s of K (even for every state in K), and every output state t of K, the probability of reaching t from s is already available from the computation of $\mathsf{Ac}(\mathcal{D}_{\psi})$ during the SCC analysis stage of Section 5.7.2.

5.8 Related Work

Recently, some work has been done on counterexample generation techniques for different variants of probabilistic models (Discrete Markov Chains and Continue Markov Chains) [AHL05, AL06, HK07a, HK07b]. In our terminology, these works consider witnesses consisting of a *single* finite path. We have already discussed in the Introduction that the single path approach does not meet the properties of accuracy, originality, significance,

and finiteness.

Instead, our witness/torrent approach provides a high level of abstraction of a counterexample. By grouping together finite paths that behave the same outside strongly connected components in a single witness, we can achieve these properties to a higher extent. Behaving the same outside strongly connected components is a reasonable way of formalizing the concept of providing *similar* debugging information. This grouping also makes witnesses significantly different from each other: each witness comes from a different rail and each rail provides a different way to reach the undesired property. Then each witness provides *original* information. Of course, our witnesses are more *significant* than single finite paths, because they are sets of finite paths. This also gives us more *accuracy* than the approach with single finite paths, as a collection of finite paths behaving the same and reaching an undesired condition with high probability is more likely to show how the system reaches this condition than just a single path. Finally, because there is a finite number of rails, there is also a *finite* number of witnesses.

Another key difference of our work with respect to previous ones is that our technique allows us to generate counterexamples for probabilistic systems *with* nondeterminism. However, an independent and concurrent study of counterexample generation for MDPs was carried out by Aljazzar and Leue [AL09]. There, the authors consider generating counterexamples for a fragment of pCTL, namely upper bounded formulas without nested temporal operators. The authors present three methods for generating counterexamples and study conditions under which these methods are suitable.

More recently, Schmalz et al. also investigated quantitative counterexample generation for LTL formulas [SVV09]. In qualitative probabilistic model checking, a counterexample is presented as a pair (α, γ), where α and γ are finite words such that all paths that extend α and have infinitely many occurrences of γ violate the property under consideration. In quantitative probabilistic model checking, a counterexample is presented as a pair (W, R), where W is a set of such finite words α and R is a set of such finite words γ.

Similar SCC reduction techniques to the one presented in this paper have been studied for different purposes. In [lGM02], the authors focus on the problem of software testing. They use Markov chains to model software behaviour and SCC analysis to decompose the state space of large Markov chains. More recently, Ábrahám et al. presented a model checker for Markov chains based on the detection and abstraction of strongly connected components [ÁJW+10]. Their algorithm has the advantage of offering abstract counterexamples, which can be interactively refined by the user.

Finally, the problem of presenting counterexamples as single paths has also been observed by Han, Katoen, and Damman [DHK08, HKD09]. There, the authors propose to use regular expressions to group paths together. Thus, in the same way that we group together paths behaving the same outside SCC, they group together paths associated to the same regular expression.

For a more extensive survey on quantitative counterexample generation for (both discrete and continuous time) Markov chains we refer the reader to chapters 3, 4, and 5 of [Han09].

Chapter 6

Interactive Systems and Equivalences for Security

In this overview chapter we briefly discuss extensions to the frameworks presented in Chapters 3 and 4[1]*. First, we consider the case in which secrets and observables interact (in contrast with the situation in Chapter 3), and show that it is still possible to define an information-theoretic notion of leakage, provided that we consider a more complex notion of channel, known in literature as* channel with memory and feedback*. Second, we extend the systems proposed in Chapter 4 by allowing nondeterminism also internally to the components. Correspondingly, we define a richer notion of admissible scheduler suitable and we use it for defining notion of process equivalences relating to nondeterminism in a more flexible way than the standard ones in the literature. In particular, we use these equivalences for defining notions of anonymity robust with respect to implementation refinement.*

[1]For more information about the topics discussed in this chapter we refer the reader to [AAP10a, AAP11, AAP10b, AAPvR10].

6.1 Interactive Information Flow

In this section we discuss the applicability of the information-theoretic approach to interactive systems. These systems were already considered in [DJGP02]. In that paper the authors proposed to define the matrix elements $\mathbb{P}(b \mid a)$ as the measure of the traces with (secret, observable)-projection (a, b), divided by the measure of the trace with secret projection a. This follows the definition of conditional probability in terms of joint and marginal probability. However, this approach does not lead to an information-theoretic channel. In fact, (by definition) a channel should be invariant with respect to the input distribution and such construction is not (as shown by Example 3.7.3).

In [AAP10a] and more recently in [AAP11], we consider an extension of the theory of channels which makes the information-theoretic approach applicable also the case of interactive systems. It turns out that a richer notion of channel, known in Information Theory as *channels with memory and feedback*, serves our purposes. The dependence of inputs on previous outputs corresponds to feedback, and the dependence of outputs on previous inputs and outputs corresponds to memory.

Let us explain more in detail the difference with the classical approach. In non-interactive systems, since the secrets always precede the observables, it is possible to group the sequence of secrets (and observables) in a single secret (respectively. observable) string. If we consider only one activation of the system, or if each use of the system is independent from the other, then we can model it as a discrete classical channel (memoryless, and without feedback) from a single input string to a single output string. When we have interactive systems, however, inputs and outputs may interleave and influence each other. Considering some sort of feedback in the channel is a way to capture this richer behavior. Secrets have a causal influence on observables via the channel, and, in the presence of interactivity, observables have a causal influence on secrets via the feedback. This alternating mutual influence between inputs and outputs can be modeled by repeated uses of the channels. However, each time the channel is used it represents a different state of the computation, and the conditional probabilities of

observables on secrets can depend on this state. The addition of memory to the model allows expressing the dependency of the channel matrix on such a state (which, as we will see, can also be represented by the history of inputs and outputs).

Recent results in Information Theory [TM09] have shown that, in channels with memory and feedback, the transmission rate does not correspond to the maximum mutual information (capacity), but rather to the maximum of the so-called *directed information*. Intuitively, this is due to the fact that mutual information expresses the correlation between the input and the output, and therefore it includes feedback. However, the feedback, i.e the way the output influences the next input, should not be considered part of the information transmitted. Directed information is essentially mutual information *minus* the dependence of the next input on previous output. We propose to adopt directed information and the corresponding notion of directed capacity to represent leakage.

Our extension is a generalization of the classical model, in the sense that it can represent both interactive and non-interactive systems. One important feature of the classical approach is that the choice of secrets is seen as external to the system, i.e. determined by the environment. This implies that the probability distribution on the secrets (input distribution) constitutes the a priori knowledge and does not count as leakage. In order to encompass the classical approach, in our extended model we should preserve this principle, and the most natural way is to consider the secret choices, at every stage of the computation, as external. Their probability distributions, which are now in general conditional probability distributions (depending on the history of secrets and observables) should be considered as part of the external knowledge, and should not be counted as leakage.

A second contribution of [AAP10a] and [AAP11] is the proof that the channel capacity is a continuous function of the Kantorovich metric on interactive systems. This was pointed out also in [DJGP02], however their construction does not work in our case due to the fact (as far as we understand) it assumes that the probability of a secret action (in any point of the computation) is different from 0. This assumption is not guaranteed in our case and therefore we had to come out with a different reasoning. The

fact that our proof does not need this assumption shows that the intuition of [DJGP02] concerning the continuity of capacity is valid in general.

6.1.1 Applications

Interactive systems can be found in a variety of disparate areas such as game theory, auction protocols, and zero-knowledge proofs. We now present two examples of interactive systems.

- In the area of auction protocols, consider the cocaine auction protocol [SA99]. The auction is organized as a succession of rounds of bidding. Round i starts with the seller announcing the bid price b_i for that round. Buyers have t seconds to make an offer (i.e. to say *yes*, meaning "I am willing to buy at the current bid price b_i"). As soon as one buyer says *yes*, he becomes the winner w_i of that round and a new round begins. If nobody says anything for t seconds, round i is concluded by timeout and the auction is won by the winner w_{i-1} of the previous round. The identities of the buyers in each round constitute the input of the channel, whereas the bid prices constitute the output of the channel. Note that inputs and outputs alternate so the system is interactive. It is also easy to see that inputs depend on past outputs (feedback): the identity of the winner of each round depends on the previous bid prices. Furthermore, outputs depend on the previous inputs (memory): (in some scenarios) the bid price of round i may depend on the identity of previous winners. For more details on the modeling of this protocol using channels with memory and feedback see [AAP11].
- In the area of game theory, consider the classic prisoner's dilemma (the present formulation is due to Albert W. Tucker [Pou92], but it was originally devised by Merrill Flood and Melvin Dresher in 1950). Two suspects are arrested by the police. The police have insufficient evidence for a conviction, and, having separated both prisoners, visit each of them to offer the same deal. If one testifies (defects from the other) for the prosecution against the other and the other remains

silent (cooperates with the other), the betrayer goes free and the silent accomplice receives the full 10-year sentence. If both remain silent, both prisoners are sentenced to only six months in jail for a minor charge. If each betrays the other, each receives a five-year sentence. Each prisoner must choose to betray the other or to remain silent. Each one is assured that the other would not know about the betrayal before the end of the investigation. In the iterated prisoner's dilemma, the game is played repeatedly. Thus each player has an opportunity to punish the other player for previous non-cooperative play. In this case the strategy (cooperate or defect) of each player is the input of the channel and the sentence is the output. Once again, it is easy to see that the system is interactive: inputs and outputs alternate. Furthermore, inputs depend on previous outputs (the strategy depend on the past sentences) and outputs depend on previous inputs (the sentence of the suspects depend on their declarations - cooperate or defect).

6.2 Nondeterminism and Information Flow

The *noise* of channel matrices, i.e. the similarity between the rows of the channel matrix, helps preventing the inference of the secret from the observables. In practice noise is created by using randomization, see for instance the DCNet [Cha88] and the Crowds [RR98] protocols.

In the literature about the foundations of Computer Security, however, the quantitative aspects are often abstracted away, and probabilistic behavior is replaced by nondeterministic behavior. Correspondingly, there have been various approaches in which information-hiding properties are expressed in terms of equivalences based on nondeterminism, especially in a concurrent setting. For instance, [SS96] defines *anonymity* as follows[2]: A protocol S is anonymous if, for every pair of culprits a and b, $S[^a/_x]$ and $S[^b/_x]$ produce the same observable traces. A similar definition is given in [AG99] for *secrecy*, with the difference that $S[^a/_x]$ and $S[^b/_x]$ are required to

[2]The actual definition of [SS96] is more complicated, but the spirit is the same.

be bisimilar. In [DKR09], an electoral system S preserves the *confidentiality of the vote* if for any voters v and w, the observable behavior of S is the same if we swap the votes of v and w. Namely, $S[^a/_v \mid^b /_w] \sim S[^b/_v \mid^a /_w]$, where $\sim$ represents bisimilarity.

These proposals are based on the implicit assumption that *all the nondeterministic executions present in the specification of S will always be possible under every implementation of S*. Or at least, that the adversary will believe so. In concurrency, however, as argued in [CNP09], nondeterminism has a rather different meaning: if a specification S contains some nondeterministic alternatives, typically it is because we want to abstract from specific implementations, such as the scheduling policy. A specification is considered correct, with respect to some property, if every alternative satisfies the property. Correspondingly, an implementation is considered correct if all executions are among those possible in the specification, i.e. if the implementation is a refinement of the specification. There is no expectation that the implementation will actually make possible all the alternatives indicated by the specification.

We argue that the use of nondeterminism in concurrency corresponds to a *demonic* view: the scheduler, i.e. the entity that will decide which alternative to select, may try to choose the worst alternative. Hence we need to make sure that "all alternatives are good", i.e. satisfy the intended property. In the above mentioned approaches to the formalization of security properties, on the contrary, the interpretation of nondeterminism is *angelic*: the scheduler is expected to actually help the protocol to confuse the adversary and thus protect the secret information.

There is another issue, orthogonal to the angelic/demonic dichotomy, but relevant for the achievement of security properties: the scheduler *should not be able to make its choices dependent on the secret*, or else nearly every protocol would be insecure, i.e. the scheduler would always be able to leak the secret to an external observer (for instance by producing different interleavings of the observables, depending on the secret). This remark has been made several times already, and several approaches have been proposed to cope with the problem of the "almighty" scheduler (aka omniscient, clairvoyant, etc.) [CCK+06a, GD07, CNP09, APvRS11, CP10, GvRS07].

The risk of a naive use of nondeterminism to specify a security property, is not only that it may rely on an implicit assumption that the scheduler behaves angelically, but also that it is clairvoyant, i.e. that it peeks at the secrets (that it is not supposed to be able to see) to achieve its angelic strategy.

Consider the following system, in a CCS-like syntax:

$$S \stackrel{\text{def}}{=} (c, out)(A \parallel Corr \parallel H_1 \parallel H_2),$$

with $A \stackrel{\text{def}}{=} \overline{c}\langle sec\rangle$,$Corr \stackrel{\text{def}}{=} c(s).\overline{out}\langle s\rangle$, $H_1 \stackrel{\text{def}}{=} c(s).\overline{out}\langle a\rangle$, $H_2 \stackrel{\text{def}}{=} c(s).\overline{out}\langle b\rangle$ and where $\parallel$ is the parallel operator, $\overline{c}\langle sec\rangle$ is a process that sends sec on channel c, $c(s).P$ is a process that receives s on channel c and then continues as P, and (c, out) is the restriction operator, enforcing synchronization on c and out. In this example, sec represents a secret information.

It is easy to see that we have $S\left[{}^{a}/_{sec}\right] \sim S\left[{}^{b}/_{sec}\right]$. Note that, in order to simulate the third branch in $S\left[{}^{a}/_{sec}\right]$, the process $S\left[{}^{b}/_{sec}\right]$ needs to select its first branch. Viceversa, in order to simulate the third branch in $S\left[{}^{b}/_{sec}\right]$, the process $S\left[{}^{a}/_{sec}\right]$ needs to select its second branch. This means that, in order to achieve bisimulation, the scheduler needs to know the secret, and change its choice accordingly.

This example shows a system that intuitively is not secure, because the third component, $Corr$, reveals whatever secret it receives. However, according to the equivalence-based notions of security discussed above, *it is secure*. But it is secure thanks to a scheduler that angelically helps the system to protect the secret, and it does so by making its choices dependent on the secret! In our opinion these assumptions on the scheduler are excessively strong.

In a recent work [AAPvR10] we address the above issue by defining a framework in which it is possible to combine both angelic and demonic nondeterminism in a setting in which also probabilistic behavior may be present, and in a context in which the scheduler is restricted (i.e. not clairvoyant). We propose safe versions of typical equivalence relations (traces and bisimulation), and we show how to use them to characterize information-hiding properties.

Chapter 7

Conclusion

In this chapter we summarize the main contributions of this thesis and discuss further directions.

7.1 Contributions

The goal of this thesis is to develop a formal framework for specifying, analyzing and verifying anonymity protocols and, more in general, information hiding protocols.

As discussed in the Introduction, conditional probabilities are a key concept in assessing the degree of information protection. In Chapter 2, we have extended the probabilistic temporal logic pCTL to cpCTL, in which it is possible to express conditional probabilities. We have also proved that optimal scheduling decisions can always be reached by a deterministic and semi history-independent scheduler. This fundamental result, allowed us to define an algorithm to verify cpCTL formulas. Our algorithm first reduces the MDP to an acyclic MDP and then computes optimal conditional probabilities in the acyclic MDP. In addition, we have defined a notion of counterexample for conditional formulas and sketched an algorithm for counterexample generation.

We then turned our attention to more practical grounds. In Chapter

3, we have addressed the problem of computing the information leakage of a system in an efficient way. We have proposed two methods: one based on reachability techniques and the other based on quantitative counterexample generation. In addition, we have shown that when the automaton is interactive it is not possible to define its channel in the standard way. An intriguing problem is how to extend the notion of channel so to capture the dynamic nature of interaction. In Chapter 6 we have briefly discussed how to solve this problem by using more complex information theoretic channels, namely channels with history and feedback.

In Chapter 4, we have attacked a well known problem of concurrent information-hiding protocols, namely full-information scheduling. To overcome this problem, we have defined a class of partial-information schedulers which can only base their decisions on the information that they have available. In particular they cannot base their decisions on the internal behavior of the components. We have used admissible schedulers to resolve nondeterminism in a realistic way, and to revise some anonymity definitions from the literature. In addition, we have presented a technique to prove the various definitions of anonymity proposed in the chapter. This is particularly interesting considering that many problems related to restricted schedulers have been shown to be undecidable. We have illustrated the applicability of our proof technique by proving that the well-known DC protocol is anonymous when considering admissible schedulers, in contrast to the situation when considering full-information schedulers.

The last major contribution of this thesis is a novel technique for representing and computing counterexamples for nondeterministic and probabilistic systems. In Chapter 5, we have shown how to carefully partition a counterexample in sets of paths. These sets are intended to provide information related to the violation of the property under consideration, so we call them *witnesses*. Five properties that witnesses should satisfy (in order to provide significant debugging information) are identified in this chapter. The key contribution of this chapter is a technique based on strongly connected component analysis that makes it possible to partition counterexamples into witnesses satisfying the desired properties.

7.2 Further directions

There are several ways of extending the work presented in this thesis.

As we have shown in Chapter 2, the most important issue when computing conditional probabilities is that optimizing schedulers are not determined by the local structure of the system. As a consequence, it is not possible to reduce the problem of verifying cpCTL to a linear optimization problem (as it is the case with pCTL). A natural question arising from this observation, is whether the problem of model checking conditional probabilities is inherently exponential or not. We believe that it is; however we are of the idea that it is also possible to find suitable restrictions (either to the formulas or to the systems under consideration) that would make it possible to model check conditional probabilities in polynomial time.

In a more practical matter, counterexample generation for probabilistic model checking is nowadays a very hot topic for which several applications in the most diverse areas have been identified. During the last few years, many techniques have been proposed for different flavours of logics and models. However, to the best of our knowledge, no practical tool to automatically generate quantitative counterexamples has been implemented. We believe that such a practical tool could be a significant contribution to the field. More concretely, we believe that a tool implementing the regular-expression and k-shortest path techniques introduced by Han et al. in combination with the SCC analysis techniques presented in this thesis would be of great value.

In Chapter 2, we have made a connection between quantitative counterexample generation and information leakage computation. Thanks to this connection, such a tool would also allow us to compute / approximate leakage of large scale protocols. Furthermore, it would make it possible to investigate in more depth how the debugging information provided by the tool can be used to identify flaws of the protocol causing high leakage.

Finally, as for most definitions of partial-information schedulers from the literature, our notions of admissible schedulers may raise undecidability issues. Thus, it would be interesting to investigate whether the notions of anonymity proposed in Chapter 4 are actually verifiable (remember that

the proof technique we proposed is sufficient but not necessary). Another interesting direction for future work is to adapt well known isomorphism-checking algorithms and tolls to our setting in order to automatically verify some anonymity properties.

Bibliography

[AAP10a] Mário S. Alvim, Miguel E. Andrés, and Catuscia Palamidessi. Information flow in interactive systems. In *Proceedings of CONCUR*, volume 6269 of LNCS, pages 102–116. Springer, 2010. **Cited** on pages 19, 20, 159, 160 and 161.

[AAP10b] Mário S. Alvim, Miguel E. Andrés, and Catuscia Palamidessi. Probabilistic information flow. In *Proceedings of LICS*, pages 314–321. IEEE Computer Society, 2010. **Cited** on pages 19, 20 and 159.

[AAP11] Mário S. Alvim, Miguel E. Andrés, and Catuscia Palamidessi. Information Flow in Interactive Systems. *Journal of Computer Security*, 2011. To appear. **Cited** on pages 19, 20, 159, 160, 161 and 162.

[AAPvR10] Mário S. Alvim, Miguel E. Andrés, Catuscia Palamidessi, and Peter van Rossum. Safe equivalences for security properties. In *Proceedings of IFIP TCS*, volume 323 of IFIP, pages 55–70. Springer, 2010. **Cited** on pages 19, 21, 105, 128, 159 and 165.

[ADvR08] Miguel E. Andrés, Pedro R. D'Argenio, and Peter van Rossum. Significant diagnostic counterexamples in probabilistic model checking. In *Proceedings of Haifa Verification Conference*, volume 5394 of LNCS, pages 129–148. Springer, 2008. **Cited** on pages 20, 64, 66, 67, 84 and 85.

[AG99] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Inf. and Comp.*, 148(1):1–70, 1999. **Cited** on page 163.

[AHL05] Husain Aljazzar, Holger Hermanns, and Stefan Leue. Counterexamples for timed probabilistic reachability. In *Proceedings of FORMATS*, volume 3829, pages 177–195, 2005. **Cited** on pages 130 and 156.

[ÁJW+10] Erika Ábrahám, Nils Jansen, Ralf Wimmer, Joost-Pieter Katoen, and Bernd Becker. Dtmc model checking by scc reduction. In *Proceedings of QEST*, pages 37–46. IEEE, 2010. **Cited** on page 158.

[AL06] Husain Aljazzar and Stefan Leue. Extended directed search for probabilistic timed reachability. In *Proceedings of FORMATS*, volume 4202 of *LNCS*, pages 33–51, 2006. **Cited** on pages 66, 130 and 156.

[AL08] Husain Aljazzar and Stefan Leue. Debugging of dependability models using interactive visualization of counterexamples. In *Proceedings of QEST*, pages 189–198. IEEE, 2008. **Cited** on page 85.

[AL09] Husain Aljazzar and Stefan Leue. Generation of counterexamples for model checking of markov decision processes. In *In Proceedings of QEST*, IEEE Computer Society, pages 197–206, December 2009. **Cited** on pages 130 and 157.

[And06] Miguel E. Andrés. Derivation of counterexamples for quantitative model checking. Master's thesis, supervised by Pedro R. D'Argenio. Universidad Nacional de Córdoba, September 2006. **Cited** on page 130.

[APvRS10a] Miguel E. Andrés, Catuscia Palamidessi, Peter van Rossum, and Geoffrey Smith. Computing the leakage of information-

hiding systems. In *Proceedings of TACAS*, volume 6015 of LNCS, pages 373–389. Springer, 2010. **Cited** on page 20.

[APvRS10b] Miguel E. Andrés, Catuscia Palamidessi, Peter van Rossum, and Ana Sokolova. Information hiding in probabilistic concurrent systems. In *Proceedings of QEST*, pages 17–26. IEEE Computer Society, 2010. **Cited** on page 20.

[APvRS11] Miguel E. Andrés, Catuscia Palamidessi, Peter van Rossum, and Ana Sokolova. Information Hiding in Probabilistic Concurrent Systems. *Journal of Theoretical Computer Science*, 412:3072–3089, 2011. **Cited** on pages 20 and 164.

[AvR08] Miguel E. Andrés and Peter van Rossum. Conditional probabilities over probabilistic and nondeterministic systems. In *Proceedings of TACAS*, volume 4963 of LNCS, pages 157–172. Springer, 2008. **Cited** on page 19.

[BCP08] Christelle Braun, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Compositional methods for information-hiding. In *Proc. of FOSSACS*, volume 4962 of *LNCS*, pages 443–457. Springer, 2008. **Cited** on page 96.

[BCP09] Christelle Braun, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Quantitative notions of leakage for one-try attacks. In *Proceedings of MFPS*, volume 249 of *ENTCS*, pages 75–91. Elsevier B.V., 2009. **Cited** on pages 70, 74, 86, 88, 97, 98, 100 and 115.

[BCPP08] Romain Beauxis, Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Prakash Panangaden. Formal approaches to information-hiding (tutorial). In *Proc. of TGC*, volume 4912 of *LNCS*, pages 347–362. Springer, 2008. **Cited** on page 13.

[BdA95] Andrea Bianco and Luca de Alfaro. Model checking of probabilistic and nondeterministic systems. In *Proceedings of*

FSTTCS, volume 1026, pages 499–513, 1995. **Cited** on pages 24, 26, 28, 30, 33, 37, 134, 135, 139, 140 and 154.

[Bel57] Richard E. Bellman. A Markovian decision process. *J. Math. Mech.*, 6:679–684, 1957. **Cited** on pages 26 and 134.

[BLR05] Gerd Behrmann, Kim G. Larsen, and Jacob I. Rasmussen. Optimal scheduling using priced timed automata. *SIGMETRICS Perform. Eval. Rev.*, 32(4):34–40, 2005. **Cited** on page 130.

[Bor06] Michele Boreale. Quantifying information leakage in process calculi. In *Proceedings of ICALP*, volume 4052 of *LNCS*, pages 119–131. Springer, 2006. **Cited** on page 16.

[BP05] Mohit Bhargava and Catuscia Palamidessi. Probabilistic anonymity. In Martín Abadi and Luca de Alfaro, editors, *CONCUR*, volume 3653 of *LNCS*, pages 171–185, 2005. **Cited** on pages 13, 25, 96, 97 and 118.

[Cas93] Christos G. Cassandras. *Discrete Event Systems: Modeling and Performance Analysis*. Richard D. Irwin, Inc., and Aksen Associates, Inc., 1993. **Cited** on pages 65 and 154.

[CCK+06a] Ran Canetti, Ling Cheung, Dilsun Kaynar, Moses Liskov, Nancy Lynch, Olivier Pereira, and Roberto Segala. Task-structured probabilistic i/o automata. In *Proc. of WODES*, 2006. **Cited** on pages 97, 127 and 164.

[CCK+06b] Ran Canetti, Ling Cheung, Dilsun Kirli Kaynar, Moses Liskov, Nancy A. Lynch, Olivier Pereira, and Roberto Segala. Time-bounded task-PIOAs: A framework for analyzing security protocols. In *Proc. of DISC*, volume 4167 of *LNCS*, pages 238–253. Springer, 2006. **Cited** on pages 97 and 127.

[CGJ+00] Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement. In *Proceedings of CAV*, pages 154–169, 2000. **Cited** on pages 66 and 130.

[Cha88] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75, 1988. **Cited** on pages 13, 25, 96, 101, 102 and 163.

[CHM01] David Clark, Sebastian Hunt, and Pasquale Malacaria. Quantitative analysis of the leakage of confidential data. In *Proceedings of QAPL*, volume 59 (3) of *ENTCS*, pages 238–251. Elsevier Science B.V., 2001. **Cited** on page 16.

[CHM05a] David Clark, Sebastian Hunt, and Pasquale Malacaria. Quantified interference for a while language. In *Proceedings of QAPL*, volume 112 of *ENTCS*, pages 149–166. Elsevier Science B.V., 2005. **Cited** on pages 16 and 96.

[CHM05b] David Clark, Sebastian Hunt, and Pasquale Malacaria. Quantitative information flow, relations and polymorphic types. *J. of Logic and Computation*, 18(2):181–199, 2005. **Cited** on pages 70 and 96.

[CL05] Jan Camenisch and Anna Lysyanskaya. A formal treatment of onion routing. In *Proceedings of CRYPTO*, volume 3621 of LNCS, pages 169–187, 2005. **Cited** on page 25.

[Cla08] Edmund M. Clarke. The birth of model checking. In *25 Years of Model Checking*, pages 1–26, 2008. **Cited** on page 8.

[CMS09] Michael R. Clarkson, Andrew C. Myers, and Fred B. Schneider. Belief in information flow. *Journal of Computer Security*, 17(5):655–701, 2009. **Cited** on pages 16, 70 and 96.

[CNP09] Konstantinon Chatzikokolakis, Gethin Norman, and David Parker. Bisimulation for demonic schedulers. In *Proc. of FOSSACS*, volume 5504 of *LNCS*, pages 318–332. Springer, 2009. **Cited** on pages 97, 128 and 164.

[CP10] Konstantinos Chatzikokolakis and Catuscia Palamidessi. Making random choices invisible to the scheduler. *Information and Computation*, 208:694–715, 2010. **Cited** on pages 97, 128 and 164.

[CPP08a] Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Prakash Panangaden. Anonymity protocols as noisy channels. *Inf. and Comp.*, 206(2–4):378–401, 2008. **Cited** on pages 16, 69, 70, 92 and 96.

[CPP08b] Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Prakash Panangaden. On the Bayes risk in information-hiding protocols. *Journal of Computer Security*, 16(5):531–571, 2008. **Cited** on page 96.

[CSWH00] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Workshop on Design Issues in Anonymity and Unobservability*, volume 2009 of *LNCS*, pages 44–66. Springer, 2000. **Cited** on page 96.

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., second edition, 2006. **Cited** on pages 73, 98 and 100.

[dA97] Luca de Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, 1997. **Cited** on page 154.

[dAHJ01] Luca de Alfaro, Thomas A. Henzinger, and Ranjit Jhala. Compositional methods for probabilistic systems. In *Proc.*

of CONCUR, volume 2154 of *LNCS*. Springer, 2001. **Cited** on page 128.

[dAKM97] Luca de Alfaro, Arjun Kapur, and Zohar Manna. Hybrid diagrams: A deductive-algorithmic approach to hybrid system verification. In *Symposium on Theoretical Aspects of Computer Science*, pages 153–164, 1997. **Cited** on page 139.

[Daw05] Conrado Daws. Symbolic and parametric model checking of discrete-time markov chains. In *ICTAC*, volume 3407 of *LNCS*, pages 280–294. Springer, 2005. **Cited** on page 82.

[DHK08] Berteun Damman, Tingting Han, and Joost-Pieter Katoen. Regular expressions for PCTL counterexamples. In *Proceedings of QEST*, pages 179–188. IEEE, 2008. **Cited** on pages 82, 85 and 158.

[DJGP02] Josee Desharnais, Radha Jagadeesan, Vineet Gupta, and Prakash Panangaden. The metric analogue of weak bisimulation for probabilistic processes. In *Proc. of LICS*, pages 413–422. IEEE, 2002. **Cited** on pages 71, 90, 160, 161 and 162.

[DKR09] Stéphanie Delaune, Steve Kremer, and Mark Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, 2009. **Cited** on page 164.

[DPW06] Yuxin Deng, Jun Pang, and Peng Wu. Measuring anonymity with relative entropy. In *Proceedings of FAST*, volume 4691 of *LNCS*, pages 65–79. Springer, 2006. **Cited** on page 70.

[DY83] Danny Dolev and Andrew C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983. **Cited** on page 26.

[Epp98] David Eppstein. Finding the k shortest paths. In *SIAM Journal of Computing*, pages 652–673, 1998. **Cited** on page 155.

[Feh02] Ansgar Fehnker. *Citius, Vilius, Melius - Guiding and Cost-Optimality in Model Checking of Timed and Hybrid Systems*. PhD thesis, KUNijmegen, 2002. **Cited** on page 130.

[FOO92] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In *Proceedings of Advances in Cryptology (AUSCRYPT '92)*, volume 718, pages 244–251, 1992. **Cited** on page 25.

[FV97] J. Filar and K. Vrieze. *Competitive Markov Decision Processes*. 1997. **Cited** on pages 26 and 134.

[GD07] Sergio Giro and Pedro R. D'Argenio. Quantitative model checking revisited: Neither decidable nor approximable. In *FORMATS*, volume 4763 of *LNCS*, pages 179–194. Springer, 2007. **Cited** on pages 121, 128 and 164.

[GHvRP05] Flavio D. Garcia, Ichiro Hasuo, Peter van Rossum, and Wolter Pieters. Provable anonymity. In *FMSE*, pages 63–72. ACM, 2005. **Cited** on pages 98 and 121.

[Gir09] Sergio Giro. Undecidability results for distributed probabilistic systems. In *SBMF*, volume 5902 of *LNCS*, pages 220–235. Springer, 2009. **Cited** on pages 121 and 128.

[Gra91] J. W. Gray, III. Toward a mathematical foundation for information flow security. In *SSP*, pages 21–35, Washington - Brussels - Tokyo, May 1991. IEEE. **Cited** on page 16.

[GvRS07] Flavio D. Garcia, Peter van Rossum, and Ana Sokolova. Probabilistic anonymity and admissible schedulers, 2007. arXiv:0706.1019v1. **Cited** on page 164.

[Han09] Tingting Han. Diagnosis, synthesis and analysis of probabilistic models. *PhD Thesis*, 2009. **Cited** on page 158.

[HJ89] H. Hansson and B. Jonsson. A framework for reasoning about time and reliability. In *Proceedings of Real Time Systems Symposium*, pages 102–111, 1989. **Cited** on page 24.

[HJ94] Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *Formal Asp. Comput.*, 6(5):512–535, 1994. **Cited** on page 17.

[HK07a] Tingting Han and Joost-Pieter Katoen. Counterexamples in probabilistic model checking. In *Proceedings of TACAS*, volume 4424, pages 60–75, 2007. **Cited** on pages 66, 130, 133, 141, 153, 155 and 156.

[HK07b] Tingting Han and Joost-Pieter Katoen. Providing evidence of likely being on time: Counterexample generation for ctmc model checking. In *Proceedings of ATVA*, volume 4762, pages 331–346, 2007. **Cited** on pages 130 and 156.

[HK07c] Ichiro Hasuo and Yoshinobu Kawabe. Probabilistic anonymity via coalgebraic simulations. In *Proceedings of the European Symposium on Programming*, volume 4421 of *LNCS*, pages 379–394. Springer, 2007. **Cited** on pages 98 and 121.

[HKD09] Tingting Han, Joost-Pieter Katoen, and Berteun Damman. Counterexample generation in probabilistic model checking. *IEEE Transactions on Software Engineering*, 35(2):241–257, 2009. **Cited** on page 158.

[HO05] Joseph Y. Halpern and Kevin R. O'Neill. Anonymity and information hiding in multiagent systems. *Journal of Computer Security*, 13(3):483–512, 2005. **Cited** on pages 11, 13, 96, 98 and 121.

[HS04] Dominic Hughes and Vitaly Shmatikov. Information hiding, anonymity and privacy: a modular approach. *Journal of Computer Security*, 12(1):3–36, 2004. **Cited** on page 11.

[HSP10] Sardaouna Hamadou, Vladimiro Sassone, and Catuscia Palamidessi. Reconciling belief and vulnerability in information flow. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 79–92. IEEE Comput. Soc. Press, 2010. **Cited** on page 17.

[JFL] Jflap website. http://www.jflap.org/. **Cited** on page 82.

[KB07] Boris Köpf and David A. Basin. An information-theoretic model for adaptive side-channel attacks. In *Proc. of CCS*, pages 286–296. ACM, 2007. **Cited** on page 16.

[KNP06] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Symmetry reduction for probabilistic model checking. In *Proceedings of CAV*, volume 4144 of *LNCS*, pages 234–248. Springer, 2006. **Cited** on page 128.

[LBB+01] Kim G. Larsen, Gerd Behrmann, Ed Brinksma, Ansgar Fehnker, Thomas S. Hune, Paul Petterson, and Judi Romijn. As Cheap as Possible: Efficient Cost-Optimal Reachability for Priced Timed Automata. In *Proceedings of CAV*, volume 2102 of LNCS, pages 493–505. Springer, 2001. **Cited** on page 66.

[lGM02] Hélène le Guen and Raymond A. Marie. Visiting probabilities in non-irreducible markov chains with strongly connected components. In *ESM*, pages 548–552, 2002. **Cited** on page 158.

[Low02] Gavin Lowe. Quantifying information flow. In *Proc. of CSFW 2002*, pages 18–31. IEEE, 2002. **Cited** on page 16.

[Mal07] Pasquale Malacaria. Assessing security threats of looping constructs. In *Proc. of POPL*, pages 225–235. ACM, 2007. **Cited** on page 96.

[MC08] Pasquale Malacaria and Han Chen. Lagrange multipliers and maximum information leakage in different observational models. In *Proc. of PLAS*, pages 135–146. ACM, 2008. **Cited** on page 96.

[McL90] John McLean. Security models and information flow. In *SSP'90*, pages 180–189. IEEE, 1990. **Cited** on page 16.

[Mil89] R. Milner. *Communication and Concurrency*. Int. Series in Computer Science. Prentice Hall, 1989. **Cited** on page 104.

[Mil99] Robin Milner. *Communicating and mobile systems: the π-calculus*. Cambridge University Press, 1999. **Cited** on page 104.

[MNCM03] Ira S. Moskowitz, Richard E. Newman, Daniel P. Crepeau, and Allen R. Miller. Covert channels and anonymizing networks. In *Proc. of PES*, pages 79–88. ACM, 2003. **Cited** on pages 16 and 70.

[MNS03] Ira S. Moskowitz, Richard E. Newman, and Paul F. Syverson. Quasi-anonymous channels. In *Proc. of CNIS*, pages 126–131. IASTED, 2003. **Cited** on page 16.

[MP91] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer, 1991. **Cited** on page 137.

[MVdV04] S. Mauw, J. Verschuren, and E.P. de Vink. A formalization of anonymity and onion routing. In *Proceedings of ESORICS*, volume 3193 of LNCS, pages 109–124, 2004. **Cited** on pages 98 and 121.

[Neu05] C. Neumann. Converting deterministic finite automata to regular expressions. 2005. http://neumannhaus.com/christoph/papers/2005-03-16.DFA_to_RegEx.pdf. **Cited** on page 82.

[Pou92] William Poundstone. *Prisoners Dilemma*. Doubleday NY, 1992. **Cited** on page 162.

[PRI] Prism website. http://www.prismmodelchecker.org. **Cited** on page 92.

[PZ93] Amir Pnueli and Lenore D. Zuck. Probabilistic verification. *Information and Computation*, 103(1):1–29, 1993. **Cited** on pages 28 and 135.

[Rén60] Alfréd Rényi. On Measures of Entropy and Information. In *Proceedings of the 4th Berkeley Symposium on Mathematics, Statistics, and Probability*, pages 547–561, 1960. **Cited** on page 16.

[RR98] Michael K. Reiter and Aviel D. Rubin. Crowds: anonymity for Web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998. **Cited** on pages 14, 25, 76, 96 and 163.

[RS01] Peter Y. Ryan and Steve Schneider. *Modelling and Analysis of Security Protocols*. Addison-Wesley, 2001. **Cited** on page 11.

[Rut00] Jan J.M.M. Rutten. Universal coalgebra: A theory of systems. *Theoretical Computer Science*, 249:3–80, 2000. **Cited** on page 121.

[SA99] Frank Stajano and Ross J. Anderson. The cocaine auction protocol: On the power of anonymous broadcast. In *Information Hiding*, pages 434–447, 1999. **Cited** on page 162.

[SdV04] Ana Sokolova and Erik P. de Vink. Probabilistic automata: System types, parallel composition and comparison. In *Validation of Stochastic Systems: A Guide to Current Research*, volume 2925 of LNCS, pages 1–43. 2004. **Cited** on pages 26 and 134.

[Seg95] Roberto Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, 1995. Tech. Rep. MIT/LCS/TR-676. **Cited** on pages 72 and 98.

[SGR97] P.F. Syverson, D.M. Goldschlag, and M.G. Reed. Anonymous connections and onion routing. In *IEEE Symposium on Security and Privacy*, pages 44–54, 1997. **Cited** on page 96.

[SL95] Roberto Segala and Nancy Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995. **Cited** on pages 26, 98, 103 and 134.

[SM03] Andrei Sabelfeld and Andrew C. Myers. Language-based information flow security. *IEEE Journal on Selected Areas in Communications*, 21(1):5–19, 2003. **Cited** on page 70.

[Smi09] Geoffrey Smith. On the foundations of quantitative information flow. In *Proc. of FOSSACS*, volume 5504 of *LNCS*, pages 288–302. Springer, 2009. **Cited** on pages 16, 70, 74, 86, 88, 96, 97, 98, 100 and 115.

[SS96] Steve Schneider and Abraham Sidiropoulos. CSP and anonymity. In *Proc. of ESORICS*, volume 1146 of *LNCS*, pages 198–218. Springer, 1996. **Cited** on pages 11 and 163.

[SS99] Paul F. Syverson and Stuart G. Stubblebine. Group principals and the formalization of anonymity. In *World Congress on Formal Methods (1)*, pages 814–833, 1999. **Cited** on page 11.

[SVV09] Matthias Schmalz, Daniele Varacca, and Hagen Völzer. Counterexamples in probabilistic ltl model checking for markov chains. In *Proceedings of CONCUR*, volume 5710 of *LNCS*, pages 587–602. Springer, 2009. **Cited** on page 157.

[TM09] Sekhar Tatikonda and Sanjoy K. Mitter. The capacity of channels with feedback. *IEEE Transactions on Information Theory*, 55(1):323–349, 2009. **Cited** on page 161.

[Var85] M.Y. Vardi. Automatic verification of probabilistic concurrent finite-state systems. In *Proc. 26th IEEE Symp. Found. Comp. Sci.*, pages 327–338, 1985. **Cited** on pages 28 and 135.

[ZB05] Ye Zhu and Riccardo Bettati. Anonymity vs. information leakage in anonymity systems. In *Proc. of ICDCS*, pages 514–524. IEEE, 2005. **Cited** on pages 16 and 96.

Samenvatting

Terwijl we het digitale tijdperk ingaan zijn er immer groeiende zorgen over de hoeveelheid digitale data die over ons verzameld wordt. Websites houden vaak het browse-gedrag van mensen bij, ziektenkostenverzekeraars verzamelen medische gegegevens en smartphones en navigatiesystemen versturen informatie die het mogelijk maakt de fysieke locatie van hun gebruikers te bepalen. Hierdoor staan anonimiteit, en privacy in het algemeen, steeds meer op het spel. Anonimiteitsprotocollen proberen iets tegen deze tendens te doen door anonieme communicatie over het Internet mogelijk te maken. Om de correctheid van dergelijke protocollen, die vaak extreem complex zijn, te garanderen, is een degelijk framework vereist waarin anonimiteitseigenschappen kunnen worden uitgedrukt en geanalyseerd. Formele methoden voorzien in een verzameling wiskundige technieken die het mogelijk maken anonimiteitseigenschappen rigoreus te specificeren en te verifiëren.

Dit proefschrift gaat over de grondslagen van formele methoden voor toepassingen in computerbeveiliging en in het bijzonder anonimiteit. Concreet, we ontwikkelen frameworks om anonimiteitseigenschappen te specificeren en algoritmen om ze te verifiëren. Omdat in de praktijk anonimiteitsprotocollen altijd *wat* informatie lekken, leggen we de focus op quantitatieve eigenschappen die de *mate* van gelekte informatie van een protocol beschrijven.

We beginnen het onderzoek naar anonimiteit vanuit de basis, namelijk voorwaardelijke kansen. Dit zijn de sleutelingrediënten van de meeste quantitatieve anonimiteitsprotocollen. In Hoofdstuk 2 prenteren we cpCTL,

de eerste temporele logica waarin voorwaardelijke kansen kunnen worden uitgedrukt. We presenteren ook een algoritme om cpCTL formules te verifiëren met een modelchecker. Samen met een modelchecker maakt deze logica het mogelijk om quantitatieve anomimiteitseigenschappen van complexe systemen waarin zowel probabilistisch als nondeterministisch gedrag voorkomt te specificeren en verifiëren.

Vervolgens gaan we meer de praktijk in: de constructie van algoritmen die de mate van het lekken van informatie meten. Om preciezer te zijn, Hoofdstuk 3 beschrijft polynomiale algoritmen om de (informatietheoretische) information leakage te quantificeren voor verscheidene soorten volledig probabilistische protocllen (d.w.z., protocollen zonder nondeterministisch gedrag). The technieken uit dit hoofdstuk zijn de eerste die het mogelijk maken de informatie leakage voor interactieve protocollen te berekenen.

In Hoofdstuk 4 behandelen we een bekend probleem in gedistribueerde anonimiteitsprotocollen, namelijk schedulers met volledige informatie. Om dit probleem op te lossen stellen we een alternatieve definitie van scheduler voor, samen met nieuwe definities voor anonomiteit (variërend met de capaciteiten van de aanvaller) en herzien de bekende definitie van sterke anonimiteit uit de literatuur. Bovendien laten we een techniek zien waarmee gecontroleerd kan worden of een gedistribueerd protocol aan enkele van deze definities voldoet.

In Hoofdstuk 5 laten we op tegenvoorbeelden gebaseerde technieken zien die het mogelijk maken complexe systemen te debuggen. Dit maakt het mogelijk fouten in security protocollen op te sporen. Tenslotte, in Hoofdstuk 6, beschrijven we kort uitbreidingen van de frameworks en technieken uit Hoofdstukken 3 en 4.

Index

Curriculum Vitae

1980 Born on 2 July, Río Cuarto, Argentina.

1994–1998 Private Institute Galileo Galilei (High School), Río Cuarto, Córdoba, Argentina.

1999–2006 Computer Science Licentiate (equivalent to MSc.), Faculty of Mathematics, Astronomy and Physics (Fa.M.A.F.). National University of Córdoba (UNC), Argentina.

2006–2010 PhD student in the Digital Security Group, Radboud University Nijmegen, The Netherlands.

2010– Postdoctoral researcher in the Comète Team, Laboratory of Informatics of the École Polytechnique (LIX), France.

Titles in the IPA Dissertation Series since 2005

E. Ábrahám. *An Assertional Proof System for Multithreaded Java -Theory and Tool Support-* . Faculty of Mathematics and Natural Sciences, UL. 2005-01

R. Ruimerman. *Modeling and Remodeling in Bone Tissue.* Faculty of Biomedical Engineering, TU/e. 2005-02

C.N. Chong. *Experiments in Rights Control - Expression and Enforcement.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-03

H. Gao. *Design and Verification of Lock-free Parallel Algorithms.* Faculty of Mathematics and Computing Sciences, RUG. 2005-04

H.M.A. van Beek. *Specification and Analysis of Internet Applications.* Faculty of Mathematics and Computer Science, TU/e. 2005-05

M.T. Ionita. *Scenario-Based System Architecting - A Systematic Approach to Developing Future-Proof System Architectures.* Faculty of Mathematics and Computing Sciences, TU/e. 2005-06

G. Lenzini. *Integration of Analysis Techniques in Security and Fault-Tolerance.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-07

I. Kurtev. *Adaptability of Model Transformations.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-08

T. Wolle. *Computational Aspects of Treewidth - Lower Bounds and Network Reliability.* Faculty of Science, UU. 2005-09

O. Tveretina. *Decision Procedures for Equality Logic with Uninterpreted Functions.* Faculty of Mathematics and Computer Science, TU/e. 2005-10

A.M.L. Liekens. *Evolution of Finite Populations in Dynamic Environments.* Faculty of Biomedical Engineering, TU/e. 2005-11

J. Eggermont. *Data Mining using Genetic Programming: Classification and Symbolic Regression.* Faculty of Mathematics and Natural Sciences, UL. 2005-12

B.J. Heeren. *Top Quality Type Error Messages.* Faculty of Science, UU. 2005-13

G.F. Frehse. *Compositional Verification of Hybrid Systems using*

Simulation Relations. Faculty of Science, Mathematics and Computer Science, RU. 2005-14

M.R. Mousavi. *Structuring Structural Operational Semantics*. Faculty of Mathematics and Computer Science, TU/e. 2005-15

A. Sokolova. *Coalgebraic Analysis of Probabilistic Systems*. Faculty of Mathematics and Computer Science, TU/e. 2005-16

T. Gelsema. *Effective Models for the Structure of pi-Calculus Processes with Replication*. Faculty of Mathematics and Natural Sciences, UL. 2005-17

P. Zoeteweij. *Composing Constraint Solvers*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2005-18

J.J. Vinju. *Analysis and Transformation of Source Code by Parsing and Rewriting*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2005-19

M.Valero Espada. *Modal Abstraction and Replication of Processes with Data*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2005-20

A. Dijkstra. *Stepping through Haskell*. Faculty of Science, UU. 2005-21

Y.W. Law. *Key management and link-layer security of wireless sensor networks: energy-efficient attack and defense*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-22

E. Dolstra. *The Purely Functional Software Deployment Model*. Faculty of Science, UU. 2006-01

R.J. Corin. *Analysis Models for Security Protocols*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-02

P.R.A. Verbaan. *The Computational Complexity of Evolving Systems*. Faculty of Science, UU. 2006-03

K.L. Man and R.R.H. Schiffelers. *Formal Specification and Analysis of Hybrid Systems*. Faculty of Mathematics and Computer Science and Faculty of Mechanical Engineering, TU/e. 2006-04

M. Kyas. *Verifying OCL Specifications of UML Models: Tool Support and Compositionality*. Faculty of Mathematics and Natural Sciences, UL. 2006-05

M. Hendriks. *Model Checking Timed Automata - Techniques and Applications.* Faculty of Science, Mathematics and Computer Science, RU. 2006-06

J. Ketema. *Böhm-Like Trees for Rewriting.* Faculty of Sciences, VUA. 2006-07

C.-B. Breunesse. *On JML: topics in tool-assisted verification of JML programs.* Faculty of Science, Mathematics and Computer Science, RU. 2006-08

B. Markvoort. *Towards Hybrid Molecular Simulations.* Faculty of Biomedical Engineering, TU/e. 2006-09

S.G.R. Nijssen. *Mining Structured Data.* Faculty of Mathematics and Natural Sciences, UL. 2006-10

G. Russello. *Separation and Adaptation of Concerns in a Shared Data Space.* Faculty of Mathematics and Computer Science, TU/e. 2006-11

L. Cheung. *Reconciling Nondeterministic and Probabilistic Choices.* Faculty of Science, Mathematics and Computer Science, RU. 2006-12

B. Badban. *Verification techniques for Extensions of Equality Logic.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2006-13

A.J. Mooij. *Constructive formal methods and protocol standardization.* Faculty of Mathematics and Computer Science, TU/e. 2006-14

T. Krilavicius. *Hybrid Techniques for Hybrid Systems.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-15

M.E. Warnier. *Language Based Security for Java and JML.* Faculty of Science, Mathematics and Computer Science, RU. 2006-16

V. Sundramoorthy. *At Home In Service Discovery.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-17

B. Gebremichael. *Expressivity of Timed Automata Models.* Faculty of Science, Mathematics and Computer Science, RU. 2006-18

L.C.M. van Gool. *Formalising Interface Specifications.* Faculty of Mathematics and Computer Science, TU/e. 2006-19

C.J.F. Cremers. *Scyther - Semantics and Verification of Security Protocols*. Faculty of Mathematics and Computer Science, TU/e. 2006-20

J.V. Guillen Scholten. *Mobile Channels for Exogenous Coordination of Distributed Systems: Semantics, Implementation and Composition*. Faculty of Mathematics and Natural Sciences, UL. 2006-21

H.A. de Jong. *Flexible Heterogeneous Software Systems*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2007-01

N.K. Kavaldjiev. *A run-time reconfigurable Network-on-Chip for streaming DSP applications*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-02

M. van Veelen. *Considerations on Modeling for Early Detection of Abnormalities in Locally Autonomous Distributed Systems*. Faculty of Mathematics and Computing Sciences, RUG. 2007-03

T.D. Vu. *Semantics and Applications of Process and Program Algebra*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2007-04

L. Brandán Briones. *Theories for Model-based Testing: Real-time and Coverage*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-05

I. Loeb. *Natural Deduction: Sharing by Presentation*. Faculty of Science, Mathematics and Computer Science, RU. 2007-06

M.W.A. Streppel. *Multifunctional Geometric Data Structures*. Faculty of Mathematics and Computer Science, TU/e. 2007-07

N. Trčka. *Silent Steps in Transition Systems and Markov Chains*. Faculty of Mathematics and Computer Science, TU/e. 2007-08

R. Brinkman. *Searching in encrypted data*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-09

A. van Weelden. *Putting types to good use*. Faculty of Science, Mathematics and Computer Science, RU. 2007-10

J.A.R. Noppen. *Imperfect Information in Software Development Processes*. Faculty of Electrical

Engineering, Mathematics & Computer Science, UT. 2007-11

R. Boumen. *Integration and Test plans for Complex Manufacturing Systems.* Faculty of Mechanical Engineering, TU/e. 2007-12

A.J. Wijs. *What to do Next?: Analysing and Optimising System Behaviour in Time.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2007-13

C.F.J. Lange. *Assessing and Improving the Quality of Modeling: A Series of Empirical Studies about the UML.* Faculty of Mathematics and Computer Science, TU/e. 2007-14

T. van der Storm. *Component-based Configuration, Integration and Delivery.* Faculty of Natural Sciences, Mathematics, and Computer Science,UvA. 2007-15

B.S. Graaf. *Model-Driven Evolution of Software Architectures.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2007-16

A.H.J. Mathijssen. *Logical Calculi for Reasoning with Binding.* Faculty of Mathematics and Computer Science, TU/e. 2007-17

D. Jarnikov. *QoS framework for Video Streaming in Home Networks.* Faculty of Mathematics and Computer Science, TU/e. 2007-18

M. A. Abam. *New Data Structures and Algorithms for Mobile Data.* Faculty of Mathematics and Computer Science, TU/e. 2007-19

W. Pieters. *La Volonté Machinale: Understanding the Electronic Voting Controversy.* Faculty of Science, Mathematics and Computer Science, RU. 2008-01

A.L. de Groot. *Practical Automaton Proofs in PVS.* Faculty of Science, Mathematics and Computer Science, RU. 2008-02

M. Bruntink. *Renovation of Idiomatic Crosscutting Concerns in Embedded Systems.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2008-03

A.M. Marin. *An Integrated System to Manage Crosscutting Concerns in Source Code.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2008-04

N.C.W.M. Braspenning. *Model-based Integration and Testing of High-tech Multi-disciplinary*

Systems. Faculty of Mechanical Engineering, TU/e. 2008-05

M. Bravenboer. *Exercises in Free Syntax: Syntax Definition, Parsing, and Assimilation of Language Conglomerates.* Faculty of Science, UU. 2008-06

M. Torabi Dashti. *Keeping Fairness Alive: Design and Formal Verification of Optimistic Fair Exchange Protocols.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2008-07

I.S.M. de Jong. *Integration and Test Strategies for Complex Manufacturing Machines.* Faculty of Mechanical Engineering, TU/e. 2008-08

I. Hasuo. *Tracing Anonymity with Coalgebras.* Faculty of Science, Mathematics and Computer Science, RU. 2008-09

L.G.W.A. Cleophas. *Tree Algorithms: Two Taxonomies and a Toolkit.* Faculty of Mathematics and Computer Science, TU/e. 2008-10

I.S. Zapreev. *Model Checking Markov Chains: Techniques and Tools.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-11

M. Farshi. *A Theoretical and Experimental Study of Geometric Networks.* Faculty of Mathematics and Computer Science, TU/e. 2008-12

G. Gulesir. *Evolvable Behavior Specifications Using Context-Sensitive Wildcards.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-13

F.D. Garcia. *Formal and Computational Cryptography: Protocols, Hashes and Commitments.* Faculty of Science, Mathematics and Computer Science, RU. 2008-14

P. E. A. Dürr. *Resource-based Verification for Robust Composition of Aspects.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-15

E.M. Bortnik. *Formal Methods in Support of SMC Design.* Faculty of Mechanical Engineering, TU/e. 2008-16

R.H. Mak. *Design and Performance Analysis of Data-Independent Stream Processing Systems.* Faculty of Mathematics and Computer Science, TU/e. 2008-17

M. van der Horst. *Scalable Block Processing Algorithms.* Faculty of

Mathematics and Computer Science, TU/e. 2008-18

C.M. Gray. *Algorithms for Fat Objects: Decompositions and Applications.* Faculty of Mathematics and Computer Science, TU/e. 2008-19

J.R. Calamé. *Testing Reactive Systems with Data - Enumerative Methods and Constraint Solving.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-20

E. Mumford. *Drawing Graphs for Cartographic Applications.* Faculty of Mathematics and Computer Science, TU/e. 2008-21

E.H. de Graaf. *Mining Semi-structured Data, Theoretical and Experimental Aspects of Pattern Evaluation.* Faculty of Mathematics and Natural Sciences, UL. 2008-22

R. Brijder. *Models of Natural Computation: Gene Assembly and Membrane Systems.* Faculty of Mathematics and Natural Sciences, UL. 2008-23

A. Koprowski. *Termination of Rewriting and Its Certification.* Faculty of Mathematics and Computer Science, TU/e. 2008-24

U. Khadim. *Process Algebras for Hybrid Systems: Comparison and Development.* Faculty of Mathematics and Computer Science, TU/e. 2008-25

J. Markovski. *Real and Stochastic Time in Process Algebras for Performance Evaluation.* Faculty of Mathematics and Computer Science, TU/e. 2008-26

H. Kastenberg. *Graph-Based Software Specification and Verification.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-27

I.R. Buhan. *Cryptographic Keys from Noisy Data Theory and Applications.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-28

R.S. Marin-Perianu. *Wireless Sensor Networks in Motion: Clustering Algorithms for Service Discovery and Provisioning.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-29

M.H.G. Verhoef. *Modeling and Validating Distributed Embedded Real-Time Control Systems.* Faculty of Science, Mathematics and Computer Science, RU. 2009-01

M. de Mol. *Reasoning about Functional Programs: Sparkle, a proof assistant for Clean.* Faculty of Science, Mathematics and Computer Science, RU. 2009-02

M. Lormans. *Managing Requirements Evolution.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2009-03

M.P.W.J. van Osch. *Automated Model-based Testing of Hybrid Systems.* Faculty of Mathematics and Computer Science, TU/e. 2009-04

H. Sozer. *Architecting Fault-Tolerant Software Systems.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-05

M.J. van Weerdenburg. *Efficient Rewriting Techniques.* Faculty of Mathematics and Computer Science, TU/e. 2009-06

H.H. Hansen. *Coalgebraic Modelling: Applications in Automata Theory and Modal Logic.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2009-07

A. Mesbah. *Analysis and Testing of Ajax-based Single-page Web Applications.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2009-08

A.L. Rodriguez Yakushev. *Towards Getting Generic Programming Ready for Prime Time.* Faculty of Science, UU. 2009-9

K.R. Olmos Joffré. *Strategies for Context Sensitive Program Transformation.* Faculty of Science, UU. 2009-10

J.A.G.M. van den Berg. *Reasoning about Java programs in PVS using JML.* Faculty of Science, Mathematics and Computer Science, RU. 2009-11

M.G. Khatib. *MEMS-Based Storage Devices. Integration in Energy-Constrained Mobile Systems.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-12

S.G.M. Cornelissen. *Evaluating Dynamic Analysis Techniques for Program Comprehension.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2009-13

D. Bolzoni. *Revisiting Anomaly-based Network Intrusion Detection Systems.* Faculty of Electrical En-

gineering, Mathematics & Computer Science, UT. 2009-14

H.L. Jonker. *Security Matters: Privacy in Voting and Fairness in Digital Exchange.* Faculty of Mathematics and Computer Science, TU/e. 2009-15

M.R. Czenko. *TuLiP - Reshaping Trust Management.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-16

T. Chen. *Clocks, Dice and Processes.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2009-17

C. Kaliszyk. *Correctness and Availability: Building Computer Algebra on top of Proof Assistants and making Proof Assistants available over the Web.* Faculty of Science, Mathematics and Computer Science, RU. 2009-18

R.S.S. O'Connor. *Incompleteness & Completeness: Formalizing Logic and Analysis in Type Theory.* Faculty of Science, Mathematics and Computer Science, RU. 2009-19

B. Ploeger. *Improved Verification Methods for Concurrent Systems.* Faculty of Mathematics and Computer Science, TU/e. 2009-20

T. Han. *Diagnosis, Synthesis and Analysis of Probabilistic Models.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-21

R. Li. *Mixed-Integer Evolution Strategies for Parameter Optimization and Their Applications to Medical Image Analysis.* Faculty of Mathematics and Natural Sciences, UL. 2009-22

J.H.P. Kwisthout. *The Computational Complexity of Probabilistic Networks.* Faculty of Science, UU. 2009-23

T.K. Cocx. *Algorithmic Tools for Data-Oriented Law Enforcement.* Faculty of Mathematics and Natural Sciences, UL. 2009-24

A.I. Baars. *Embedded Compilers.* Faculty of Science, UU. 2009-25

M.A.C. Dekker. *Flexible Access Control for Dynamic Collaborative Environments.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-26

J.F.J. Laros. *Metrics and Visualisation for Crime Analysis and Ge-*

nomics. Faculty of Mathematics and Natural Sciences, UL. 2009-27

C.J. Boogerd. *Focusing Automatic Code Inspections*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2010-01

M.R. Neuhäußer. *Model Checking Nondeterministic and Randomly Timed Systems*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2010-02

J. Endrullis. *Termination and Productivity*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2010-03

T. Staijen. *Graph-Based Specification and Verification for Aspect-Oriented Languages*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2010-04

Y. Wang. *Epistemic Modelling and Protocol Dynamics*. Faculty of Science, UvA. 2010-05

J.K. Berendsen. *Abstraction, Prices and Probability in Model Checking Timed Automata*. Faculty of Science, Mathematics and Computer Science, RU. 2010-06

A. Nugroho. *The Effects of UML Modeling on the Quality of Software*. Faculty of Mathematics and Natural Sciences, UL. 2010-07

A. Silva. *Kleene Coalgebra*. Faculty of Science, Mathematics and Computer Science, RU. 2010-08

J.S. de Bruin. *Service-Oriented Discovery of Knowledge - Foundations, Implementations and Applications*. Faculty of Mathematics and Natural Sciences, UL. 2010-09

D. Costa. *Formal Models for Component Connectors*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2010-10

M.M. Jaghoori. *Time at Your Service: Schedulability Analysis of Real-Time and Distributed Services*. Faculty of Mathematics and Natural Sciences, UL. 2010-11

R. Bakhshi. *Gossiping Models: Formal Analysis of Epidemic Protocols*. Faculty of Sciences, Department of Computer Science, VUA. 2011-01

B.J. Arnoldus. *An Illumination of the Template Enigma: Software Code Generation with Templates*. Faculty of Mathematics and Computer Science, TU/e. 2011-02

E. Zambon. *Towards Optimal IT Availability Planning: Methods and Tools*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2011-03

L. Astefanoaei. *An Executable Theory of Multi-Agent Systems Refinement*. Faculty of Mathematics and Natural Sciences, UL. 2011-04

J. Proença. *Synchronous coordination of distributed components*. Faculty of Mathematics and Natural Sciences, UL. 2011-05

A. Moralı. *IT Architecture-Based Confidentiality Risk Assessment in Networks of Organizations*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2011-06

M. van der Bijl. *On changing models in Model-Based Testing*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2011-07

C. Krause. *Reconfigurable Component Connectors*. Faculty of Mathematics and Natural Sciences, UL. 2011-08

M.E. Andrés. *Quantitative Analysis of Information Leakage in Probabilistic and Nondeterministic Systems*. Faculty of Science, Mathematics and Computer Science, RU. 2011-09