Quantifying information flow

David Mestel
Wolfson College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy
Trinity 2018

A little learning is a dangerous thing
—Alexander Pope

Acknowledgements

I would like to thank my supervisor, Bill Roscoe, for his encouragement and guidance. I also owe a particular debt to the verification and algorithms groups, from whose seminars I learned a great deal. I would also like to thank Cas Cremers and his students, especially Martin, Dennis and Katriel, for welcoming me so warmly. I also thank University College for two fantastic years as a college lecturer, where I benefited both from excellent students and a wide variety of scintillating dinner companions.

I am eternallly grateful to all those who have inspired me in mathematics down the years, but especially to my parents, who were the first.

Abstract

The main problem addressed by this thesis is that of characterising information leakage channels in interactive systems as either 'dangerous' or 'safe', on the basis of the amount of information that can be leaked. We define information information leakage in a fashion very similar to the classical definition of Goguen and Meseguer, modeling systems as finite state transducers. Interesting issues arise in the selection of the appropriate measure of information, and we propose some novel definitions which address the problems which have been identified with the traditional notion of mutual information.

We show that for deterministic systems, the problem of quantifying information flow can be reduced to a natural combinatorial problem on regular languages, namely that of computing their 'width' with respect to the lexicographic partial order. We solve this problem, and show that there is a dichotomy between exponential and polynomial width, corresponding to linear and logarithmic information flow (which we characterise as 'dangerous' and 'safe' respectively). We give a polynomial time algorithm to distinguish the two cases, as well as to compute the precise rate of logarithmic information flow. We also study the analagous question for some other partially ordered structures: we show that for context-free languages there is a similar dichotomy but the problem of distinguishing the two cases is undecidable, and that for regular tree languages there is a trichotomy between doubly exponential, singly exponential and polynomial growth.

Finally we consider information flow in the setting of the process algebra CSP. In order to reduce to the framework of languages and automata, we establish a general theorem showing that a wide class of finite observational models of CSP (including all known models) can be reduced to the traces model using the 'priority' operator.

Contents

List of Figures

Chapter 1

Introduction

The discovery of the Meltdown [41] and Spectre [37] vulnerabilities brought the issue of side-channel attacks, and their mitigation, to great public prominence. However, the need to formally verify the absence of information flow has been recognised since at least the 1970s [21]. The absence of information flow is known as 'non-interference', and a great deal of theory has been developed surrounding it. More recently, however, it has been recognised that to require the total absence of information flow may be too conservative.

In the abstract for a discussion session at CSF 2001 entitled 'Non-interference, who needs it?' [59], Ryan, McLean, Millen and Gligor observe that 'In most non-inteference models, a single bit of compromised information is flagged as a security violation, even if one bit is all that is lost. To be taken seriously, a non-interference violation should imply a more significant loss... Characterization of unbounded channels is suggested as the kind of goal that would advance the study of this subject'. The objective of this thesis is to achieve this goal for deterministic systems: we show that, for the standard notion of noninterference introduced by Goguen and Meseguer in 1982, there is a natural characterisation of channels as 'safe' or 'dangerous' (corresponding to linear versus logarithmic information flow), and a polynomial algorithm to detect the two cases.

We begin in Chapter 2 by setting out some background and existing work in non-interference analysis and quantified information flow. We observe that a number of attempts have been made to define quantified information flow, but none are without problems and virtually no algorithms for explicit calculation are known.

In Chapter 3 we illustrate the utility of quantified information flow by considering a toy scenario in which a number of users share a single resource, stating at the beginning of the execution the number of cycles they wish to use. We show that

permitting a small amount of information flow (in number of bits proportional to $\log \log n$) allows a greatly reduced overhead.

In Chapter 4 we set out the information-theoretic foundations for our study of information flow. We define the information flow for an interactive system, and show (Theorem 4.12) that for the case of interactive systems there is a characterisation in terms of possible sets of observations for the observer, which we will find simplifies calculation considerably. Finally we discuss some of the objections that have been raised to the consensus definition of information flow, and suggest possible alternatives.

In Chapter 5 we explain how to model interactive systems as finite-state transducers, and how to apply the theory of Chapter 4. We then show how to reduce the problem of calculating information flow for a given transducer to the problem of calculating the *width* of the language of a finite automaton derived from it (the width of a language L is the size of the largest antichain in L, with respect to the lexicographic order).

In Chapter 6 we show how to solve this problem, showing that there is a dichotomy between polynomial and exponential antichain growth, corresponding to logarithmic and linear information flow respectively (Theorem 6.16), and an algorithm to determine which holds for a given automaton (Theorem 6.18). We argue that it is reasonable to interpret these two cases as 'safe' and 'dangerous' respectively. In the case of polynomial antichain growth, we show that there is an algorithm to determine the precise order of polynomial growth. Finally we consider the mathematical problem of antichain growth in other automatic structures, and show that there is a similar dichotomy for context-free languages but that the decision problem is undecidable. We also show that for regular tree languages there is a trichotomy between polynomial, exponential and doubly exponential antichain growth.

In Chapter 7 we consider quantified non-interference in the paradigm of Communicating Sequential Processes (CSP). Because the rich structure of denotational models for CSP are essential to quantified non-interference, whereas all of our work in this thesis has been based on languages of automata and transducers, we must reduce these models to the traces model. We give a general result (Theorem 7.30) showing that any one of a wide class of finite observational models (including all that are known) can be reduced to the traces model by the use of an appropriate context, a technique which we refer to as 'model shifting'. We then discuss the application of this technique to quantify information flow with respect to the finite linear observations model $\mathcal{FL}$.

Finally in Chapter 8 we conclude and discuss directions for future work.

The model shifting results of Chapter 7 have appeared as [46], joint with A.W. Roscoe. The antichain results of Chapter 6 have appeared on the arXiv as [45].

Chapter 2

Background

2.1 Noninterference analysis

The foundations of noninterference analysis were laid by Goguen and Meseguer in [31]. They take an essentially automata-theoretic approach, modeling the system as an automaton to which users may send commands (causing the state to change), and which displays outputs to users depending on the state.

Definition 2.1. A *GM state machine* is a tuple $(S, U, C, \mathrm{Out}, \mathrm{do}, s_0)$, where

(i) S is a set of *states*,

(ii) U is a set of *users*,

(iii) C is a set of *commands*,

(iv) Out is a set of *outputs*,

(v) out is the *output function*, a function $S \times U \to \mathrm{Out}$, denoting the output seen by each user in each state,

(vi) do is the *state transition function*, a function $S \times U \times C \to S$, determining how the state changes when command c is received from user u, and

(vii) $s_0 \in U$ is the *initial machine state*.

They also include a more complicated notion of 'system transition functions', which we do not discuss here since as they observe it can be simulated by the basic state machine definitions.

Given a sequence of commands $w = (u_1, c_1) \ldots (u_k, c_k) \in (U \times C)^*$, and a single user $u \in U$, we write $[\![w]\!]_u$ for the output seen by u after the sequence w; that is,

$[\![w]\!]_u = \text{out}(s_k, u)$, where $s_0 s_1 \dots s_k \in S^*$ is the sequence of states satisfying $s_i = \text{do}(s_{i-1}, u_i, c_i)$.

Definition 2.2. Given sets $G, G' \subseteq U$, we say that G *does not interfere with* G', and write $G :| G'$, if for every $u \in G'$ and every $w \in (U \times C)^*$ we have

$$[\![w]\!]_u = [\![P_G(w)]\!]_u,$$

where $P_G(w)$, referred to as the *purge* of w, denotes w with all commands sent by users in G removed (that is, a pair (u_i, c_i) is removed if $u_i \in G$).

Trivially (by transitivity of $=$) this definition is equivalent to saying that $[\![w]\!]_u = [\![w']\!]_u$ whenever w and w' disagree only on G-actions; that is, whenever

$$w|_{((U \setminus G) \times C)^*} = w'|_{((U \setminus G) \times C)^*}.$$

The complexity of deciding this property is not analysed in [31], but van der Meyden and Zhang show in [65] that it is in PTIME.

For the purposes of this work, it will be convenient to have a recharacterisation (which so far as we are aware is original) of the Goguen-Meseguer noninterference property in terms of *stutter traces* ([40]).

Definition 2.3. For a sequence of commands $w = (u_1, c_1) \dots (u_k, c_k)$ and a user $u \in U$, let $\phi_{u,G}(w) \in (\text{Out} \sqcup C)^*$, the *trace observed by u over w with respect to G*, be the sequence of outputs for u, interleaved with the commands sent by users not in G. Formally, let

$$\phi_{u,G}(w) = \text{out}(s_0, u) f_G(u_1, c_1) \text{out}(s_1, u) f_G(u_2, c_2) \text{out}(s_2, u) \dots f_G(u_k, c_k) \text{out}(s_k, u),$$

where $s_0 s_1 \dots s_k \in S^*$ is the sequence of states satisfying $s_i = \text{do}(s_{i-1}, u_i, c_i)$, and the function $f_G : U \times C \to C^*$ is defined by $f_G(u', c) = \epsilon$ if $u' \in G$ and $f_G(u', c) = c$ otherwise.

Let $\psi_{u,G}(w) \in (\text{Out} \sqcup C)^*$, the *stutter trace observed by u over w with respect to G*, be equal to $\phi_{u,G}(w)$ with duplicate outputs removed. Formally, let $\psi_{u,G}(w)$ be the minimum-length element of the class of $\phi_{u,G}(w)$ under the equivalence relation $\sim$ generated by $xx \sim x$ for all $x \in \text{Out}$.

Theorem 2.4. *Let $M = (S, U, C, Out, do, s_0)$ be a GM state machine, and let $G, G' \subseteq U$. Then $G :| G'$ if and only if for every $u \in G'$ and every $w, w' \in (U \times C)^*$ we have that $\psi_{u,G}(w)$ and $\psi_{u,G}(w')$ do not first differ by an element of Out (that is, we do not have $\psi_{u,G}(w) = vxv_1$ and $\psi_{u,G}(w') = vx'v_2$ with $v, v_1, v_2 \in (Out \sqcup C)^*$ and $x \neq x' \in Out^*$).*

Proof. Without loss of generality we may assume that w and w' are minimal such that $\psi_{u,G}(w)$ and $\psi_{u,G}(w')$ first differ by an element of Out^*; hence we have that $\psi_{u,G}(w) = vx$ and $\psi_{u,G}(w) = vx'$ for some $v \in (\mathrm{Out} \sqcup C)^*$ and (without loss of generality) $x \in \mathrm{Out}$, $x' \in \mathrm{Out} \cup \{\epsilon\}$ with $x \neq x'$. Hence we have that $w|_{((U\setminus G)\times C)^*} = w|_{((U\setminus G)\times C)^*}$ (since all commands sent by users outside G are recorded in v), but

$$[\![w]\!]_u = x \neq [\![w']\!]_u,$$

since if we have $x' \in \mathrm{Out}$ then $[\![w']\!]_u = x' \neq x$, and on the other hand if $x' = \epsilon$ then by the definition of $\psi_{u,G}(w)$ as a stutter trace we have that the final letter of v is not x. Hence $G \not| G'$.

Conversely, suppose that $G \not| G'$, and in particular that $G \not| \{u\}$. Let w be minimal such that $[\![w]\!]_u \neq [\![P_G(w)]\!]_u$. Let $P_G(w) = (u_1, c_1)\dots(u_k, c_k)$, and write

$$w = (u_0^1, c_0^1)\dots(u_0^{n_0}, c_0^{n_0})(u_1, c_1)(u_1^1, c_1^1)\dots(u_1^{n_1}, c_1^{n_1})(u_2, c_2)\dots(u_k, c_k)(u_k^1, c_k^1)\dots(u_k^{n_k}, c_k^{n_k}),$$

where $u_i \notin G$ for all i and $u_i^j \in G$ for all i, j.

By minimality of w, we have $[\![(u_0^1, c_0^1)\dots(u_i^j, c_i^j)]\!]_u = [\![(u_1, c_1)\dots(u_i, c_i)]\!]_u$ for all i, j apart from $i = k, j = n_k$. Hence $\psi_{u,G}(w) = \psi_{u,G}(P_G(w))[\![w]\!]_u$, and so $\psi_{u,G}(w)$ and $\psi_{u,G}(P_G(w))$ first differ by an element of Out^*. □

In the case $G' = \{u\}, G = U \setminus G'$, Theorem 2.4 provides a perspective on noninterference which we will see reflected both in other approaches to the problem, and in the quantitative approach of this thesis. We can view noninterference as saying that the view of the system presented to u is in some sense deterministic: in any two distinct interactions she may have with the system, they must be distinguished first by an action of u herself, rather than by a nondeterministic choice made by the system. Moreover, in later parts of this work we will show that an information-theoretic measure of information flow is obtained by determining the maximum size of sets of executions obeying a constraint equivalent to that in the theorem that no two executions may first differ by an element of Out.

In our work, we will largely follow these definitions, with a few minor changes. Firstly, we will view outputs as determined by transitions, rather than states: that is, rather than a function out associating an output to each state, we will have a function $(S \times U \times C) \times U \to \mathrm{Out}$, associating to each transition an output for each user. The technical terminology is that we will consider *action-observed* rather than *state-observed* state machines. Note that this notion is strictly more general than the state-observed definition above: in particular, we can always take the function $((s, u, c), u') \mapsto \mathrm{out}(\mathrm{do}(s, u, c), u')$.

Secondly, we will mainly consider *synchronous* rather than *asynchronous* machines: that is, rather than accepting a sequence of commands from different users, the machine will demand a command from each user at each step. The main reason for this is that in the absence of timing, the asynchronous model poses problems for quantifying information flow. However, we will consider the asynchronous case in Section 5.6, where we will see that (after making reasonable choices to resolve the semantic issues that arise) the theory works out as more-or-less equivalent.

The considerations of state-observed versus action-observed systems are considered in more detail by van der Meyden and Zhang in [66], in which they show (Theorem 4.3) that a number of notions of noninterference are (in a suitable sense) equivalent for the two types of semantic model.

2.2 Quantified noninterference

As discussed in the introduction, it has been recognised that the Goguen-Meseguer notion of noninterference may be too strong. A number of attempts have been made, therefore, to define and calculate quantitive measures of information flow.

In [20], Denning proposes a definition of quantified information flow based on differences in entropy: in the state transition $s \to s'$ there is information flow from from variable Y to variable X if the entropy of Y conditional on seeing the value of X in state s' is less than that conditional on seeing the value of X in state s; that is, if $H(Y|X = x') < H(Y|X = x)$ (where x and x' are the values of X in states s and s' respectively).

In [16], Clark, Hunt and Malacaria point out that Denning's definition is flawed, essentially because the later observation may contain information that supplements that of the earlier observation, even if the total information in the second observation is less. They argue that the appropriate definition is the standard information-theoretic notion of *mutual information*, as proposed by Gray, who showed ([32], Theorem 5.1) that the usual definition of noninterference corresponds to having a channel capacity of zero. As will be developed below, we adopt this standard definition, and provide further evidence with a Shannon-style theorem (Theorem 4.11) showing that information flow of C under our definition corresponds with the existence of a channel from High to Low with capacity C. Other authors favouring mutual information as the appropriate measure of information flow include Moskowitz, Newman, Crepeau and Miller in [49] and Chatzikokolakis, Palamidessi and Panangaden in [14].

On the other hand, in [63] Smith argues against the 'emerging consensus' that mutual information is the appropriate measure. The reason for this is essentially that mutual information is the *expected* amount of information in bits. This means, for example, that if there is a 99% chance of transmitting no information and a 1% chance of transmitting 100 bits then the mutual information is 1 bit. However, there is a clearly a significant difference between this situation and that of always transmitting 1 bit. In the former case, a 100 bit key (say) will be revealed 1% of the time, whereas in the latter it will always be unfeasible to exhaust the remaining 99 bits of entropy.

Smith proposes to resolve this by measuring the change in the *vulnerability*, which is the modal probability of High's action, on the grounds that this bounds the probability with which Low can guess what High's action was. With this definition, the relevant quantity is the change in the *min-entropy*

$$H_\infty(X) = \log \frac{1}{\max_x P(X = x)}$$

(or rather the difference $H_\infty(X) - H_\infty(X|Y)$).

However, this definition is itself flawed. For example, consider the system which accepts from High an input from the set $[0, 2^k - 1] \cup \{X\}$, and outputs to Low an element of the set $[0, 2^k - 1]$, where the system just relays inputs from $[0, 2^k - 1]$, and on receiving 'X' outputs a uniformly random element of $[0, 2^k - 1]$. Suppose that High's strategy is to input 'X' with probability $1/2$, and otherwise a uniformly random element of $[0, 2^k - 1]$. Then the modal action of High is 'X', both *a priori* and also after seeing any particular output to Low (in the latter case, along with the relevant integer). Smith's definition thus finds no information flow, which is clearly incorrect.

An alternative approach, discussed briefly by Köpf and Basin in [38], is to adopt a 'worst case' measure, but this is also clearly wrong in general: an exponential amount of information flow is acceptable if it is exponentially improbable. Since none of the existing measures are entirely satisfactory, we discuss the correct notion in Section 4.5.

Building on Smith's work, a more general notion referred to as 'g-leakage' has been developed in a series of papers starting with [3]. We give more detailed consideration to this theory and its relationship to our proposed definitions in Section 4.7.

In [15], Clark and Hunt consider the Goguen-Mesegeur definition of noninterference, and show (Theorem 1) that if a noninterference property fails then in fact it fails for the setting where the parties are only allowed to use deterministic strategies. We prove a quantitative version of this fact (i.e. that the amount of information flow, rather than just whether it is non-zero, is preserved by allowing only deterministic

strategies) in Theorem 4.12. They further show (Theorem 2) that noninterference is preserved by considering only *stream strategies*: that is, strategies where the sequence of events emitted by the experimenter is independent of what he sees. This is not surprising: a single counterexample suffices to violate noninterference, so we can take stream strategies matching that example. Note that this does not hold for quantified information flow, since it is possible that the high-level user can send the system down two alternative paths, such that if the adversary wishes to profit from both then she must behave differently in the two cases.

2.3 Computing information flow for interactive systems

In [5], Andrés, Palamidessi, van Rossum and Smith attempt to compute the leakage of what they term 'information-hiding systems' (IHS). These are essentially automata over (secret) inputs and (observed) outputs, of a fairly similar flavour to our definitions in Chapter 5. However, this work has several important limitations. Firstly, it assumes an essentially passive attacker: apart from the values of the secret (whose distribution they sometimes allow to be chosen so as to maximise information flow), the system is assumed to follow known probabilistic behaviour.

Secondly, having oberved that where the system is 'interactive' (that is, choices of secrets are interleaved with outputs, rather than preceding them) it cannot simply be represented as the channel matrix of a memoryless channel, they are only able to analyse information flow where the secret choices are made with known probabilities (rather than finding the maximum information flow over probability distributions for the secret choices). We resolve this apparent problem in Chapters 4 and 5 by observing that we may choose to think of the secret choice as being a choice of *strategy*: that is, a choice of secret value for each sequence of events that may have occurred up to the point where the choice is made. Having done this, we are again able to study the system as a memoryless channel in the usual way (we suspect this may be the essence of what is described in the follow-up paper [4]).

In [38], Köpf and Basin show how to calculate information leakage for a highly restrictive model (which they relate to side-channel attacks). They assume that the secret is drawn uniformly from some finite set K, and that the attacker has access to some finite set M of (non-injective) functions on K. At each round, the attacker chooses a function from M and observes its output. The goal is to compute the

expected reduction in the uncertainty in the secret (for a variety of uncertainty measures) after n steps, assuming that the attacker chooses her queries so as to maximise this.

The authors show (Theorem 1) that the amount of information obtained after n rounds can be computed in time $O\left(n|M|^{r^n}|K|\log|K|\right)$ by exhaustive search (where $r = \max_{f\in M}|f(K)|$ is the maximum size of the images of functions in M). They also show (Theorem 2) that the amount of information obtained by a 'greedy' attacker (who maximises the amount of information she obtains at each step) can be computed in time $O\left(nr|M||K|^2\right)$. Unfortunately no nontrivial relationship is known between these two quantities. However, since the information obtained by the greedy attacker is clearly a lower bound for that obtained by the optimal attacker, the greedy heuristic may be able to show the existence of a side-channel attack for some systems.

In [8], Boreale and Pampaloni consider the more general case of repeated queries issued by the attacker (possibly adaptively) to a system which is *stateless*: that is, responses to particular queries are not affected by the queries which have been asked previously, although the queries chosen by the attacker may vary depending on what information she has previously received.

The authors show (in Theorem 4.3) that the problem of computing the maximum information flow after n queries is NP-hard, for deterministic systems specified as boolean formulae (that is, the deterministic function specifying the response to a given query on a given value of the secret, all encoded in binary, is specified as a boolean function).

In [9], Boreale, Pampaloni and Paolini consider the asymptotics of the information flow resulting from n independent uses of a single channel for large n. This is in some sense dual to the situation we will be studying, of the asymptotics of a single, long execution of a stateful system.

In [10], Boreale and Paolini consider the problem of obtaining statistically valid estimates of information flow capacity of a system, given only 'black box' access, and show that under certain reasonable assumptions this is impossible.

Chapter 3

Scheduling a shared resource

3.1 Introduction

In this chapter, we illustrate the potential utility of quantified information flow by considering a heavily simplified model of scheduling a shared resource, such as a processor or a communication channel with limited capacity. In such a setting there is an inevitable trade-off between efficiency and privacy: as we shall see, any scheduler which prevents any information flow must have a fairly large overhead, and on the other hand a scheduler optimised for efficiency will leak information (in bits) proportional to $\log n$, the logarithm of the amount of work being scheduled.

We shall show, however, that if we are prepared to take an intermediate position between these two extremes then we can obtain an arbitrarily low overhead while revealing only information proportional to $\log \log n$, an exponential decrease from the scheduler optimised for performance.

We should note, however, that the model treated in this chapter is much simplified, and in particular assumes a one-off interaction wherein each agent simultaneously asks to schedule a single job. This is unlikely to be realistic in most situations: usually what is required is a protocol whereunder agents can repeatedly ask to schedule jobs, doing so at times of their choosing. The more complex semantics required to model this setting, and the mathematical techniques required to analyse them, will be the subject of the remainder of this thesis.

We first consider the case of just two agents sharing the resource, in Section 3.2. After setting out the relevant basic definitions and giving some example schedulers, we set out the relevant figures of merit, encapsulating the twin goals of efficiency and security, which in this case as so often are in tension. 'Efficiency' is measured by the excess of total waiting time over the minimum possible. 'Security' is measured by the amount of information which can flow between the two agents. This in turn is

measured by the total number of different possible waiting times that each agent can observe for a single choice of their own request size n. We are not yet in a position to justify this definition, but after Chapter 4 it will become clear that it is correct.

We show in Theorem 3.12 that there is in some sense an optimal tradeoff between these two desiderata: for any δ there exists a scheduler with efficiency overhead at most δ such that the information flow in bits is at most $\log\log n - \log\log(1+\delta)$. On the other hand any scheduler which allows information flow less than $\log\log n - \log\log(1+\delta)$ must have efficiency overhead greater than δ.

We then proceed in Section 3.3 to consider the case of more than two agents. This is particularly significant because if there are many agents with wildly differing demands then the cautious scheduler (which allows no information flow) my have very high overhead. However, we again show in Theorem 3.25 that there is an optimal tradeoff between efficiency and security, in a scheduler which again has efficiency overhead δ and allows information flow at most $\log\log n - \log\log(1+\delta)$.

3.2 The two-agent case

We consider first the case of only two agents. The task of the system is to receive a request from the two agents specifying the sizes n_1, n_2 of their respective jobs. It must decide how to allocate the shared resource at each time unit, and then notify each agent when their job is complete, although of course it could choose to delay a notification until some later time.

Assuming (as we shall) that the scheduler is deterministic, it induces functions T_1, T_2, the times after which the two agents are notified of the completion of their respective jobs, each taking as arguments the two job sizes n_1, n_2. We will choose to identify the scheduler with these functions, imposing the condition that they be genuinely implementable.

Throughout this chapter we will denote by $\mathbb{N}$ the set of natural numbers including zero, and by $\mathbb{N}^+$ the set of strictly positive natural numbers.

Definition 3.1. A *2-scheduler* is a pair (T_1, T_2) of functions $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$.

A scheduler is implementable if there is a scheduling function f, with $f(k)$ determining which agent's job is worked on at time k, such that each job receives at least n_i units of work and is completed by time T_i.

Definition 3.2. A 2-scheduler (T_1, T_2) is *valid* if for every $n_1, n_2 \in \mathbb{N}$ there exists a function

$$f : \mathbb{N}^+ \to \{0, 1, 2\}$$

such that for each $i \in \{1, 2\}$ we have

(i) $|f^{-1}(i)| = n_i$, and

(ii) $\max\{f^{-1}(i)\} \leq T_i(n_1, n_2)$.

There are simple healthiness conditions for a 2-scheduler to be valid.

Proposition 3.3. *A 2-scheduler (T_1, T_2) is valid if and only if the following hold for all n_1, n_2:*

(i) $T_1(n_1, n_2) \geq n_1$,

(ii) $T_2(n_1, n_2) \geq n_2$, *and*

(iii) $\max(T_1(n_1, n_2), T_2(n_1, n_2)) \geq n_1 + n_2$.

Proof. For the first and second conditions, we have $n_i = |f^{-1}(i)| \leq \max\{f^{-1}(i)\} \leq T_i(n_1, n_2)$. For the third condition we have that

$$\begin{aligned} n_1 + n_2 &= \left|f^{-1}(1)\right| + \left|f^{-1}(2)\right| \\ &= \left|f^{-1}(1) \cup f^{-1}(2)\right| \\ &\leq \max\left\{f^{-1}(1) \cup f^{-1}(2)\right\} \\ &= \max\left(\max\left\{f^{-1}(1)\right\}, \max\left\{f^{-1}(2)\right\}\right) \\ &\leq \max\left(T_1(n_1, n_2), T_2(n_1, n_2)\right). \end{aligned}$$

Conversely, suppose that the above conditions hold. Without loss of generality $T_1(n_1, n_2) \leq T_2(n_1, n_2)$. Now let $f([1, n_1]) = 1$, $f([n_1 + 1, n_1 + n_2]) = 2$ and $f([n_1 + n_2 + 1, \infty)) = 0$. Then it is clear that f satisfies the first condition of Definition 3.2. For the second, we have $\max\{f^{-1}(1)\} = n_1 \leq T_1(n_1, n_2)$, and $\max\{f^{-1}(2)\} = n_1 + n_2 \leq \max(T_1(n_1, n_2), T_2(n_1, n_2)) = T_2(n_1, n_2)$. □

Perhaps the simplest possible scheduler is the one which always gives priority to the first agent's job, and completes it before the second starts. It is easy to compute the functions T_1, T_2 in this case.

Example 3.4. The *priority* scheduler is given by $(T_1, T_2)(n_1, n_2) = (n_1, n_1 + n_2)$.

For this example the total time $T_1 + T_2$ spent by both agents waiting for their jobs to complete is $n_2 + 2n_1$. If $n_1 \gg n_2$ this may be very far from optimal. A more efficient scheduler will choose to put the shorter job first.

Example 3.5. The *impatient* scheduler is given by

$$(T_1, T_2)(n_1, n_2) = \begin{cases} (n_1, n_1 + n_2) & \text{if } n_1 \leq n_2 \\ (n_1 + n_2, n_2) & \text{otherwise.} \end{cases}$$

This clearly has total waiting time $n_1+n_2+\min(n_1, n_2)$, and in fact this is minimal among all valid 2-schedulers.

Proposition 3.6. *Let (T_1, T_2) be a valid 2-scheduler. Then $(T_1 + T_2)(n_1, n_2) \geq n_1 + n_2 + \min(n_1, n_2)$.*

Proof. We have

$$\begin{aligned}(T_1 + T_2)(n_1, n_2) &= \max\left(T_1(n_1, n_2), T_2(n_1, n_2)\right) + \min\left(T_1(n_1, n_2), T_2(n_1, n_2)\right) \\ &\geq n_1 + n_2 + \min(n_1, n_2)\end{aligned}$$

by Proposition 3.3. □

We will measure efficiency by the excess of waiting time over this minimum.

Definition 3.7. Let $T = (T_1, T_2)$ be a valid 2-scheduler. The *overhead* $W : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ of T is given by

$$W(n_1, n_2) = (T_1 + T_2)(n_1, n_2) - \left(n_1 + n_2 + \min(n_1, n_2)\right).$$

Note that the impatient scheduler has $W(n_1, n_2) = 0$ and the priority scheduler has $W(n_1, n_2) = n_1 - \min(n_1, n_2)$.

So it is clear that if our only concern is efficiency the impatient scheduler is optimal. But what about privacy? We consider the sizes of the sets of possible waiting times experienced by each individual agent, for each possible size of their job.

Definition 3.8. Let (T_1, T_2) be a 2-scheduler. Then the *observation count functions* I_1 and I_2 are given by $I_1(n) = |T_1(n, \mathbb{N})|, I_2(n) = |T_2(\mathbb{N}, n)|$.

The amount of information (in bits) obtained by agent k, subject to the constraint that the job she requests must have size at most N, is then

$$\max_{n \leq N} \log I_k(n).$$

The disadvantage of the impatient scheduler is now apparent: it has $I_1(n) = n$ and $I_2(n) = n+1$ (the corresponding sets being $\{n, n+1, \ldots, 2n-1\}$ and $\{n, n+1, \ldots, 2n\}$ respectively). The priority scheduler performs similarly poorly, with $I_1(n) = 1$ but $I_2(n) = \infty$ (although it would be appropriate for the case where the second agent is trusted but the first agent is not).

If we want to ensure that neither agent is able to learn anything about the other, we can adopt a simple *time division multiplexing* system, where the system allocates, say, the odd-numbered time steps to the first agent's job and the even-numbered steps to the second agent's job.

Example 3.9. The *cautious* scheduler is given by $T_1(n_1, n_2) = 2n_1-1$ and $T_2(n_1, n_2) = 2n_2$.

This has $I_1(n) = I_2(n) = 1$, and $W(n_1, n_2) = n_1 + n_2 - 1 - \min(n_1, n_2) = \max(n_1, n_2) - 1$. In fact this factor-two overhead is unavoidable among schedulers allowing no information flow.

Proposition 3.10. *Let (T_1, T_2) be a valid 2-scheduler with $I_1(n) = I_2(n) = 1$ for all n. Then*

$$\lim_{N\to\infty} \sup_{n_1,n_2>N} \frac{(T_1 + T_2)(n_1, n_2)}{n_1 + n_2} \geq 2.$$

Proof. Since we have $I_1(n) = I_2(n) = 1$, we may treat T_1, T_2 as functions of a single variable (respectively n_1 and n_2).

Suppose for contradiction there exists some $C < 2$ and some N such that for all $n_1, n_2 \geq N$ we have $T_1(n_1) + T_2(n_2) < C(n_1 + n_2)$.

Now let

$$n = \left\lceil \frac{C}{2 - C} N \right\rceil + 1.$$

Then without loss of generality we have $T_1(n) \geq T_2(n)$, and hence $T_1(n) \geq 2n$. But then

$$\begin{aligned} T_1(n) + T_2(N) &> 2n \\ &= C(n + N) + n(2 - C) - CN \\ &> C(n + N) + CN - CN \\ &= C(n + N), \end{aligned}$$

which is a contradiction. □

However, if we are prepared to accept a small amount of information flow (proportional in bits to $\log\log n$), we can reduce the overhead to an arbitrary constant factor δ. The basic idea is that we 'round' the output given to each agent to the next power of $(1+\delta)$. On the one hand since $(1+\delta)^k$ grows exponentially we have that the number of possible outputs to each agent is only approximately $\log n$, but on the other hand this cannot increase the waiting time by more than a factor of $(1+\delta)$.

Example 3.11. For any fixed $\delta > 0$, the *δ-optimal scheduler* is given by

$$T_1(n_1, n_2) = \begin{cases} n_1 & \text{if } n_1 \leq n_2 \\ n_1 + \min\left\{ \left\lceil (1+\delta)^k \right\rceil \middle| k \in \mathbb{N}, \left\lceil (1+\delta)^k \right\rceil \geq n_2 \right\} & \text{otherwise,} \end{cases}$$

and $T_2(n_1, n_2) = T_1(n_2, n_1)$ whenever $n_1 \neq n_2$, and

$$T_2(n, n) = n + \min\left\{ \left\lceil (1+\delta)^k \right\rceil \middle| k \in \mathbb{N}, \left\lceil (1+\delta)^k \right\rceil \geq n \right\}.$$

We shall justify this ambitious title by Theorem 3.12.

For this scheduler, we have

$$\begin{aligned} (T_1 + T_2)(n_1, n_2) &= n_1 + n_2 + \min\left\{ \left\lceil (1+\delta)^k \right\rceil \middle| k \in \mathbb{N}, \left\lceil (1+\delta)^k \right\rceil \geq \min(n_1, n_2) \right\} \\ &\leq n_1 + n_2 + (1+\delta)\min(n_1, n_2), \end{aligned}$$

and hence $W(n_1, n_2) \leq \delta \min(n_1, n_2)$.

On the other hand, we have

$$\begin{aligned} I_1(n) &= \left| \{n\} \cup \left\{ n + \left\lceil (1+\delta)^k \right\rceil \middle| k \in \mathbb{N}, \left\lceil (1+\delta)^{k-1} \right\rceil < n \right\} \right| \\ &\leq 2 + \log_{1+\delta} n \\ &= 2 + \frac{\log n}{\log(1+\delta)}, \end{aligned}$$

and similarly for $I_2(n)$, save that the inequality in the set comprehension in the first line above ceases to be strict, which does not affect the final bound.

Thus we have that $I_1(n), I_2(n) \sim \log n / \log(1+\delta)$ as $n \to \infty$. In fact this is optimal among schedulers with overhead bounded by $\delta \min(n_1, n_2)$.

Theorem 3.12. *Let (T_1, T_2) be a valid scheduler, and let $\delta > 0$. Suppose that either*

$$\limsup_{n\to\infty} \frac{I_1(n)}{\log n} < \frac{1}{\log(1+\delta)}$$

or

$$\limsup_{n\to\infty} \frac{I_2(n)}{\log n} < \frac{1}{\log(1+\delta)}.$$

Then

$$\lim_{N\to\infty} \sup_{n_1, n_2 > N} \frac{W(n_1, n_2)}{\min(n_1, n_2)} > \delta.$$

Proof. The general strategy of the proof is to show that because of the constraint on observation count functions there must exist two values $n_2 < n_2'$ of substantially different magnitudes such that $T_1(n, n_2) = T_1(n, n_2')$. We then show that the healthiness conditions on $T_1(n, n_2')$ for validity are not consistent with efficiency for $T_1(n, n_2)$.

Let (T_1, T_2) be some valid scheduler, and suppose without loss of generality that $\limsup_{n\to\infty} I_1(n)/\log n < 1/\log(1+\delta)$. Then there exists some $\widetilde{\delta} > \delta$ such that for all sufficiently large n

$$|T_1(n, \mathbb{N})| < \frac{\log n}{\log(1+\widetilde{\delta})}.$$

Indeed, otherwise we have that for all $\widetilde{\delta} > \delta$, $I_1(n)/\log n \geq 1/\log(1+\widetilde{\delta})$ infinitely often, and hence $\limsup_{n\to\infty} I_1(n)/\log n \geq 1/\log(1+\delta)$.

Now fix some δ' such that $\delta < \delta' < \widetilde{\delta}$ and fix $\epsilon > 0$ such that $(1+\epsilon)\log(1+\delta') < \log(1+\widetilde{\delta})$. Fix $\kappa > 2 + \delta'$. Then we have that for all sufficiently large n

$$\begin{aligned}\frac{\log n}{\log(1+\widetilde{\delta})} &< \frac{\log n - \log\kappa - \log\log n}{(1+\epsilon)\log(1+\delta')} \\ &= \frac{1}{1+\epsilon}\log_{1+\delta'}(n/\kappa) - \frac{1}{1+\epsilon}\log_{1+\delta'}(\log n)\end{aligned}$$

Now let the set X contain a single integer from the range

$$((1+\delta')^{(1+\epsilon)k}\log n, (1+\delta')^{(1+\epsilon)k+\epsilon}\log n)$$

for each $k \geq 0$ such that $(1+\delta')^{(1+\epsilon)k+\epsilon}\log n < n/\kappa$, noting that for sufficiently large x the range $(x, (1+\delta')^{\epsilon}x)$ is guaranteed to contain an integer. X has the property that for any $a < b \in X$ we have $(1+\delta')a < b$. Also

$$\begin{aligned}|X| &= \left\{k \geq 0 \middle| (1+\delta')^{(1+\epsilon)k+\epsilon}\log n < n/\kappa\right\} \\ &= \max\left\{k \geq 0 \middle| (1+\delta')^{(1+\epsilon)k-1}\log n < n/\kappa\right\} \\ &\geq \frac{1}{1+\epsilon}\log_{1+\delta'}\frac{n/\kappa}{\log n} \\ &= \frac{1}{1+\epsilon}\log_{1+\delta'}(n/\kappa) - \frac{1}{1+\epsilon}\log_{1+\delta'}(\log n) \\ &> \frac{\log n}{\log(1+\widetilde{\delta})} \\ &> |T_1(n, \mathbb{N})|.\end{aligned}$$

Hence by the pigeon-hole principle for all sufficiently large n there must exist some $n_2 \neq n_2' \in X$ with $\log n < n_2, n_2' < n/\kappa$ such that $T_1(n, n_2) = T_1(n, n_2')$. Without loss of generality say $n_2 < n_2'$, and then we have also that $(1+\delta')n_2 < n_2'$. Suppose for contradiction that $W(n, n_2) \leq \delta' n_2$ and $W(n, n_2') \leq \delta' n_2'$.

We now show that, intuitively speaking, the scheduler when given job sizes (n, n_2') must schedule the second agent's job first.

Claim. $T_1(n, n_2') \geq n + n_2'$.

Proof. Suppose for contradiction that $T_2(n, n_2) \geq T_1(n, n_2')$. Then

$$\begin{aligned} W(n, n_2') &\geq 2T_1(n, n_2') - n - 2n_2 \\ &\geq 2n - n - 2n_2' = n - 2n_2' \\ &> (1 - 2/\kappa)n > \delta' n_2', \end{aligned}$$

a contradiction. Hence $T_1(n, n_2') > T_2(n, n_2')$ and so $T_1(n, n_2') \geq n + n_2'$. □

Now we have

$$\begin{aligned} W(n, n_2) &= T_1(n, n_2) + T_2(n, n_2) - n - 2n_2 \\ &= T_1(n, n_2') + T_2(n, n_2) - n - n_2 \\ &\geq n + n_2' + n_2 - n - 2n_2 = n_2' - n_2 \\ &> \delta' n_2, \end{aligned}$$

a contradiction. Hence in fact either $W(n, n_2) > \delta' n_2$ or $W(n, n_2') > \delta' n_2'$. Either way we have produced n_1, n_2 both arbitrarily large such that

$$\frac{W(n_1, n_2)}{\min(n_1, n_2)} > \delta' > \delta,$$

as required. □

We conclude this Section by summarising the properties of the various schedulers we have considered in Table 3.2.

	$W(n_1, n_2)$	$\max(I_1(n), I_2(n))$
Priority	$n_2 - \min(n_1, n_2)$	∞
Impatient	0	$n + 1$
Cautious	$\max(n_1, n_2) - 1$	1
δ-optimal	$\delta \min(n_1, n_2)$	$2 + \frac{\log n}{\log(1+\delta)}$

Table 3.1: The vital statistics of our schedulers

3.3 Arbitrarily many agents

It is reasonable to wonder at this point why we would ever consider deviating from fair time division multiplexing (which above we have called the cautious scheduler). This prevents any information flow and increases the total waiting time by at worst a factor of two, which is likely to be acceptable in most cases. However, if the resource is to be shared among a large number of agents the picture is far less rosy: essentially, with k agents the cautious scheduler can increase total waiting time by up to a factor of k. In this section we will see that this can be substantially reduced by allowing a small amount of information flow.

Definition 3.13. A *k-scheduler* is a k-tuple $(T_1, \ldots, T_k)$ of functions $\mathbb{N}^k \to \mathbb{N}$.

We will often write $[n]$ for the set $\{1, \ldots, n\}$.

Definition 3.14. A k-scheduler $(T_1, \ldots, T_k)$ is *valid* if for every $\mathbf{n} = n_1, \ldots, n_k \in \mathbb{N}$ there exists a function

$$f : \mathbb{N}^+ \to [k] \cup \{0\}$$

such that for each $i \in [k]$ we have

(i) $|f^{-1}(i)| = n_i$, and

(ii) $\max\{f^{-1}(i)\} \leq T_i(n_1, \ldots, n_k)$.

Once again, there is a straightforward healthiness condition for a k-scheduler to be valid.

Proposition 3.15. *A k-scheduler $(T_1, \ldots, T_k)$ is valid if and only if for all $\mathbf{n} = (n_1, \ldots, n_k)$ and all $I \subseteq [k]$ with $I \neq \emptyset$ we have*

$$\max_{i \in I} (T_i(\mathbf{n})) \geq \sum_{i \in I} n_i.$$

Proof. For any $I \subseteq [k]$, we have

$$\begin{aligned}
\sum_{i \in I} n_i &= \sum_{i \in I} \left| f^{-1}(i) \right| \\
&= \left| \bigcup_{i \in I} f^{-1}(i) \right| \\
&\leq \max \bigcup_{i \in I} f^{-1}(i) \\
&= \max_{i \in I} \left(\max \left\{ f^{-1}(i) \right\} \right) \\
&\leq \max_{i \in I} (T_i(\mathbf{n})) .
\end{aligned}$$

For the converse we proceed by induction on k. The base $k = 1$ is trivial.

Let $T = (T_1, \ldots, T_k)$ be a scheduler satisfying the above condition. Without loss of generality we may assume that $T_k(\mathbf{n}) \geq T_i(\mathbf{n})$ for all i. Now clearly $T' = (T_1, \ldots, T_{k-1})(-, 0)$ also satisfies the condition[1] and so by the inductive hypothesis it is a valid $(k-1)$-scheduler.

Hence there exists a function $f' : \mathbb{N} \to [k-1] \cup \{0\}$ such that for each $i \in [k-1]$ we have that f' satisfies the conditions in Definition 3.14. Now let $f : \mathbb{N}^+ \to [k] \cup \{0\}$ be defined by

$$f(n) = \begin{cases} f'(n) & \text{if } f'(n) \neq 0 \\ k & \text{if } f'(n) = 0 \text{ and } |f'^{-1}(0)| \cap [n] \leq n_k \\ 0 & \text{otherwise.} \end{cases}$$

It is clear that f satisfies the conditions of Definition 3.14 for $i \in [k-1]$, and also that $|f^{-1}(k)| = n_k$. Now we also have that $f(n) \neq 0$ for all $n \leq \max\{f^{-1}(k)\}$, and so

$$\begin{aligned} \max\left\{f^{-1}(k)\right\} &\leq \left|\bigcup_{i\in[k]} f^{-1}(i)\right| \\ &\leq \sum_{i\in[k]} n_i \\ &\leq \max_{i\in[k]} \left(T_i(\mathbf{n})\right), \end{aligned}$$

by the hypothesis of the Proposition with $I = [k]$. □

Once again, we have the priority scheduler, which prioritises jobs by agent number.

Example 3.16. The *priority* k-scheduler is given by

$$T_i(\mathbf{n}) = \sum_{j\leq i} n_j.$$

This is valid since for any $I \subseteq [k]$ we have

$$\max_{i\in I} \left(T_i(\mathbf{n})\right) = T_{\max(I)}(\mathbf{n}) = \sum_{i\leq \max(I)} n_i \geq \sum_{i\in I} n_i.$$

This has total waiting time

$$(T_1 + \ldots + T_k)(\mathbf{n}) = \sum_{i\in[k]} \sum_{j\leq i} n_j = \sum_{i\in[k]} (k+1-i) n_i.$$

As before, we can save time by scheduling the larger jobs first.

[1] This notation means that $T'_i(n_1, \ldots, n_{k-1}) = T_i(n_1, \ldots, n_{k-1}, 0)$ for all $i \in [k-1]$.

Example 3.17. The *impatient* k-scheduler is given by

$$T_i(\mathbf{n}) = \sum_{j:(n_j<n_i)\vee((n_j=n_i)\wedge(j\leq i))} n_j.$$

This has total waiting time

$$\sum_{i\in[k]} T_i(\mathbf{n}) = \sum_{i\in[k]} i \times \mathbf{sort}(\mathbf{n})_i,$$

where $\mathbf{sort}(\mathbf{n})$ is $\mathbf{n}$ sorted into descending order.

Once again we have that this is the minimum possible time.

Proposition 3.18. *Let $(T_1, \ldots, T_k)$ be a valid k-scheduler. Then we have*

$$\sum_{i\in[k]} T_i(\mathbf{n}) \geq \sum_{i\in[k]} i \times \mathbf{sort}(\mathbf{n})_i.$$

This optimality is essentially due to the *rearrangement inequality*.

Lemma 3.19 (Rearrangement inequality)**.** *Let $(a_1, \ldots, a_n)$ and $(b_1, \ldots, b_n)$ be sequences of real numbers in ascending order. Let $\sigma \in S_n$ be any permutation. Then we have*

$$\sum_{i=1}^{n} a_i b_{n+1-i} \leq \sum_{i=1}^{n} a_i b_{\sigma(i)} \leq \sum_{i=1}^{n} a_i b_i.$$

Proof. [33], Theorem 368. □

Proof of Proposition 3.18. Let $i_1, i_2, \ldots, i_k \in [k]$ be distinct such that $T_{i_1}(\mathbf{n}) \leq T_{i_2}(\mathbf{n}) \leq \ldots \leq T_{i_k}(\mathbf{n})$. Then by Proposition 3.15 we have

$$T_{i_j}(\mathbf{n}) = \max_{i\in\{i_1,\ldots,i_j\}} (T_i(\mathbf{n})) \geq \sum_{i\in\{i_1,\ldots,i_j\}} n_i.$$

Hence

$$\begin{aligned}\sum_{i\in[k]} T_i(\mathbf{n}) &\geq \sum_{j\in[k]} \sum_{i\in\{i_1,\ldots,i_j\}} n_i \\ &= \sum_{j\in[k]} j \times n_{i_j} \\ &\geq \sum_{i\in[k]} i \times \mathbf{sort}(\mathbf{n})_i,\end{aligned}$$

by the rearrangement inequality. □

Again we measure efficiency by the excess of waiting time over this minimum.

Definition 3.20. Let $T = (T_1, \ldots, T_k)$ be a valid k-scheduler. The *overhead* $W : \mathbb{N}^k \to \mathbb{N}$ of T is given by

$$W(\mathbf{n}) = \sum_{i \in [k]} T_i(\mathbf{n}) - \sum_{i \in [k]} i \times \mathbf{sort}(\mathbf{n})_i.$$

As before we may be worried about the amount of information that can reach a rogue agent about the behaviour of the others.

Definition 3.21. Let $(T_1, \ldots, T_k)$ be a k-scheduler. Then the *observation count functions* $I_1, \ldots, I_k$ are given by

$$I_j(n) = |T_j(\mathbb{N}, \ldots, \mathbb{N}, n, \mathbb{N}, \ldots, \mathbb{N})|,$$

where in the above n is the jth argument.

Once again the amount of information obtained by agent k requesting a job of size at most N is

$$\max_{n \leq N} \log I_k(n).$$

For the impatient scheduler, we have

$$\begin{aligned} I_j(n) &= |\{n, n+1, \ldots, jn + (k-j)(n-1)\}| \\ &= (k-1)n + 1 - (k-j). \end{aligned}$$

On the other hand if we wish to avoid any information flow we can use fair time division multiplexing.

Example 3.22. The *cautious* k-scheduler is given by $T_i(\mathbf{n}) = kn_i$.

This has $I_j(n) = 1$ for all n, but overhead

$$W(\mathbf{n}) = \sum_{i \in [k]} (k-i) \times \mathbf{sort}(\mathbf{n})_i.$$

The disadvantages of fair time division multiplexing now become apparent. For instance, we have

$$W((n, 0, \ldots, 0)) = (k-1)n.$$

This is extremely bad if k is large. However, this kind of behaviour is unavoidable for schedulers with no information flow.

Proposition 3.23. *Let $(T_1, \ldots, T_k)$ be a valid k-scheduler, with $I_j(n) = 1$ for all n and all $j \in [k]$. Then*

$$\lim_{N\to\infty} \sup_{n_1,\ldots,n_k>M} \frac{(T_1 + \ldots + T_k)(n_1, \ldots, n_k)}{n_1 + \ldots + n_k} \geq k.$$

Proof. As in the proof of Proposition 3.10, we treat each T_i as a function of a single variable, and assume that there exists some $C < k$ and some N such that for all $n_1, \ldots, n_k \geq N$ we have $T_1(n_1) + \ldots + T_k(n_k) < C(n_1 + \ldots + n_k)$. Now let

$$n = \left\lceil \frac{C(k-1)}{k-C} \right\rceil + 1.$$

Without loss of generality we have $T_1(n) \geq T_j(n)$ for all $j \neq 1$, and hence $T_1(n) \geq kn$. But then

$$\begin{aligned} T_1(n) + T_2(N) + \ldots + T_k(N) &> kn \\ &= C(n + (k-1)N) + n(k-C) - (k-1)CN \\ &> C(n + (k-1)N) + (k-1)CN - (k-1)CN \\ &= C(n + (k-1)N), \end{aligned}$$

which is a contradiction. □

Again, however, we can trade off some small amount of information flow for much lower overhead. As before the idea is to round off the output given to each agent to the next power of $(1 + \delta)$.

Example 3.24. The *δ-optimal* k-scheduler is given by

$$T_i(\mathbf{n}) = n_i + \min \left\{ \left\lceil (1+\delta)^k \right\rceil \middle| k \in \mathbb{N}, \left\lceil (1+\delta)^k \right\rceil \geq \sum_{j:(n_j<n_i)\vee((n_j=n_i)\wedge(j<i))} n_j \right\}.$$

For this scheduler we have in the same way as before that

$$I_j(n) \leq 2 + \log_{1+\delta}(k-1)n \sim \log_{1+\delta} n.$$

Hence the amount of information in bits obtained by any agent requesting a job of size at most n is

$$\begin{aligned} \log I_j(n) &= \log\log n - \log\log(1+\delta) + o(1) \\ &\approx \log\log n + \log \delta^{-1} + o(1) \end{aligned}$$

as $n \to \infty$ (where the approximation is from a Taylor expansion for small δ).

To compute the overhead, let σ be the permutation of $[k]$ defined by $\sigma(j) < \sigma(i)$ if and only if either $n_j < n_i$ or $n_j = n_i$ and $j < i$. Then we have $\sum_{i\in[k]} i \times \mathbf{sort}(\mathbf{n})_i = \sum_{i\in[k]} \sum_{j:\sigma(j)\leq\sigma(i)} n_j$, and so

$$\begin{aligned} W(\mathbf{n}) &= \sum_{i\in[k]} \left[T_{\sigma(i)}(\mathbf{n}) - \sum_{j:\sigma(j)\leq\sigma(i)} n_j \right] \\ &\leq \sum_{i\in[k]} \left[n_{\sigma(i)} + (1+\delta) \sum_{j:\sigma(j)<\sigma(i)} n_j - \sum_{j:\sigma(j)\leq\sigma(i)} n_j \right] \\ &= \delta \sum_{i\in[k]} \sum_{j:\sigma(j)<\sigma(i)} n_j \\ &= \delta \sum_{i\in[k]} (i-1) \times \mathbf{sort}(\mathbf{n})_i \end{aligned}$$

Note in particular that we have

$$T_{\delta-\mathrm{opt}}(\mathbf{n}) < (1+\delta)\, T_{\min}(\mathbf{n}),$$

where $T_{\delta-\mathrm{opt}}$ and $T_{\min}$ denote the total waiting times of the δ-optimal and the impatient scheduler respectively.

Again this is essentially optimal.

Theorem 3.25. *Let $(T_1, \ldots, T_k)$ be a valid k-scheduler with $k \geq 2$, and let $\delta > 0$. Suppose that*

$$\min_{j\in[k]} \limsup_{n\to\infty} \frac{I_j(n)}{\log n} < \frac{1}{\log(1+\delta)}.$$

Then

$$\lim_{N\to\infty} \sup_{n_1,\ldots,n_k>N} \frac{W(\mathbf{n})}{\sum_{i\in[k]} (i-1) \times \mathbf{sort}(\mathbf{n})_i} > \delta.$$

The proof will be by induction on k. Given a k-scheduler T with small information flow, we will produce a $k-1$-scheduler T' by setting the final input to zero. This cannot have information flow greater than the original scheduler, and so by the inductive hypothesis we will have that the overhead of T' is bounded below as in the Theorem. We will then use the family of examples on which T' has not too high efficiency to construct a family on which T has not too high efficiency.

However, in order for this final step to work we require a technical lemma showing that, essentially, we may assume that setting one or more job sizes to zero does not increase the waiting time.

Lemma 3.26. *Let $T = (T_1, \ldots, T_k)$ be a valid k-scheduler with observation count functions $I_1, \ldots, I_k$. Then there exists a valid k-scheduler $T' = (T'_1, \ldots, T'_k)$, with observation count functions $I'_1, \ldots, I'_k$, such that*

(i) $\sum_{i\in[k]} T'_i(\mathbf{n}) \leq \sum_{i\in[k]} T_i(\mathbf{n})$ *for all* $\mathbf{n} \in \mathbb{N}^k$,

(ii) For all $j \in [k]$ *and all* $n \geq 1$ *we have* $I'_j(n) \leq I_j(n)$, *and*

(iii) For any $\mathbf{n} \in \mathbb{N}^k$ *and any* $\mathbf{n}' \in \mathbb{N}^k$ *such that for each* i *we have either* $n'_i = n_i$ *or* $n'_i = 0$, *we have* $\sum_{i\in[k]} T'_i(\mathbf{n}') \leq \sum_{i\in[k]} T'_i(\mathbf{n}')$.

Proof. We show by induction on l that there exists such a function T' where in the third condition above we require $n'_i = n_i$ for all $i \in [k-l]$. The base case $l = 0$ is trivial: take $T' = T$.

For the inductive step let T' be as above. Let T'' be a k-scheduler with $T''(\mathbf{n}) = T'(\mathbf{n})$ whenever $n_{k-l} \neq 0$, and

$$T''(n_1, n_{k-l-1}, 0, n_{k-l+1}, \ldots, n_k) = T'(n_1, \ldots, n_{k-l-1}, n, n_{k-l+1}, \ldots, n_k),$$

where

$$n = \arg\min_{n\in\mathbb{N}} \sum_{i\in k} T'_i(n_1, \ldots, n_{k-l-1}, n, n_{k-l+1}, \ldots, n_k).$$

The first and third conditions above are now clear. For the second, observe that for any $i \neq k-l$ and any $n \in \mathbb{N}$ we have $\{T''_i(\mathbf{n}) | n_i = n\} \subseteq \{T'_i(\mathbf{n}) | n_i = n\}$, and also for any $n \geq 1$ we have $\left\{T''_{k-l}(\mathbf{n}) \middle| n_{k-l} = n\right\} = \left\{T'_{k-l}(\mathbf{n}) \middle| n_{k-l} = n\right\}$. □

We are now ready to prove Theorem 3.25.

Proof of Theorem 3.25. Induction on k. The base case $k = 2$ is Theorem 3.12. For the inductive step, suppose that $T = (T_1, \ldots, T_k)$ is a valid k-scheduler, such that, without loss of generality,

$$\limsup_{n\to\infty} \frac{I_1(n)}{\log n} < \frac{1}{\log(1+\delta)}.$$

By Lemma 3.26 we may assume that T has the property that $\sum_{i\in[k]} T_i(n_1, \ldots, n_{k-1}, 0) \leq \sum_{i\in[k]} T_i(n_1, \ldots, n_k)$ for any $\mathbf{n} = (n_1, \ldots, n_k)$.

Plainly $T' = (T_1, \ldots, T_{k-1})(-, 0)$ is a valid $(k-1)$-scheduler. Let T' have overhead function W' and observation count functions $I'_1, \ldots, I'_{k-1}$. Now $I'_1(n) \leq I_1(n)$ and so by the inductive hypothesis we have that

$$\lim_{N\to\infty} \sup_{n_1,\ldots,n_{k-1}>N} \frac{W'(\mathbf{n})}{\sum_{i\in[k-1]}(i-1) \times \mathbf{sort}(\mathbf{n})_i} > \delta.$$

Hence there exists some $\delta' > \delta$ and some $\mathbf{n}^{(1)}, \mathbf{n}^{(2)}, \ldots \in \mathbb{N}^{k-1}$ with $\min_{i\in[k-1]} n_i^{(j)} \to \infty$ as $j \to \infty$ such that for each j we have

$$W'\left(\mathbf{n}^{(j)}\right) > \delta' \sum_{i\in[k-1]} (i-1) \times \mathbf{sort}\left(\mathbf{n}^{(j)}\right)_i .$$

Also there exists some δ'' with $\delta' > \delta'' > \delta$ such that for sufficiently large j we have

$$\delta' \sum_{i\in[k-1]} (i-1)\times\mathbf{sort}\left(\mathbf{n}^{(j)}\right)_i > \delta'' \sum_{i\in[k-1]} (i-1)\times\mathbf{sort}\left(\mathbf{n}^{(j)}\right)_i + (k+(k-1)\delta'')\sqrt{\min_{i\in[k-1]} n_i^{(j)}}.$$

Now let $\widetilde{\mathbf{n}}^{(j)}$ be given by

$$\widetilde{n}_i^{(j)} = \begin{cases} n_i^{(j)} & \text{if } 1 \le i \le k-1 \\ \sqrt{\min_{i\in[k-1]} n_i^{(j)}} & \text{if } i = k. \end{cases}.$$

Now we have

$$\begin{aligned}
\sum_{i\in[k-1]} T_i'\left(n_1^{(j)}, \ldots, n_{k-1}^{(j)}\right) &= \sum_{i\in[k-1]} T_i\left(n_1^{(j)}, \ldots, n_{k-1}^{(j)}, 0\right) \\
&\le \sum_{i\in[k]} T_i\left(n_1^{(j)}, \ldots, n_{k-1}^{(j)}, 0\right) \\
&\le \sum_{i\in[k]} T_i\left(n_1^{(j)}, \ldots, n_{k-1}^{(j)}, \widetilde{n}_k^{(j)}\right) \\
&= \sum_{i\in[k]} T_i\left(\widetilde{\mathbf{n}}^{(j)}\right),
\end{aligned}$$

where the final inequality is by the assumption derived from Lemma 3.26.

Hence

$$\begin{aligned}
W\left(\widetilde{\mathbf{n}}^{(j)}\right) &\ge W'\left(\mathbf{n}^{(j)}\right) - \sum_{i\in[k]} i \times \mathbf{sort}\left(\widetilde{\mathbf{n}}^{(j)}\right)_i + \sum_{i\in[k-1]} i \times \mathbf{sort}\left(\mathbf{n}^{(j)}\right)_i \\
&= W'\left(\mathbf{n}^{(j)}\right) - k\widetilde{n}_k^{(j)} \\
&> \delta' \sum_{i\in[k-1]} (i-1) \times \mathbf{sort}\left(\mathbf{n}^{(j)}\right)_i - k\sqrt{\min_{i\in[k-1]} n_i^{(j)}} \\
&> \delta'' \left(\sum_{i\in[k-1]} (i-1) \times \mathbf{sort}\left(\mathbf{n}^{(j)}\right)_i + (k-1)\sqrt{\min_{i\in[k-1]} n_i^{(j)}} \right) \\
&\ge \delta'' \sum_{i\in[k]} (i-1) \times \mathbf{sort}\left(\widetilde{\mathbf{n}}^{(j)}\right),
\end{aligned}$$

as required. □

3.4 Conclusions

The objective of this chapter was to illustrate the purpose of quantifying information flow by showing the efficiency gains that can be obtained by allowing a small amount of information flow between agents. We have shown that whereas the cautious scheduler for many agents can have extremely large efficiency overhead (and by Proposition 3.23 so can any scheduler which does not allow any information flow), we can make the efficiency overhead arbitrarily small by allowing information flow of approximately $\log \log n$.

We have considered only the threat model of a single compromised user attempting to reason about the rest of the system, whereas in reality it may be that several users collude. We might therefore want to consider more detailed information flow policies, constraining the amounts of information that can flow between arbitrary sets of users. This would also allow us to model the case where each user is permitted to make a number of initial requests, since we could treat these as made by separate agents who share information.

Moreover, this toy example covers only the highly unrealistic situation that each agent makes a one-time request for use of the resource, at the beginning of the interaction. More generally, a shared system may allow its users to interact with it throughout its execution, and the purpose of the remainder of this thesis will be to model this situation, and ultimately to calculate or approximate the rate at which a given system allows information to flow.

Chapter 4

Information-theoretic foundations

4.1 Introduction

In this chapter we set out the information-theoretic foundations for our study of information flow, which we will then proceed to apply in Chapter 5. In Section 4.2, we review some basic definitions in information theory which we will use in this chapter, namely entropy, conditional entropy, mutual information and conditional mutual information. These definitions are taken from [19].

In Section 4.3 we discuss how to model information flow in a system requiring choices to be made by both the high-level user (whom we call Alice) and the low-level user (whom we call Bob). We adopt the consensus definition involving mutual information and relate it to the existence of a low-error coding scheme (Theorem 4.11). We also show (Proposition 4.7) that we may assume that Bob chooses a purely deterministic strategy.

In Section 4.4 we move to considering the special case of deterministic systems: that is, systems where the output seen by Bob is a deterministic function of the strategies chosen by Alice and Bob. We show in Theorem 4.12 that in this case the information flow is governed by the size of the largest set of outputs that are consistent with a single strategy for Bob; this simplifies calculation considerably, as we will see in the next chapter.

In Section 4.5 we consider the objections that have been raised by Smith in [63] to the use of mutual information as the appropriate information-theoretic definition for information flow. We explain what we consider to be the true underlying issues causing these problems, and suggest that the key to resolving them is essentially to take logarithms after taking expectation rather than before. We define a few possible quantities, all of which do this and therefore address the problems with

mutual information, in Definition 4.16 and Definition 4.17. We refer to these as *LogSumExp* measures of information flow.

In Section 4.6 we establish some mathematical properties of our new quantities, and in particular we show in Theorem 4.24 that for deterministic systems they agree with mutual information. Finally, in Section 4.7, we use the theory of g-leakage to show that one of our quantities precisely measures the 'Dalenius' leakage.

4.2 Information and entropy

The most fundamental concept in the mathematics of information is the notion of *entropy*. This measures the 'uncertainty' in a random variable.

Definition 4.1. Let X be a random variable. The *entropy* $H(X)$ of X is defined by

$$H(X) = \mathbb{E} \log p_X(X) = \sum_{x \in \mathcal{X}} -p_X(x) \log p_X(x),$$

where $\mathcal{X}$ is the set of outcomes for X, and by convention we take $0 \log 0 = 0$.

Note that we may consider the logarithm to be taken base 2, in which case entropy is measured in bits (also known as shannons), or base e, in which case it is measured in 'nats'. For the majority of this thesis we will leave this unspecified, but where it is necessary to choose we will generally consider logarithms base 2, and thus information measured in bits.

If Y is another random variable then the uncertainty in X after observing that $Y = y$ is $H(X|Y = y) = \mathbb{E}_X \log p_{X|Y}(X|y)$. The expected value of this quantity is called the *conditional entropy*.

Definition 4.2. Let X and Y be random variables. The *conditional entropy* $H(X|Y)$ is defined by

$$\begin{aligned} H(X|Y) &= \sum_{y \in \mathcal{Y}} p_Y(y) H(X|Y = y) \\ &= \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} -p_Y(y) p_{X|Y}(x|y) \log p_{X|Y}(x|y) \\ &= \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} -p_{X,Y}(x,y) \log \frac{p_{X,Y}(x,y)}{p_Y(y)}. \end{aligned}$$

The difference between the prior uncertainty in X and the expected remaining uncertainty after observing Y is a measure of the amount of information about X contained in Y. This is called the *mutual information* between X and Y.

Definition 4.3. Let X and Y be random variables. The *mutual information* between X and Y, denoted $I(X\,;Y)$, is given by

$$\begin{aligned} I(X\,;Y) &= H(X) - H(X|Y) \\ &= \sum_{x\in\mathcal{X}} -p_X(x)\log p_X(x) - \sum_{x\in\mathcal{X},y\in\mathcal{Y}} -p_{X,Y}(x,y)\log\frac{p_{X,Y}(x,y)}{p_Y(y)} \\ &= \sum_{x\in\mathcal{X},y\in\mathcal{Y}} p_{X,Y}(x,y)\log\frac{p_{X,Y}(x,y)}{p_X(x)p_Y(y)}. \end{aligned}$$

From the final line of the above it is immediate that mutual information is symmetric; that is, that $I(X\,;Y) = I(Y\,;X)$ for all random variables X and Y. Note that while $H(X) - H(X|Y=y)$ may be negative for some values of y, we have that $I(X\,;Y) \geq 0$ for all random variables X and Y, with equality if and only if X and Y are independent ([19], corollary to Theorem 2.6.3).

Mutual information also has a conditional version, *conditional mutual information.*

Definition 4.4. Let X, Y and Z be random variables. Then the *conditional mutual information* of X and Y given Z is given by

$$\begin{aligned} I(X\,;Y|Z) &= H(X|Z) - H(X|Y,Z) \\ &= \sum_{x\in\mathcal{X},z\in\mathcal{Z}} -p_{X,Z}(x,z)\log\frac{p_{X,Z}(x,z)}{p_Z(z)} \\ &\quad - \sum_{x\in\mathcal{X},y\in\mathcal{Y},z\in\mathcal{Z}} -p_{X,Y,Z}(x,y,z)\log\frac{p_{X,Y,Z}(x,y,z)}{p_{Y,Z}(y,z)} \\ &= \sum_{x\in\mathcal{X},y\in\mathcal{Y},z\in\mathcal{Z}} p_{X,Y,Z}(x,y,z)\log\frac{p_{X,Y,Z}(x,y,z)p_Z(z)}{p_{Y,Z}(y,z)p_{X,Z}(x,z)} \\ &= \sum_{x\in\mathcal{X},y\in\mathcal{Y},z\in\mathcal{Z}} p_{X,Y,Z}(x,y,z)\log\frac{p_{X,Y|Z}(x,y|z)}{p_{X|Z}(x|z)p_{Y|Z}(y|z)}. \end{aligned}$$

There are a couple of elementary results about entropy which we will use later in this chapter. The first is the *chain rule for entropy* ([19], Theorem 2.5.1).

Lemma 4.5 (Chain rule for entropy)**.** *Let $X_1,\ldots,X_n$ be random variables. Then*

$$H(X_1,\ldots,X_n) = \sum_{i=1}^{n} H(X_i|X_1,\ldots,X_{i-1}).$$

Combining this with the fact that conditioning reduces entropy ([19], Theorem 2.6.5) gives the *independence bound on entropy* ([19], Theorem 2.6.6).

Lemma 4.6 (Independence bound for entropy)**.** *Let $X_1, \ldots, X_n$ be random variables. Then*

$$H(X_1, \ldots, X_n) \leq \sum_{i=1}^{n} H(X_i).$$

4.3 What is information flow?

We shall assume throughout that information is flowing from one user, Alice, assumed to be in possession of some secret, to another user Bob. In practice several different situations are possible: Alice may be a spy collaborating with Bob, or Alice may be an innocent victim being snooped on. Throughout this work, however, we shall make the conservative assumption that Alice is co-operating with Bob.

The simplest situation is that Bob is purely passive: Alice selects an action according to a chosen distribution function p_X, and the system displays a result to Bob according to some fixed conditional distribution $p_{Y|X}$ (where the matrix $p_{Y|X}(y|x)$ is the system specification).

Note that this is sufficient to model a system which is interactive from the perspective of Alice: her 'action' is in fact a strategy, determining what action she will take at each moment based on what she has seen so far.

This situation is just a noisy communication channel from Alice to Bob. Averaging over repeated rounds, Shannon's classical noisy coding theorem ([62], Theorem 11) gives that the rate at which information can be transmitted from Alice to Bob (with a suitable coding scheme) is given by the mutual information between X and Y,

$$\begin{aligned} I(Y\,;X) &= H(X) - H(X|Y) \\ &= \sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p_{X,Y}(x,y) \log \frac{p_{X,Y}(x,y)}{p_X(x)p_Y(y)} \\ &= \mathbb{E}_{X,Y} \log \frac{p_{X,Y}(X,Y)}{p_X(X)p_Y(Y)}. \end{aligned}$$

Hence for a fixed system specification, the maximum amount of information that can be conveyed from Alice to Bob per round is just

$$\max_{p_X} I(Y\,;X).$$

Note that this is precisely the expression for channel capacity in [62].

For example, consider a system in which Alice selects either 0 or 1, and the system gives Bob an output which with probability $\frac{1}{2}$ is Alice's input and with probability $\frac{1}{2}$ is 'X'. The specification of this system is shown in Figure 4.1.

	Alice	
	0	1
Bob	0: 0.5 X: 0.5	1: 0.5 X: 0.5

Figure 4.1: A simple non-interactive system

Suppose that Alice chooses 1 with probability p and 0 with probability $1-p$ (so $p_X(1) = p$ and $p_X(0) = 1-p$). Then the marginal distribution of the output Y is $p_Y(\mathrm{X}) = \frac{1}{2}$, $p_Y(1) = \frac{p}{2}$ and $p_Y(0) = \frac{1-p}{2}$. The joint distribution on X, Y is $p_{X,Y}(1,\mathrm{X}) = p_{X,Y}(1,1) = \frac{p}{2}$ and $p_{X,Y}(0,\mathrm{X}) = p_{X,Y}(0,0) = \frac{1-p}{2}$, with other values 0. Thus we have that the conditional mutual information is

$$\begin{aligned}
I(Y\,;X) &= p_{X,Y}(1,\mathrm{X}) \log \frac{p_{X,Y}(1,\mathrm{X})}{p_X(1)p_Y(\mathrm{X})} + p_{X,Y}(1,1) \log \frac{p_{X,Y}(1,1)}{p_X(1)p_Y(1)} \\
&\qquad + p_{X,Y}(0,\mathrm{X}) \log \frac{p_{X,Y}(0,\mathrm{X})}{p_X(0)p_Y(\mathrm{X})} + p_{X,Y}(0,0) \log \frac{p_{X,Y}(0,0)}{p_X(0)p_Y(0)} \\
&= \frac{p}{2} \log \frac{\frac{p}{2}}{p \cdot \frac{1}{2}} + \frac{p}{2} \log \frac{\frac{p}{2}}{p \cdot \frac{p}{2}} \\
&\qquad + \frac{1-p}{2} \log \frac{\frac{1-p}{2}}{(1-p) \cdot \frac{1}{2}} + \frac{1-p}{2} \log \frac{\frac{1-p}{2}}{(1-p) \cdot \frac{1-p}{2}} \\
&= 0 - \frac{p}{2} \log p + 0 - \frac{1-p}{2} \log(1-p) \\
&= \frac{1}{2}(-p \log p - (1-p) \log(1-p)) \\
&= \frac{1}{2} H_{\mathrm{b}}(p),
\end{aligned}$$

where H_{b} is the *binary entropy function*. This is well known to be maximised at $p = \frac{1}{2}$, and so we have that

$$\max_{p_X} I(X\,;Y) = \frac{1}{2} H_{\mathrm{b}}\left(\frac{1}{2}\right) = \frac{1}{2}$$

(if the logs are taken base 2, and hence information is measured in bits).

Note that this is exactly equal to the capacity of the *binary erasure channel* with erasure probability $\frac{1}{2}$ (see [19], Section 7.1.5). This is as expected since this is precisely what this system simulates. Note that the terms which evaluate to 0 in the expression for $I(Y\,;X)$ correspond to the fact that when Bob sees an output of 'X' he learns nothing about Alice's input.

More generally, however, Bob may be able to interact with the system. Then what he must choose is a 'strategy', determining what he will do at each moment,

conditional on what he has seen so far; Alice must do likewise. These strategies may in general be probabilistic, but by viewing all randomness as being pre-determined we may view Alice and Bob as choosing deterministic strategies, drawn from finite spaces $\mathcal{X}_A$ and $\mathcal{X}_B$. We denote the corresponding independent random variables by X_A, X_B, and write $X = (X_A, X_B)$.

Once again the system produces to Bob an output Y according to a specification matrix $p_{Y|X}(y|x)$. The rate at which information can be conveyed from Alice to Bob is now governed by the conditional mutual information,

$$\begin{aligned} I(Y\,;X_A|X_B) &= H(X_A|X_B) - H(X_A|Y,X_B) \\ &= H(X_A) - H(X_A|Y,X_B) \\ &= \sum_{\substack{y\in\mathcal{Y},x_A\in\mathcal{X}_A,\\ x_B\in\mathcal{X}_B}} p_{Y,X_A,X_B}(y,x_A,x_B)\log\frac{p_{Y,X_A|X_B}(y,x_A|x_B)}{p_{Y|X_B}(y|x_B)p_{X_A|X_B}(x_A|x_B)} \\ &= \sum_{\substack{y\in\mathcal{Y},x_A\in\mathcal{X}_A,\\ x_B\in\mathcal{X}_B}} p_{X_A}(x_A)p_{X_B}(x_B)p_{Y|X_A,X_B}(y|x_A,x_B)\log\frac{p_{Y|X_A,X_B}(y|x_A,x_B)}{p_{Y|X_B}(y|x_B)} \\ &= \mathbb{E}_{X_A,X_B,Y}\log\frac{p_{Y|X_A,X_B}(Y|X_A,X_B)}{p_{Y|X_B}(Y|X_B)}, \end{aligned} \tag{4.1}$$

where the second and fourth equalities are by the independence of X_A and X_B.

We shall justify this shortly, but first we observe that in fact there is no reason for Bob to choose a probabilistic strategy.

Proposition 4.7. *Let p_{X_A}, p_{X_B} be probability distributions over finite sets, and let $p_{Y|X}$ be fixed. Then there exists some probability distribution p'_{X_B} with $p'_{X_B}(x_B) \in \{0,1\}$ for all $x_B \in \mathcal{X}_B$ such that*

$$\begin{aligned} &\sum_{\substack{y\in\mathcal{Y},x_A\in\mathcal{X}_A,\\ x_B\in\mathcal{X}_B}} p_{X_A}(x_A)p_{X_B}(x_B)p_{Y|X_A,X_B}(y|x_A,x_B)\log\frac{p_{Y|X_A,X_B}(y|x_A,x_B)}{p_{Y|X_B}(y|x_B)} \\ &\qquad\leq \sum_{\substack{y\in\mathcal{Y},x_A\in\mathcal{X}_A,\\ x_B\in\mathcal{X}_B}} p_{X_A}(x_A)p'_{X_B}(x_B)p_{Y|X_A,X_B}(y|x_A,x_B)\log\frac{p_{Y|X_A,X_B}(y|x_A,x_B)}{p_{Y|X_B}(y|x_B)} \end{aligned}$$

Proof. Choose x_B so as to maximise

$$\sum_{y\in\mathcal{Y},x_a\in\mathcal{X}_A} p_{X_A}(x_A)p_{Y|X_A,X_B}(y|x_A,x_B)\log\frac{p_{Y|X_A,X_B}(y|x_A,x_B)}{p_{Y|X_B}(y|x_B)}.$$

Set $p'_{X_B}(x_B) = 1$ and $p'_{X_B}(x) = 0$ for $x \neq x_B$, and the result is clear. □

Hence we obtain the fundamental definition of information flow for interactive systems.

Definition 4.8. Let $\mathcal{X}_A, \mathcal{X}_B, \mathcal{Y}$ be any finite sets, and let $M = p_{Y|X_A,X_B}(y|x_A, x_B)$ be any stochastic matrix. Then the *information flow permitted by* M is

$$I_M = \max_{x_B \in \mathcal{X}_B} \max_{p_{X_A}} I(Y\,; X_A | X_B = x_B).$$

For example, consider a system in which Alice selects an input x_A either 0 or 1, as before, but now Bob also selects an input x_B. The output is 'X' with probability $\frac{1}{2}$, and is $x_A x_B$ with probability $\frac{1}{2}$. This system is shown in Figure 4.2.

		Alice: 0	Alice: 1
Bob	0	0: 0.5 X: 0.5	0: 0.5 X: 0.5
	1	0: 0.5 X: 0.5	1: 0.5 X: 0.5

Figure 4.2: A simple interactive specification

If $X_B = 0$ then Y and X_A are independent, and so $I(Y\,; X_A | X_B = 0) = 0$. Hence information flow is maximised by $X_B = 1$. In this case the system reduces to that shown in Figure 4.1, and so we have

$$\max_{x_B \in \mathcal{X}_B} \max_{p_{X_A}} I(Y\,; X_A | X_B = x_B) = \frac{1}{2}.$$

To justify our use of conditional mutual information in this way we will prove a version of Shannon's theorem for the case where Bob also chooses an action.

Definition 4.9. Let X, Y, Z be random variables. We say that X, Y, Z *form a Markov chain*, and write $X \to Y \to Z$ if Z depends only on the value of Y, and conditional on this is independent of X. That is, if $p_{X,Z|Y}(x, z|y) = p_{X|Y}(x|y) p_{Z|Y}(z|y)$.

Observe that this definition is symmetrical in X and Z, so we have $X \to Y \to Z$ if and only if $Z \to Y \to X$.

If $X \to Y \to Z$ then in particular Z cannot reveal more information about X than Y does. This fact is called the *data-processing inequality* ([19], Theorem 2.8.1).

Lemma 4.10 (Data-processing inequality)**.** *Let X, Y, Z be random variables such that $X \to Y \to Z$. Then*

$$I(X\,; Y) \geq I(X\,; Z).$$

We are now able to prove a precise analogue of the noisy coding theorem ([62], Theorem 11). The statement is that if Z is a source with entropy $H \leq I_M$ then there exists a coding system such that n repeated values from X can be transmitted by n repeated instances of the interaction with arbitrarily low error probability as $n \to \infty$. On the other hand if $H > I_M$ then the error probability is bounded away from 0.

Theorem 4.11. *Let Z be a source with entropy $H = H(Z)$, and let $\mathcal{X}_A, \mathcal{X}_B, \mathcal{Y}$ and M be as in Definition 4.8. Then there exists a coding system to transmit outputs of the source with arbitrarily small error probability if and only if $H \leq I_M$.*

Proof. If $H \leq I_M$ then fix $x_B \in \mathcal{X}_B$ so as to maximise $\max_{p_{X_A}} I(Y\,;X_A|X_B = x_B)$. Once x_B is fixed we have a non-interactive system, which as discussed above is equivalent to a noisy channel and so Shannon's theorem gives the result.

For the converse, we will write $Z^{(n)} = Z_1, \ldots, Z_n$ for the sequence of n outputs from the source, and similarly let $X_A^{(n)} = (X_A)_1, \ldots, (X_A)_n$ be the n strategy choices made by Alice, $X_B^{(n)} = (X_B)_1, \ldots, (X_B)_n$ be the choices made by Bob and $Y^{(n)} = Y_1, \ldots, Y_n$ be the sequence of outputs produced by the system.

For any fixed $x_B^{(n)} \in \mathcal{X}_B^n$ we have conditional on $X_B^{(n)} = x_B^{(n)}$ that $Z^{(n)} \to X_A^{(n)} \to Y^{(n)}$, and so also $Y^{(n)} \to X_A^{(n)} \to Z^{(n)}$. Hence by the data-processing inequality we have $I(Y^{(n)}\,;Z^{(n)}|X_B^{(n)} = x_B^{(n)}) \leq I(Y^{(n)}\,;X_A^{(n)}|X_B^{(n)} = x_B^{(n)})$, and taking expectation with respect to $X_B^{(n)}$ gives

$$I(Y^{(n)}\,;Z^{(n)}|X_B^{(n)}) \leq I(Y^{(n)}\,;X_A^{(n)}|X_B^{(n)}).$$

Now, imitating the proof of Lemma 8.9.2 in [19], we have

$$\begin{aligned} I(Y^{(n)}\,;X_A^{(n)}|X_B^{(n)}) &= H(Y^{(n)}|X_B^{(n)}) - H(Y^{(n)}|X_A^{(n)}, X_B^{(n)}) \\ &= H(Y^{(n)}|X_B^{(n)}) - \sum_{i=1}^{n} H(Y_i|Y_1, \ldots, Y_{i-1}, X_A^{(n)}, X_B^{(n)}), \end{aligned}$$

by the chain rule for entropy (Lemma 4.5).

Since Y_i depends only on $(X_A)_i$ and $(X_B)_i$, and is conditionally independent of

everything else, we have $H(Y_i|Y_1,\ldots,Y_{i-1},X_A^{(n)},X_B^{(n)}) = H(Y_i|(X_A)_i,(X_B)_i)$. Hence

$$\begin{aligned} I(Y^{(n)}\,;X_A^{(n)}|X_B^{(n)}) &= H(Y^{(n)}|X_B^{(n)}) - \sum_{i=1}^n H(Y_i|(X_A)_i,(X_B)_i) \\ &\leq \sum_{i=1}^n H(Y_i|X_B^{(n)}) - \sum_{i=1}^n H(Y_i|(X_A)_i,(X_B)_i) \\ &= \sum_{i=1}^n H(Y_i|(X_B)_i) - \sum_{i=1}^n H(Y_i|(X_A)_i,(X_B)_i) \\ &= \sum_{i=1}^n I(Y_i\,;(X_A)_i|(X_B)_i), \end{aligned}$$

where the inequality is by the independence bound for entropy (Lemma 4.6).

By the definition of I_M, we have $I(Y_i\,;(X_A)_i|(X_B)_i) \leq I_M$, and so we have

$$\begin{aligned} H(Z^{(n)}|X_B^{(n)},Y^{(n)}) &= H(Z^{(n)}|X_B^{(n)}) - I(Z^{(n)}\,;Y^{(n)}|X_B^{(n)}) \\ &\geq H(Z^{(n)}|X_B^{(n)}) - I(Y^{(n)}\,;X_A^{(n)}|X_B^{(n)}) \\ &\geq nH - nI_M = n(H-I_M). \end{aligned}$$

We conclude by analogy to the proof of Fano's inequality ([19], Theorem 2.11.1). Let E denote the event that Bob makes an error in guessing the value of $Z^{(n)}$ (so E is 1 if Bob makes an error and 0 otherwise). Then applying the chain rule for entropy in two different ways we have

$$\begin{aligned} H(E,Z^{(n)}|Y^{(n)},X_B^{(n)}) &= H(Z^{(n)}|Y^{(n)},X_B^{(n)}) + H(E|Z^{(n)},Y^{(n)},X_B^{(n)}) \\ &= H(E|Y^{(n)},X_B^{(n)}) + H(Z^{(n)}|E,Y^{(n)},X_B^{(n)}). \end{aligned}$$

Now E is a binary variable and so $H(E|Z^{(n)}) \leq 1$ (in bits). Also $H(E|Z^{(n)},Y^{(n)},X_B^{(n)}) = 0$ (if we know $Y^{(n)}$ and $X_B^{(n)}$ then we know Bob's guess, and if we also know $Z^{(n)}$ then we know whether it is correct). Hence, rearranging the previous equation we have

$$\begin{aligned} H(Z^{(n)}|X_B^{(n)},Y^{(n)}) &= H(E|Y^{(n)},X_B^{(n)}) + H(Z^{(n)}|E,Y^{(n)}) \\ &\leq 1 + H(Z^{(n)}|E,Y^{(n)}) \\ &= 1 + P(E=0)H(Z^{(n)}|Y^{(n)},E=0) + P(E=1)H(Z^{(n)}|Y^{(n)},E=1) \\ &\leq 1 + P(E=1)H(Z^{(n)}) \\ &= 1 + P(E=1)nH. \end{aligned}$$

Hence $P(E=1) \geq H - I_M - \frac{1}{n} \to H - I_M > 0$ as $n \to \infty$. Hence $P(E=1)$ is bounded away from 0 for sufficiently large n, and hence also for all n since if we

have $P(E = 1) = 0$ for some n then we can build a coding scheme for nk for all k by concatenating codes.

□

4.4 Deterministic specifications

The problem with the expression in Definition 4.8 is that it is unclear how we might ever attempt to evaluate it in general: in particular, we are required to quantify over the possible probability distributions on $\mathcal{X}_A$, as well as over the elements of $\mathcal{X}_B$. However, in the case where the system itself is deterministic—that is, where the output Y is uniquely determined by the values of X_A and X_B—we shall see that matters simplify considerably.

The simplification which occurs is that we may assume without loss of generality that Alice chooses a strategy which results in a uniform distribution on some set of possible outcomes, and computing the information flow is reduced to computing the maximum possible size of this set.

Theorem 4.12. *Let $\mathcal{X}_A, \mathcal{X}_B, \mathcal{Y}$ be any finite sets, and let $M = p_{Y|X_A,X_B}(y|x_A, x_B)$ be any stochastic* 0-1 *matrix (that is, for each $x_A \in \mathcal{X}_A, x_B \in \mathcal{X}_B$ we have $p_{Y|X_A,X_B}(y|x_A, X_B) = 1$ for precisely one value of $y \in \mathcal{Y}$, and $p_{Y|X_A,X_B}(y|x_A, x_B) = 0$ otherwise). Then*

$$I_M = \max_{x_B \in \mathcal{X}_B} \log |\{f(x_A, x_B)|x_A \in \mathcal{X}_A\}| ,$$

where the function $f : \mathcal{X}_A \times \mathcal{X}_B \to \mathcal{Y}$ is defined by $p_{Y|X_A,X_B}(f(x_A, x_B)|x_A, x_B) = 1$ for all $x_A \in \mathcal{X}_A, x_B \in \mathcal{X}_B$.

In order to prove this theorem, we will require as a lemma *Jensen's inequality.*

Definition 4.13. Let $a, b \in \mathbb{R} \cup \{\pm\infty\}$. A function $f : (a, b) \to \mathbb{R}$ is *convex* if for any $x_1, x_2 \in (a, b)$ and any $t \in [0, 1]$ we have

$$f(tx_1 + (1 - t)x_2) \leq tf(x_1) + (1 - t)f(x_2).$$

If the above holds with the inequality reversed (that is, we have $f(tx_1 + (1 - t)x_2) \geq tf(x_1) + (1 - t)f(x_2)$ for all $t \in [0, 1]$) then f is *concave.*

If the inequalities above are strict then we say f is *strictly convex* (respectively strictly concave).

Lemma 4.14 (Jensen's inequality)**.** *Let f be a convex function, and X a random variable. Then we have*

$$\mathbb{E}f(X) \geq f(\mathbb{E}X).$$

If f is strictly convex then we have equality if and only if X is constant (with probability 1).

Proof. [19], Theorem 2.6.2. □

Trivially, if f is concave then $-f$ is convex and so by Jensen's inequality we have $-\mathbb{E}f(X) \geq -f(\mathbb{E}(X))$ and so $\mathbb{E}f(X) \leq f(\mathbb{E}X)$.

Proof of Theorem 4.12. By Definition 4.8 and equation (4.1), we have

$$\begin{aligned} I_M &= \max_{x_B \in \mathcal{X}_B} \max_{p_{X_A}} I(I\,; X_A | X_B = x_B) \\ &= \max_{x_B \in \mathcal{X}_B} \max_{p_{X_A}} \mathbb{E}_{X_A,Y} \log \frac{p_{Y|X_A,X_B}(Y|X_A, x_B)}{p_{Y|X_B}(Y|x_B)} \\ &= \max_{x_B \in \mathcal{X}_B} \max_{p_{X_A}} \mathbb{E}_{X_A} \log \frac{1}{p_{Y|X_B}(f(X_A, x_B)|x_B)}, \end{aligned}$$

where f is defined as in the statement of the theorem. Hence by Jensen's inequality, since log has its second derivative everywhere negative and hence ([19], Theorem 2.7.1) is concave, we have

$$I_M \leq \max_{x_B \in \mathcal{X}_B} \max_{p_{X_A}} \log \mathbb{E}_{X_A} \frac{1}{p_{Y|X_B}(f(X_A, x_B)|x_B).} \tag{4.2}$$

Now

$$\begin{aligned} \mathbb{E}_{X_A} \frac{1}{p_{Y|X_B}(f(X_A, x_B)|x_B)} &= \sum_{x_A \in \mathcal{X}_A} \frac{p_{X_A}(x_A)}{p_{Y|X_A}(f(x_A, x_B)|x_B)} \\ &= \sum_{y \in f(\mathcal{X}_A, x_B)} \sum_{\substack{x_A \in \mathcal{X}_A, \\ f(x_A, x_B) = y}} \frac{p_{X_A}(x_A)}{p_{Y|X_A}(y|x_B)} \\ &= \sum_{y \in f(\mathcal{X}_A, x_B)} \frac{\sum_{\substack{x_A \in \mathcal{X}_A, \\ f(x_A, x_B) = y}} p_{X_A}(x_A)}{p_{Y|X_A}(y|x_B)} \\ &= |f(\mathcal{X}_A, x_B)|\,, \end{aligned}$$

since $p_{Y|X_A}(y|x_B) = \sum_{\substack{x_A \in \mathcal{X}_A \\ f(x_A, x_B) = y}} p_{X_A}(x_A)$. Substituting this into equation (4.2) gives

$$I_M \leq \max_{x_B \in \mathcal{X}_B} \log |f(\mathcal{X}_A, x_B)|.$$

On the other hand equality can be achieved. Indeed, let $x_B \in \mathcal{X}_B$ be fixed and let $Z \subseteq \mathcal{X}_A$ contain a single element of the set $\{x_A \in \mathcal{X}_A | f(x_A, x_B) = y\}$ for each $y \in f(\mathcal{X}_A, x_B)$. Now let

$$p_{X_A}(x_A) = \begin{cases} \frac{1}{|Z|} & \text{if } x_A \in Z \\ 0 & \text{otherwise.} \end{cases}$$

Then for each $x_A \in \mathcal{X}_A$, we have $p_{Y|X_B}(f(x_A, x_B)|x_B) = 1/|Z|$, and so

$$E_{X_A} \log \frac{1}{p_{Y|X_B}(f(X_A, x_B)|x_B)} = \log |Z| = \log |f(\mathcal{X}_A, x_B)|.$$

Choosing x_B so as to maximise $|f(\mathcal{X}_A, x_B)|$ gives the result. □

This theorem essentially shows that for a deterministic system it is sufffficient to take a *possibilistic* view of Alice's actions, rather than a probabilistic one.

4.5 LogSumExp information flow

As discussed in Chapter 2, there are problems with the use of mutual information as the measure of information flow, and since none of the other proposed definitions is entirely satisfactory, we here give some consideration to what the appropriate definition is. We will begin by considering a simple setup where the input from High is a random variable X, and the output to Low is a random variable Y, and then extend our definition to the interactive framework developed above.

The fundamental problem with mutual information as a measure of information flow is due to the fact that the underlying theory was developed in the context of channel capacity: for instance, the main theorem justifying its use (Theorem 4.11) is about building a noisy channel out of repeated instances of the experiment. In reality, however, the attacker will often be allowed only a single instance of the game for each attack.

Another perspective is that we are not interested in the *average* saving of work by the attacker. Rather, the relevant question is: supposing that the attacker has a fixed budget of computational work, what is the probability that she can recover the secret (say, an encryption key)?

We assume that the secret is encoded into the coins used to make the random choice of the High user's input. The attacker's strategy will be to perform the interaction with the system, obtaining some information about the secret, and then use her computational work budget to make guesses consistent with this information.

Note that it seems reasonable to wonder whether a more 'adaptive' strategy might be better for the attacker: she could make some guesses during the interaction and use the outcomes to guide her actions. However, since we assume that her guesses are made to an oracle which simply answers 'right' or 'wrong', and that if she ever receives the answer 'right' then she has won, we can assume that she defers any oracle queries until the end, and in a putative adaptive strategy acts on the basis that the answers to all her queries were 'wrong'.

Suppose that on receiving the output $y \in Y$, the information in bits received by the attacker about the key is $I_{X,Y}(y)$. If the key length is n, and the attacker's work budget is $W \ll 2^n$, then the probability of success is approximately

$$\frac{W}{2^n} 2^{I_{X,Y}(y)}.$$

.

The total probability of success is therefore

$$\sum_{y \in \mathcal{Y}} P(Y = y) \frac{W}{2^n} 2^{I_{X,Y}(y)} = \frac{W}{2^n} \mathbb{E}_y 2^{I_{X,Y}(y)}. \tag{4.3}$$

Since the probability of success without the interaction is $\frac{W}{2^n}$, the probability has increased by a factor of $\mathbb{E}_y 2^{I_{X,Y}(y)}$. To obtain information flow in bits we take the logarithm of this quantity.

Definition 4.15. Let X and Y be random variables, and let $I_{X,Y} : \mathcal{Y} \to \mathbb{R}$ be any function, where $I_{X,Y}(y)$ is interpreted as the information in bits obtained about X from seeing output y. The *LogSumExp information flow from X to Y with respect to I*, denoted $I_{\mathrm{LSE}}(X \to Y)$, is defined in bits by

$$I_{\mathrm{LSE}}(X \to Y) = \log_2 \mathbb{E}_{y \sim Y} 2^{I_{X,Y}(y)}.$$

Note that there is another related point of view which justifies this definition. Suppose that the attacker is attacking many systems (with different values of the secret), where she can interact with each system once and the cost of doing so is a constant multiple of the cost of guessing a value for a secret. What is the minimum work per success that she can achieve, assuming that she selects a strategy to optimise this?

Up to constant factors, the attacker's strategy will be optimised by spending equal resources in interacting with the system and on guessing key values (indeed, no strategy can beat this by more than a factor of 2). We may therefore assume that the

attacker expends some fixed budget W on guessing key values after each interaction, where W is equal to the cost of a single interaction. The work per success is therefore $\Theta(W/P) = \Theta(2^{n-I_{\mathrm{LSE}}(X \to Y)})$, where P is the probability of success in (4.3). Since the work per success by simply guessing keys would be $\Theta(2^n)$, we have that $I_{\mathrm{LSE}}(X \to Y)$ is the saving (in bits) in work per success caused by the availability of the interaction.

In both the 'probability' and 'work per success' viewpoints, one interpretation of $I_{\mathrm{LSE}}(X \to Y)$ is that having the interaction available is equivalent to being provided with $I_{\mathrm{LSE}}(X \to Y)$ bits of the key for free.

What definition of $I_{X,Y}$ should we use? Given the well-known characterisation of the mutual information $I(X\,;Y)$ as

$$I(X\,;Y) = \mathbb{E}_{y \sim Y} H(X) - H(X|Y=y),$$

it is tempting to think that we should take $I_{X,Y}(y) = H(X) - H(X|Y=y)$.

However, this cannot be right, for the very simple reason that $H(X) - H(X|Y=y)$ can be negative, whereas it makes no sense to say that Bob receives a negative amount of information about the secret.[1] A more sensible choice is obtained by grouping the expression for $I(X\,;Y)$ in a different way:

$$\begin{aligned} I(X\,;Y) &= -\sum_{x,y} p_{X,Y}(x,y) \log \frac{p_{X,Y}(x,y)}{p_X(x) p_Y(y)} \\ &= -\sum_{y} p_Y(y) \sum_{x} p_{X|Y}(x|y) \log \frac{p_{X|Y}(x|y)}{p_X(x)} \end{aligned}$$

It seems reasonable therefore to adopt the information measure

$$\begin{aligned} I_{X,Y}(y) &= -\sum_{x} p_{X|Y}(x|y) \log \frac{p_{X|Y}(x|y)}{p_X(x)} \\ &= \left(-\sum_{x} p_{X|Y}(x|y) \log p_X(x) \right) - H(X|Y=y) \\ &= H_y(X) - H(X|Y=y), \end{aligned}$$

where we define $H_y(X) = -\sum_x p_{X|Y}(x|y) \log p_X(x)$ to be the *y-weighted entropy* of X (note that this is not a standard concept). Observe that $I_{X,Y}(y)$ can also be characterised as the log of the Bayes factor on values of x, weighted by the posterior probability $p_{X|Y}(x|y)$.

[1] One might think that it could represent Bob somehow being 'misled', but since Bob is assumed to know the distribution of X, he will also know for which y the information is misleading, and could choose to ignore it.

We have

$$\begin{aligned}\mathbb{E}_{y\sim Y} H_y(X) &= -\sum_{x,y} p_Y(y) p_{X|Y}(x|y) \log p_X(x) \\ &= -\sum_x p_X(x) \log p_X(x) \\ &= H(X),\end{aligned}$$

and so

$$\mathbb{E}_{y\sim Y} I_{X,Y}(y) = I(X\,;Y).$$

This definition is mathematically satisfying, and seems to be a fairly plausible characterisation of observation-by-observation information flow. Unfortunately, we have been unable to come up with a threat model which is directly represented by it. However, after a small modification we can do so.

Suppose that the secret is a uniformly distributed key $k \in \mathcal{K}$, which is used to generate the distribution of X; that is, $x = f(k)$, where $f : \mathcal{K} \to \mathcal{X}$ is some function such that

$$p_X(x) = \frac{|f^{-1}(x)|}{|\mathcal{K}|}.$$

Of course with no observations a guess for the value of k will be correct with probability $1/|\mathcal{K}|$. On the other hand, for all $k \in \mathcal{K}$ and $y \in \mathcal{Y}$ we have that

$$\begin{aligned}P(K = k|Y = y) &= \frac{P(Y = y|K = k)P(K = k)}{P(Y = y)} \\ &= \frac{1}{|\mathcal{K}|}\frac{P(Y = y|X = f(k))}{P(Y = y)} \\ &= \frac{1}{|\mathcal{K}|}\frac{P(X = f(k)|Y = y)}{P(X = f(k))},\end{aligned}$$

where both the equalities are by Bayes' theorem.

After the attacker observes that $Y = y$, therefore, her best guess is to choose x so as to maximise $\frac{p_{X|Y}(x|y)}{p_X(x)}$ and then guess an arbitrary element of $f^{-1}(x)$. By doing this, the attacker gains an advantage of $\frac{p_{X|Y}(x|y)}{p_X(x)} = \frac{p_{Y|X}(y|y)}{p_Y(y)}$, so we define

$$I'_{X,Y}(y) = \max_{x\in\mathcal{X}} \log \frac{p_{X|Y}(x|y)}{p_X(x)} = \max_{x\in\mathcal{X}} \log \frac{p_{Y|X}(y|x)}{p_Y(y)}.$$

Concretely, the above discussion means that in trying to guess the value of k, the attacker obtains the same advantage from being told that the value of Y is y as being given $I'_{X,Y}(y)$ bits of the key for free. Similarly, she is *a priori* indifferent between being allowed to observe the value of Y and receiving I'_{LSE} bits of key. It is in this

sense that we are able to say that a single use of the channel reveals I'_{LSE} bits of information.

Since both of the above definitions have attractive features, we will discuss both below. We will denote their corresponding LSE information flows by I_{LSE} and I'_{LSE} respectively.

Definition 4.16. Let X and Y be random variables. Define (in bits)

$$\begin{aligned} I_{\mathrm{LSE}}(X \to Y) &= \log_2 \mathbb{E}_{y\sim Y} 2^{\sum_x p_{X|Y}(x|y) \log \frac{p_{X|Y}(x|y)}{p_X(x)}} \\ &= \log_2 \mathbb{E}_{y\sim Y} 2^{H_y(X) - H(X|Y=y)}, \end{aligned}$$

where $H_y(X)$ defined by

$$H_y(X) = -\sum_{x\in\mathcal{X}} p_{X|Y}(x|y) \log p_X(x)$$

is the *y-weighted entropy* of X. Define also

$$\begin{aligned} I'_{\mathrm{LSE}}(X \to Y) &= \log_2 \mathbb{E}_{y\sim Y} 2^{\max_x \log \frac{p_{Y|X}(y|x)}{p_Y(y)}} \\ &= \log_2 \mathbb{E}_{y\sim Y} \max_x \frac{p_{Y|X}(y|x)}{p_Y(y)} \\ &= \log_2 \sum_{y\in\mathcal{Y}^+} \max_x p_{Y|X}(y|x), \end{aligned}$$

where $\mathcal{Y}^+$ is the support of p_Y (i.e. those $y \in \mathcal{Y}$ such that $p_Y(y) > 0$).

(Definitions in nats are similar with natural logarithms and exponentials.)

We illustrate these definitions with an example. Suppose that $\mathcal{X} = [0, 2^n - 1], \mathcal{Y} = [0, 2^n - 1] \cup \{\perp\}$ and we have $p_X(k) = 2^{-n}$ for all k, and $p_{Y|X}(\perp|k) = 1 - 2^{-n/2}, p_{Y|X}(k|k) = 2^{-n/2}$ and $p_{Y|X}(k'|k) = 0$ for all $k \neq k'$. This represents a system that relays its input with probability $2^{-n/2}$ and otherwise outputs $\perp$, and for which the input is uniformly distributed. Then

$$\begin{aligned} I_{\mathrm{LSE}}(X \to Y) = I'_{\mathrm{LSE}}(X \to Y) &= \log_2 \left(2^n 2^{-n/2} + 1 - 2^{-n/2}\right) \\ &\approx n/2. \end{aligned}$$

Both definitions correctly find that an observer of Y receives the equivalent of $n/2$ bits of information about the secret, since the attacker will have to do about $n/2$ runs of the experiment in order to hit a run with $Y \neq \perp$, whereupon she obtains n bits of information about the secret.

By contrast, noting that

$$H(Y) = -\left(1 - 2^{-n/2}\right) \log_2 \left(1 - 2^{-n/2}\right) - 2^n 2^{-3n/2} \log_2 2^{-3n/2}$$

and

$$H(Y|X = x) = -\left(1 - 2^{-n/2}\right) \log_2 \left(1 - 2^{-n/2}\right) - 2^{-n/2} \log_2 2^{-n/2}$$

and so

$$\begin{aligned} H(Y) - H(Y|X = x) &= 2^{-n/2} \log_2 2^{-n/2} - 2^{-n/2} \log_2 2^{-3n/2} \\ &= n2^{-n/2} \end{aligned}$$

for all x, we have

$$I(Y\,;X) = n2^{-n/2} \ll n/2.$$

We thus find that mutual information gives a spuriously low value for the information flow. This is unsurprising, since this system is similar to the example used by Smith in [63] to show the problems with mutual information as a measure of information flow.

Smith's proposed definitions are based on *min-entropy* H_∞:

$$H_\infty(X) = -\log_2 \max_x p_X(x).$$

Smith then defines a version of conditional entropy, and defines the information flow from X to Y as $H_\infty(X) - H_\infty(X|Y)$. He then shows that systems similar to the above (which leak a large amount of the secret but with fairly small probability) are correctly found to be vulnerable by his definitions. His paper implies that the reason for this is the use of min-entropy rather than Shannon entropy.

We do not believe this to be the case, however. In fact the reason is in his definition of conditional entropy $H_\infty(X|Y)$:

$$H_\infty(X|Y) = -\log_2 \mathbb{E}_y \max_x p_{X|Y}(x|y).$$

The critical point is that the log is taken *outside* the expectation, rather inside as in the normal definition of conditional Shannon entropy:

$$H(X|Y) = -\mathbb{E}_y \log_2 \sum_x p_{X|Y}(x|y) \log_2 p_{X|Y}(x|y).$$

This means that he obtains effectively a LogSumExp measure, just as discussed above. Indeed, if we were to take a definition of conditional Shannon entropy as

$$H'(X|Y) = -\log_2 \mathbb{E}_y \sum_x p_{X|Y}(x|y) \log p_{X|Y}(x|y)$$

and define $I'(X\,;Y) = H(X) - H'(X|Y)$ we would obtain similar results (although as discussed above this is not the right thing to do, because $H(X) - H(X|Y = y)$ is not the right measure of the information about X obtained on observing $Y = y$).

Althought Smith's definition is appropriate for the threat model he considers, where the adversary succeeds by guessing the value of X, it is not appropriate for our work, where the adversary's objective is to guess the value of a secret from which the value of X is derived. This is witnessed by this example, if we consider the information flow from Y to X.

For this system, we have

$$H_\infty(Y) = -\log_2 p_Y(\bot) = -\log_2\left(1 - 2^{-n/2}\right).$$

For any $x \in \mathcal{X}$, we have $\max_y P_{Y|X}(y|x) = P_{Y|X}(\bot|x) = 1 - 2^{-n/2}$. Hence we have

$$H_\infty(Y|X) = -\log_2\left(1 - 2^{-n/2}\right) = H_\infty(Y),$$

and so the information flow from Y to X is

$$H_\infty(Y) - H_\infty(Y|X) = 0.$$

This is clearly not correct: the value of Y does contain some information about the value of X, even though the modal value ($\bot$) is the same in the prior and posterior distributions.

What about I_{LSE} and I'_{LSE}? As suggested by the notation, unlike mutual information neither of these quantities is symmetric in X and Y. Indeed, we have

$$I'_{\mathrm{LSE}}(Y \to X) = \log_2 2^n = n,$$

and

$$\begin{aligned} I_{\mathrm{LSE}}(Y \to X) &= \log_2 2^{2^{-n/2}\log\frac{2^{-n/2}}{2^{-3n/2}} + \left(1-2^{-n/2}\right)0} \\ &= n2^{-n/2}. \end{aligned}$$

Which of these answers is most plausible? I'_{LSE} effectively ignores the fact that there is a large amount (indeed, a significant majority) of the probability space of Y about which we learn nothing. This is justified by the attack model discussed above, but seems in some sense implausible. On the other hand, I_{LSE} considers this to reduce the information flow to a negligible amount. Depending on the threat model, this also seems unsatisfactory.

Perhaps a compromise between $I(y)$, which is the expected value of $\log_2 \frac{p_{X|Y}(x|y)}{p_X(x)}$ (with respect to the posterior distribution) and $I'(y)$ which is the maximum value, is appropriate. We could instead use the LogSumExp. This corresponds to the threat model of the attacker guessing an element of $f^{-1}(x)$ with probability $p_{X|Y}(x|y)$.

Definition 4.17. Let X, Y be probability distributions. Define

$$\begin{aligned} I''(y) &= \log_2 \sum_x p_{X|Y}(x|y) \frac{p_{X|Y}(x|y)}{p_X(x)} \\ &= \log_2 \sum_x p_{X|Y}(x|y) \frac{p_{Y|X}(y|x)}{p_Y(y)}. \end{aligned}$$

Consequently define

$$\begin{aligned} I''_{\mathrm{LSE}}(X \to Y) &= \log_2 \mathbb{E}_{y \sim Y} 2^{I''(y)} \\ &= \log_2 \sum_y p_Y(y) \sum_x p_{X|Y}(x|y) \frac{p_{Y|X}(y|x)}{p_Y(y)} \\ &= \log_2 \sum_{x,y} p_{X|Y}(x|y) p_{Y|X}(y|x). \end{aligned}$$

It is immediate from this definition that $I''_{\mathrm{LSE}}(X \to Y) = I''_{\mathrm{LSE}}(Y \to X)$ for all X and Y.

Applying this definition to the problem considered above, we have

$$I''_{\mathrm{LSE}}(Y \to X) = I''_{\mathrm{LSE}}(X \to Y) = I'_{\mathrm{LSE}}(X \to Y) = I_{\mathrm{LSE}}(X \to Y) = n/2.$$

4.6 LSE information flow in interactive systems

In this section, we begin by establishing some mathematical properties of our new definitions of information flow, and then study their application to the case of interactive systems, and in particular deterministic interactive systems.

Proposition 4.18. *Let X, Y be random variables. Then*

$$I'_{LSE}(X \to Y) \geq I''_{LSE}(X \to Y) \geq I_{LSE}(X \to Y) \geq I(X\,;Y),$$

with equality in the first and second inequalities if and only if X and Y are independent, and in the third if and only if $H_y(X) - H(X|Y = y)$ is constant for all y.

Proof. The first inequality is immediate from the definitions. For the second, we will show that $I''(y) \geq I(y)$ for all y. Indeed, by Jensen's inequality (Lemma 4.14), since $\log_2$ is concave we have

$$\begin{aligned} I''(y) &= \log_2 \mathbb{E}_{x \sim X|Y=y} \frac{p_{X|Y}(x|y)}{p_X(x)} \\ &\geq \mathbb{E}_{x \sim X|Y=y} \log_2 \frac{p_{X|Y}(x|y)}{p_X(x)} \\ &= I(y). \end{aligned}$$

For the third inequality, by Jensen's inequality we have

$$\begin{aligned} I_{\mathrm{LSE}}(X \to Y) &= \log_2 \mathbb{E}_{y \sim Y} 2^{H_y(X) - H(X|Y=y)} \\ &\geq \mathbb{E}_{y \sim Y} \log_2 2^{H_y(X) - H(X|Y=y)} \\ &= \mathbb{E}_{y \sim Y} H_y(X) - H(X|Y=y) \\ &= H(X) - H(X|Y) \\ &= I(X\,;Y). \end{aligned}$$

□

The following is immediate from Proposition 4.18 and the corresponding results for mutual information ([19], corollary to Theorem 2.6.3).

Corollary 4.19. *Let X, Y be random variables. Then*

$$I_{LSE}(X \to Y), I'_{LSE}(X \to Y), I''_{LSE}(X \to Y) \geq 0,$$

with equality if and only if X and Y are independent.

Just as mutual information has a conditional version, defined by $I(X\,;Y|Z) = \mathbb{E}_{z \in Z} I(X\,;Y|Z=z)$, we would like to have a conditional version of LSE information flow. However, for the same reasons as in the original definition of I_{LSE}, we will take not the expected value but the LogSumExp.

Definition 4.20. Let X, Y, Z be random variables. The *conditional LSE information flow* $I_{\mathrm{LSE}}(X \to Y|Z)$ is defined (in bits) by

$$\begin{aligned} I_{\mathrm{LSE}}(X \to Y|Z) &= \log_2 \mathbb{E}_{z \sim Z} 2^{I_{\mathrm{LSE}}(X \to Y|Z=z)} \\ &= \log_2 \sum_{z \in \mathcal{Z}} \sum_{y \in \mathcal{Y}} p_{Y,Z}(y,z) 2^{H(X|Z=z) - H(X|Y=y,Z=z)}, \end{aligned}$$

and in nats the same way but with logarithms, exponentials and entropy to base e. The conditional versions of I'_{LSE} and I''_{LSE} are defined similarly.

We now show an equivalent of Lemma 4.10, showing that if $X \to Y \to Z$ is a Markov chain then the I'_{LSE} information flow from Z to X is at most that from Y to X.

Theorem 4.21 (Data-processing inequalities for LSE information flow)**.** *Let X, Y, Z be random variables such that $X \to Y \to Z$. Then*

$$I'_{LSE}(Z \to X) \leq I'_{LSE}(Y \to X).$$

Proof. Recall that

$$I'_{\mathrm{LSE}}(Y \to X) = \log_2 \sum_x \max_y p_{X|Y}(x|y).$$

Now we have

$$\begin{aligned} I'_{\mathrm{LSE}}(Z \to X) &= \log_2 \sum_x \max_z p_{X|Z}(x|z) \\ &= \log_2 \sum_x \max_z \sum_y p_{Y|Z}(y|z) p_{X|Y,Z}(x|y,z) \\ &= \log_2 \sum_x \max_z \sum_y p_{Y|Z}(y|z) p_{X|Y}(x|y) \\ &\leq \log_2 \sum_x \max_z \max_y p_{X|Y}(x|y) \\ &= I'_{\mathrm{LSE}}(Y \to X) \end{aligned}$$

as required, where the third equality is by Markovity of X, Y, Z and the inequality is because $\sum_y p_{Y|Z}(y|z) = 1$. □

We now aim to reproduce the theory of the earlier part of this chapter for LSE measures of information flow, with the goal of showing that for deterministic systems they yield the same answers as mutual information.

As before, we have random variables $X = (X_A, X_B)$, where X_A and X_B are required to be independent, and Y. The distribution $Y|X$ is fixed, and our task is to choose probability distribution functions for X_A and X_B so as to maximise

$$\begin{aligned} I(X_A \to Y|X_B) &= \log_2 \sum_{x_B \in \mathcal{X}_B} p_{X_B}(x_B) 2^{I(X_A \to Y|X_B = x_B)} \\ &= \log_2 \sum_{x_B \in \mathcal{X}_B, y \in \mathcal{Y}} p_{X_B}(x_B) p_{Y|X_B}(y|x_B) 2^{I(y|X_B = x_B)}, \end{aligned}$$

where $p_{Y|X_B}(y|x_B)$ is the marginal probability distribution, given by $p_{Y|X_B}(y|x_B) = \sum_{x_A \in \mathcal{X}_A} p_{X_A}(x_A) p_{Y|X_A,X_B}(y|x_A, x_B)$.

We first observe a version of Proposition 4.7, showing that we may assume that Bob chooses a deterministic strategy.

Proposition 4.22. *Let P_{X_A}, p_{X_B} be probability distributions over finite sets, and let $p_{Y|X}$ be fixed. Let I be any measure of information flow from X_A to Y. Then there exists some probability distribution p'_{X_B} with $p'_{X_B}(x_B) \in \{0,1\}$ for all $x_B \in \mathcal{X}_B$ such that*

$$\sum_{\substack{x_B \in \mathcal{X}_B, \\ y \in \mathcal{Y}}} p_{X_B}(X_B) p_{Y|X_B}(y|x_B) 2^{I(y|X_B=x_B)} \leq \sum_{\substack{x_B \in \mathcal{X}_B, \\ y \in \mathcal{Y}}} p'_{X_B}(X_B) p_{Y|X_B}(y|x_B) 2^{I(y|X_B=x_B)}.$$

Proof. Choose x_B so as to maximise

$$\sum_{y \in \mathcal{Y}} p_{Y|X_B}(y|x_B) 2^{I(y|X_B=x_b)}.$$

Set $p'_{X_B}(x_B) = 1$ and $p'_{X_B}(x) = 0$ for $x \neq x_B$, and the result is clear. □

We can therefore define LSE information flow for interactive systems analagously to Definition 4.8.

Definition 4.23. Let $\mathcal{X}_A, \mathcal{X}_B, \mathcal{Y}$ be any finite sets, and let $M = p_{Y|X_A,X_B}(y|x_A, x_B)$ be any stochastic matrix. Then the *LSE information flow permitted by M* is defined by

$$\begin{aligned}
I_{\mathrm{LSE}}(M) &= \max_{x_B \in \mathcal{X}_B} \max_{p_{X_A}} I_{\mathrm{LSE}}(X_A \to Y | X_B = x_B) \\
I'_{\mathrm{LSE}}(M) &= \max_{x_B \in \mathcal{X}_B} \max_{p_{X_A}} I'_{\mathrm{LSE}}(X_A \to Y | X_B = x_B) \\
I''_{\mathrm{LSE}}(M) &= \max_{x_B \in \mathcal{X}_B} \max_{p_{X_A}} I''_{\mathrm{LSE}}(X_A \to Y | X_B = x_B)
\end{aligned}$$

The main theorem of this section is that for a deterministic system, the LSE information flow is equal to the 'mutual information' information flow.

Theorem 4.24. *Let $\mathcal{X}_A, \mathcal{X}_B, \mathcal{Y}$ be any finite sets, and let $M = p_{Y|X_A,X_B}(y|x_A, x_B)$ be any stochastic 0-1 matrix. Then*

$$I_{LSE}(M) = I'_{LSE}(M) = I''_{LSE}(M) = I_M.$$

Proof. By Proposition 4.18, it suffices to prove that $I'_{\mathrm{LSE}}(M) \leq I_M$.

For fixed x_B, p_{X_A} we have

$$\begin{aligned}
I'_{\mathrm{LSE}}(X_A \to Y | X_B = x_B) &= \sum_{y \in \mathcal{Y}} \max_{x_A} p_{Y|X_A,X_B}(y|x_A, x_B) \\
&= |\{f(x_A, x_B) | x_A \in \mathcal{X}_A : p_{X_A}(x_A) \neq 0\}| \\
&\leq I_M,
\end{aligned}$$

where f is as in Theorem 4.12, defined by $p_{Y|X_A,X_B}(f(x_A, x_B)|x_A, x_B) = 1$ for all $x_A \in \mathcal{X}_A, x_B \in \mathcal{X}_B$. □

4.7 Dalenius leakage

In our earlier discussion justifying the definition of I'_{LSE}, we characterised it as the maximum (multiplicative) advantage that Bob could obtain in guessing the value of a uniformly distributed cryptographic key, whose value determines the value of X according to some function f so as to realise the correct prior distribution p_X. More generally we can ask what advantage Bob can obtain in guessing the value of *any* random variable Z which is correlated with X (but otherwise independent of Y, such that $X \to Y \to Z$ form a Markov chain). This is sometimes referred to as the 'Dalenius leakage', after a desideratum attributed to T. Dalenius by Dwork in [23].

Definition 4.25. Let X, Y be any random variables. The *Dalenius leakage* from X to Y, $I_D(X \to Y)$, is defined by

$$I_D(X \to Y) = \sup_{Z \in \mathcal{D}} I_\infty(Z \to Y),$$

where $\mathcal{D}$ is the family of random variables Z such that $Z \to X \to Y$ form a Markov chain, and I_∞ is the min-entropy leakage

$$I_\infty(X \to Y) = \log \mathbb{E}_{y \sim Y} \frac{\max_{x \in \mathcal{X}} p_{X|Y}(x|y)}{\max_{x \in \mathcal{X}} p_X(x)}.$$

Clearly by the earlier discussion, we have that $I_D(X \to Y) \geq I'_{\mathrm{LSE}}(X \to Y)$. In fact, we are able to show that they are equal, using the theory of *g-leakage* developed by Alvim, Chatzikokolakis, Palamidessi and Smith in [3]. Briefly, we assume that Bob makes a 'guess' w drawn from some set $\mathcal{W}$, and receives a payoff of $g(w, x)$, where x is the true value of X and the function $g : \mathcal{W} \times \mathcal{X} \to [0, 1]$ is the *gain function*. The (multiplicative) gain to Bob from being allowed to observe y is the (multiplicative) *g-leakage*.

Definition 4.26. Let X, Y be any random variables, $\mathcal{W}$ any set and $g : \mathcal{W} \times \mathcal{X} \to [0, 1]$ a gain function. Then the g-leakage $I_g(X \to Y)$ is given by

$$I_g(X \to Y) = \log \mathbb{E}_{y \sim Y} \frac{\max_{w \in \mathcal{W}} \mathbb{E}_{x \sim X|Y=y} g(w, x)}{\max_{w \in \mathcal{W}} \mathbb{E}_{x \sim X} g(w, x)}.$$

Note that the min-entropy leakage is just I_g with $\mathcal{W} = \mathcal{X}$ and $g(w, x) = 1_{w=x}$.

The remarkable fact about g-leakage is that although for particular random variables X, Y we can have that $I_g(X \to Y)$ greatly exceeds the min-entropy leakage $I_\infty(X \to Y)$, the *capacity* (that is the maximum possible $I_g(X \to Y)$ for some fixed channel matrix $p_{Y|X}(y|x)$) is at most the min-entropy capacity. This is referred to the authors of [3] as the 'miracle' theorem.

Theorem 4.27 ('Miracle'). *Let $p_{Y|X}$ be any stochastic matrix, $\mathcal{W}$ any set and $g : \mathcal{W} \times \mathcal{X} \to [0,1]$ any gain function. Then we have*

$$\sup_{p_X} I_g(X \to Y) \leq \sup_{p_X} I_\infty(X \to Y).$$

Proof. [3], Theorem 5.1. □

Braun, Chatzikokolakis and Palamidessi observe in [11] that there is a simple formula for the min-entropy capacity.

Proposition 4.28. *Let $p_{Y|X}$ be any stochastic matrix. Then we have*

$$\sup_{p_X} I_\infty(X \to Y) = \sum_{y \in \mathcal{Y}} \max_{x \in \mathcal{X}} p_{Y|X}(y|x).$$

Proof. [11], Proposition 5.1. □

Looking back to the final line of Definition 4.16, we see that this is precisely equal to the value of $I'_{\mathrm{LSE}}(X \to Y)$, so long as p_Y is everywhere supported. Consequently, we have that the I'_{LSE} capacity is equal to the min-entropy capacity.

Theorem 4.29. *Let $p_{Y|X}$ be any stochastic matrix. Then we have*

$$\sup_{p_X} I'_{LSE}(X \to Y) = \sup_{p_X} I_\infty(X \to Y).$$

The question of bounding the Dalenius leakage is considered by Alvim, Chatzikokolakis, McIver, Morgan, Palamidessi and Smith in [2]. They show that for any Z such that $Z \to X \to Y$ form a Markov chain, the maximum g-leakage over all gain functions from Z to Y is at most that from X to Y.

Lemma 4.30. *Let $Z \to X \to Y$ form a Markov chain. Then*

$$\sup_{g} I_g(Z \to Y) \leq \sup_{g} I_g(Y \to X).$$

Proof. [2], Corollary 23. □

Of course this does not tell us anything about Delanius leakage with respect to any particular g, and indeed as our earlier examples showed it is possible that (for instance) $I_\infty(Z \to Y) \gg I_\infty(X \to Y)$. However, combining Lemma 4.30 with the Miracle theorem gives that the Delanius leakage is bounded by the min-entropy *capacity*. Of course, this bound may not be tight. However, this upper bound is precisely what we need to complete our proof that I'_{LSE} measures the Delanius leakage.

Theorem 4.31. *Let X, Y be any random variables. Then we have*

$$I_D(X \to Y) = I'_{LSE}(X \to Y).$$

Proof. The lower bound is by the discussion in Section 4.5. For the upper bound, we may assume without loss of generality that p_Y is supported on the whole of $\mathcal{Y}$. By Lemma 4.30 we have $I_D(X \to Y) \leq \sup_g I_g(Y \to X)$. By the Miracle theorem this is at most the min-entropy capactiy of $p_{Y|X}$, which by Theorem 4.29 is equal to the I'_{LSE} capacity of $p_{Y|X}$. But by the final line of Definition 4.16 we have that $I'_{\mathrm{LSE}}(X \to Y)$ does not depend on the prior p_X (so long as p_Y is everywhere supported), and so the I'_{LSE} capacity of $p_{X|Y}$ is equal to $I'_{\mathrm{LSE}}(X \to Y)$. □

4.8 Conclusions

In this chapter we have given a concrete model for information flow in interactive systems. We have shown that in the case of deterministic systems the formidable problem of computing this quantity is reduced to the far more tractable one of computing the maximum-sized set of possible outputs to Bob consistent with a single strategy for him. In Chapter 5 we will proceed to apply this to the situation of interactive systems modelled by deterministic finite transducers.

Finally, we have given consideration to the problems with mutual information as a measure of information flow, isolated what we believe to be the common underlying cause, and proposed a number of definitions avoid the problems inherent in both mutual information and previous proposals to replace it. However, it remains unclear precisely which definition is most appropriate, and this is certainly a topic requiring further research. On the other hand, when it comes to the question of *capacity* (rather than leakage for a particular prior) we believe that our results further reinforce the emerging consensus that min-entropy capacity is the correct measure.

Chapter 5

State machines

5.1 Introduction

In this chapter we show how to model interactive systems as state machines, and eventually reduce the problem of calculating the quantities defined in the previous chapter to a rather natural combinatorial problem about finite automata, which we shall then proceed to solve in Chapter 6.

We begin in Section 5.2 by setting out some basic definitions of finite autamata and finite-state transducers, and explaining how we use them to model interactive systems.

In Section 5.3 we place this within the context of the theory of Chapter 4, defining the sets of possible strategies for Alice and Bob, and the (deterministic) function relating strategy choices to outputs for Bob. We are then able to apply Theorem 4.12 to obtain (as Corollary 5.11) a characterisation of the information flow as the maximum size of certain sets in the language of the transducer (projected onto $\Sigma_B \times \Gamma_B$).

In Section 5.4 we show in Theorem 5.14 how to reduce this problem about transducers to one (Problem 5.16) purely about nondeterministic finite automata. This is by a process somewhat analagous to the well-known *lazy abstraction* technique used for noninterference modelling in CSP: here, we effectively treat Alice's actions as nondeterministic and hide them from Bob. We then represent Bob's view of the resulting system as a finite automaton.

In Section 5.5 we show in Theorem 5.20 that Problem 5.16 reduces to a very natural combinatorial problem about regular languages (Problem 5.21), namely that of calculating their *width*: that is, the size of their largest antichains with respect to the lexicographic order. The whole structure of the reduction in this chapter and

Chapter 4 from the problem of computing information flow in interactive systems to that of calculating the widths of regular languages is shown in Figure 5.6.

Finallly in Section 5.6 we consider the case of *asynchronous* systems (that is, where we do not assume that both Alice and Bob provide an input at each step). We consider some of the modelling issues which arise in this situation, and show that there is an equivalent reduction to antichain growth.

5.2 Automata and transducers

In this section, we introduce the main model that we will use to represent interactive systems. In common with Goguen-Meseguer (G-M), we adopt an automata-theoretic approach. However, unlike G-M we prefer to deal principally with *synchronous* systems: that is, where at each step both users choose an action and receive an output. There are a number of reasons for this. One is technical convenience: as we shall see later, the fitting an asynchronous system into the framework of strategies and outcomes described above requires the resolution of some subtle ambiguities.

A second reason is that without an explicit representation of time, an asynchronous model along the lines of G-M will not accurately reflect the information available to the attacker. As we saw in Theorem 2.4, such a model amounts to the attacker seeing stutter traces of the system, whereas in reality she will be able to distinguish between, say, the system changing from output 1 to output 2 after 1 unit of time and the same change occuring after 2 units of time. Note that this is not per se a problem with the G-M model: if we require total noninterference then changes in the output can only be precipitated by the actions of the attacker (or by actions which are allowed to interfere with her), so this issue does not arise.

Given the importance of asynchronous models in the existing literature, however, we will give some consideration to this setting in Section 5.6.

Perhaps the most fundamental concept of automata theory, and one which we will make extensive use of in the sequel, is that of a *non-deterministic finite automaton*.

Definition 5.1. A *non-deterministic finite automaton* (NFA) is a 5-tuple $\mathcal{A} = (Q, q_0, F, \Sigma, \Delta)$, where Q is a finite set of *states*, $q_0 \in Q$ is the *initial state*, $F \subseteq Q$ is the set of *accepting states*, and $\Delta : Q \times \Sigma \to \mathcal{P}(Q)$ is the *transition relation*.

A word $a_1 a_2 \dots a_k \in \Sigma^*$ is accepted by $\mathcal{A}$ if there exists a sequence of states $q_1 \dots q_k \in Q^*$ such that $q_k \in F$ and for every $0 \leq i < k$ we have $q_{i+1} \in \Delta(q_i, a_{i+1})$. We write $\mathcal{L}(\mathcal{A})$ for the subset of Σ^* accepted by $\mathcal{A}$.

This is not directly appropriate to model the behaviour of an interactive system: an automaton is most naturally viewed as taking a sequence of inputs, and either accepting or not, whereas for us an interactive system will have both inputs and outputs. A machine which deals with actions containing both inputs and outputs is called a *transducer*.

Definition 5.2. A *deterministic finite-state transducer* (DFST) is a 7-tuple $\mathcal{T} = (Q, q_0, F, \Sigma, \Gamma, \delta, \sigma)$, where Q is a finite set of *states*, $q_0 \in Q$ is the *initial state*, $F \subseteq Q$ is the set of *accepting states*, $\delta : Q \times \Sigma \to Q$ is the *transition function* and $\sigma : Q \times \Sigma \to \Gamma \cup \{\epsilon\}$ is the *output function*.

A pair $(a_1 a_2 \dots a_k, b_1 b_2 \dots b_l) \in \Sigma^* \times \Gamma^*$ is accepted by $\mathcal{T}$ if there exists a sequence of states $q_1 \dots q_k \in Q^*$ such that $q_k \in F$, for every $0 \leq i < k$ we have $q_{i+1} = \delta(q_i, a_{i+1})$ and $b_1 \dots b_l = \sigma(q_0, a_1)\sigma(q_1, a_2) \dots \sigma(q_{k-1}, a_k)$. We will write $\mathcal{L}(\mathcal{T})$ for the subset of $\Sigma^* \times \Gamma^*$ accepted by $\mathcal{T}$; such a set is a deterministic finite-state *transduction*, which we will also abbreviate by DFST.

Perhaps the simplest conceivable transduction is the identity, realised by the transducer shown in Figure 5.1 (for the alphabet $\Sigma = \Gamma = \{a, b\}$).

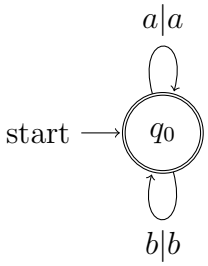

Figure 5.1: The identity transduction.

Another simple example is the transducer which counts (in unary) the number of as which occur before the first b, shown in Figure 5.2. Concretely, this relays all the as which appear before the first b, but not subsequent letters.

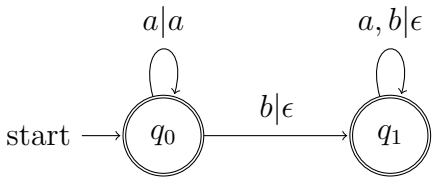

Figure 5.2: Another simple transducer.

This definition is not quite convenient for our purposes, because we assume that the agents are able to observe the passage of time. Hence even at a timestep where the machine does nothing, there should be a record in the trace of the fact that time has passed. We ensure this by requiring that there should be an output at each step, and apply the non-standard term 'synchronised' to describe this property (such a transducer is also sometimes called 'letter-to-letter').

Definition 5.3. A DFST $\mathcal{T} = (Q, q_0, F, \Sigma, \Gamma, \delta, \sigma)$ is *synchronised* if $\sigma(Q, \Sigma) \subseteq \Gamma$ (that is, we do not have $\sigma(q, a) = \epsilon$ for any $q \in Q$ and $a \in \Sigma$). In this case we say that $\mathcal{T}$ is a synchronised deterministic finite-state transducer, SDFST.

Note that this definition almost corresponds with the original definition of a *Mealy machine* ([44]), except that we allow for a set of final states $F \neq Q$. Observe that the transducer in Figure 5.1 is an SDFST, whereas the transducer in Figure 5.2 is not.[1]

It is clear that if $\mathcal{T}$ is synchronised then $(a_1 \dots a_k, b_1 \dots b_l) \in \Sigma^* \times \Gamma^*$ is accepted by $\mathcal{T}$ only if $l = k$. We shall therefore apply the 'zip' operation and view $\mathcal{T}$ as accepting elements of $(\Sigma \times \Gamma)^*$. Indeed, after applying this transformation, an SDFST is equivalent to a nondeterministic finite automaton over alphabet $\Sigma \times \Gamma$, with the property that for any $q \in Q$ and $a \in \Sigma$ there is exactly one $b \in \Gamma$ with $\Delta(q, (a, b)) \neq \emptyset$, and for this b we have that $\Delta(q, (a, b))$ is a singleton.

It is worth noting at this point that an SDFST cannot contain two words which first differ by an element of Γ. Indeed, this property characterises SDFSTs among regular sets in $(\Sigma \times \Gamma)^*$.

Proposition 5.4. *Let $L \subseteq (\Sigma \times \Gamma)^*$ be a regular language. Then L is an SDFST if and only if for every $w_1, w_2 \in L$ we have that w_1 and w_2 do* not *first differ by an element of Γ; that is, we do not have $w_1 = w(a, b)w'$ and $w_2 = w(a, b')w''$ with $w, w', w'' \in (\Sigma \times \Gamma)^*, a \in \Sigma$ and $b, b' \in \Gamma$ with $b \neq b'$.*

Proof. The 'only if' direction is trivial: let $\mathcal{T}$ be an SDFST accepting L. Then there is a unique state $q \in Q$ that can be reached after w, and so we must have $b = b' = \sigma(q, a)$.

Conversely, suppose that L is regular and has the property mentioned in the statement of the proposition. Then L is recognised by a deterministic finite automaton $\mathcal{A}$. Let $\mathcal{A}' = (Q, q_0, F, \Sigma \times \Gamma, \Delta)$ be a nondeterministic finite automaton with the same

[1] And it is clear that there is no SDFST realising this transduction. Moreover there is no SDFST which induces the same equivalence relation on Σ^*, since this transduction identifies inputs of different lengths, whereas an SDFST does not.

states and transitions as $\mathcal{A}$ (so in particular $\Delta(q,(a,b))$ is a singleton for every $q \in Q$ and $(a,b) \in \Sigma \times \Gamma$).

Let the set of accessible states in $\mathcal{A}$ be $A \subseteq Q$. Now for each $q \in A$ and each $a \in \Sigma$ we must have that $\bigcup \Delta(q,(a,b))$[2] is co-accessible for at most one b, since otherwise we could produce two accepted words first differing by an element of Γ. Let $X \subseteq Q \times \Sigma$ be the set of pairs (q,a) such that there exists $b \in \Gamma$ such that $\bigcup \Delta(q,(a,b))$ is co-accessible. Now let q_1 be fresh, let $b_0 \in \Gamma$ be arbitrary and let $\Delta' : Q \cup \{q_1\} \times (\Sigma \times \Gamma) \to Q \cup \{q_1\}$ be defined by

$$\Delta'(q,(a,b)) = \begin{cases} \Delta(q,(a,b)) & \text{if } \bigcup \Delta(q,(a,b)) \text{ is co-accessible} \\ \{q_1\} & \text{if } (q,a) \notin X \text{ and } b = b_0 \\ \emptyset & \text{otherwise} \end{cases}$$

for $q \in Q$, and $\Delta'(q_1,(a,b)) = q_1$ if $b = b_0$ and $\emptyset$ otherwise.

Now let

$$\mathcal{A}'' = (Q \cup \{q_1\}, q_0, F, \Sigma \times \Gamma, \Delta').$$

Note that q_1 is not co-accessible in $\mathcal{A}''$, and $\mathcal{A}''$ has the same transitions as $\mathcal{A}'$ ending in co-accessible states. Hence $\mathcal{L}(\mathcal{A}'') = \mathcal{L}(\mathcal{A}') = L$. On the other hand $\mathcal{A}''$ has the property that for any $q \in Q \cup \{q_1\}$ and any $a \in \Sigma$ there is exactly one $b \in \Gamma$ such that $\Delta(q,(a,b))$ is a singleton, and otherwise $\Delta(q,(a,b))$ is empty. Hence $\mathcal{L}(\mathcal{A}'')$ is an SDFST. □

We are interested in SDFSTs of a special kind, representing the fact that the system communicates separately with Alice and Bob. We will consider SDFSTs whose input and output alphabets Σ and Γ are of the form $\Sigma_A \times \Sigma_B$ and $\Gamma_A \times \Gamma_B$ respectively. The pairs (Σ_A, Γ_A) and (Σ_B, Γ_B) represent the input and output alphabets used for communication with Alice and Bob respectively.

A simple example of this is the system which simply relays messages between the two agents (with $\Sigma_A = \Sigma_B = \Gamma_A = \Gamma_B = \{a,b\}$). This is shown in Figure 5.3.

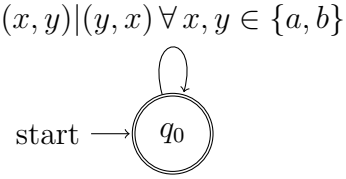

Figure 5.3: A relay system.

[2] i.e. the single element of the singleton $\Delta(q,(a,b))$

5.3 Strategies and information flow

In order to apply the framework of Chapter 4, we must define the spaces $\mathcal{X}_A, \mathcal{X}_B$ of strategies for Alice and Bob, the space Y of outcomes visible to Bob, and the matrix $p_{Y|X_A,X_B}(y|x_A, x_B)$ governing which outcomes occur. Since we are considering deterministic specifications, the matrix $p_{Y|X_A,X_B}$ will be 0-1-valued.

Alice and Bob must each decide on an action based on the trace they have seen thus far, so a strategy for Alice is a function

$$x_A : (\Sigma_A \times \Gamma_A)^* \to \Sigma_A$$

(and similarly a strategy for Bob is a function $x_B : (\Sigma_B \times \Gamma_B)^* \to \Sigma_B$). Recall that we can consider Alice and Bob to have tossed any necessary coins before beginning the protocol, and so it suffices to consider deterministic strategies for each.

Given an SDFT $\mathcal{T}$, and strategies x_A and x_B for Alice and Bob respectively, what output or outputs can be shown to Bob? We consider first the case where $F = Q$, postponing for later the issues that arise when $F \subsetneq Q$.

Definition 5.5. We will say that a word

$$w = ((a_1, a'_1), (b_1, b'_1)) \dots ((a_k, a'_k), (b_k, b'_k)) \in (\Sigma \times \Gamma)^* = ((\Sigma_A \times \Sigma_B) \times (\Gamma_A \times \Gamma_B))^*$$

(so $a_i \in \Sigma_A, a'_i \in \Sigma_B, b_i \in \Gamma_A$ and $b'_i \in \Gamma_B$) is *consistent* with SDFT $\mathcal{T}$ and strategies x_A, x_B if

(i) $w \in \mathcal{L}(\mathcal{T})$, and

(ii) for every $1 \leq i \leq k$ we have $a_i = x_A((a_1, b_1), \dots, (a_{k-1}, b_{k-1}))$, and $a'_i = x_B((a'_1, b'_1), \dots, (a'_{k-1}, b'_{k-1}))$.

A word $(a'_1, b'_1) \dots (a'_k, b'_k) \in (\Sigma_B \times \Gamma_B)$ is consistent with $\mathcal{T}, x_A$ and x_B if there exist $a_1, \dots, a_k \in \Sigma_A$ and $b_1, \dots, b_k \in \Gamma_A$ such that $(((a_1, a'_1), (b_1, b'_1)) \dots ((a_k, a'_k), (b_k, b'_k)))$ is consistent with $\mathcal{T}, x_A$ and x_B.

We will sometimes refer to limb (ii) of the above Definition as 'being consistent with x_A, x_B'; then being consistent with $\mathcal{T}, x_A, x_B$ means being an element of $\mathcal{L}(\mathcal{T})$ and being consistent with x_A, x_B.

Could we choose to have $Y = (\Sigma_B \times \Gamma_B)^*$, and say that $p_{Y|X_A,X_B}(y|x_A, x_B) = 1$ if y is consistent with $\mathcal{T}, x_A$ and x_B?

No, because such a y may not be unique, and so the matrix $p_{Y|X_A,X_B}(y|x_A, x_B)$ would not in general be stochastic. For example, in the relay system shown in Figure

5.3, where x_A and x_B are both constant a, we have that $(a,a)^k$ is consistent with $\mathcal{T}, x_A$ and x_B for all k. But prefixes are the only way this can happen.

Proposition 5.6. *Let $\mathcal{T}$ be an SDFT as above and let x_A, x_B be strategies for Alice and Bob. Then there exists some $w_0 \in (\Sigma \times \Gamma)^\omega$ such that for any $w \in \mathcal{L}(\mathcal{T})$ we have that w is consistent with x_A and x_B if and only if $w \leq w_0$.*

Proof. Define the infinite word $w_0 = ((a_1, a_1'), (b_1, b_1'))((a_2, a_2'), (b_2, b_2')) \ldots \in (\Sigma \times \Gamma)^\omega$ by

$$\begin{aligned} a_i &= x_A((a_1, b_1) \ldots (a_{i-1}, b_{i-1})), \\ a_i' &= x_B((a_1', b_1') \ldots (a_{i-1}', b_{i-1}')), \text{ and} \\ (b_i, b_i') &= \sigma(q_{i-1}, (a_i, a_i')), \end{aligned}$$

where q_0 is the initial state and the sequence $q_0 q_1 \ldots$ is defined by $q_i = \delta(q_{i-1}, (a_i, a_i'))$ for $i \geq 1$.

Clearly if $w \leq w_0$ then w satisfies limb (ii) of Definition 5.5, and so if also $w \in \mathcal{L}(\mathcal{T})$ then w is consistent with $\mathcal{T}, x_A$ and x_B.

Conversely suppose that $w \not\leq w_0$. Then we have that $w = w'(a,b)w''$ for some $w' = ((a_1, a_1'), (b_1, b_1')) \ldots ((a_k, a_k'), (b_k, b_k')) \leq w_0$, some $w'' \in \Sigma \times \Gamma^*$ and some $(a,b) \in \Sigma \times \Gamma$ with $(a,b) \neq ((a_{k+1}, a_{k+1}'), (b_{k+1}, b_{k+1}'))$. But if $a \neq (a_{k+1}, a_{k+1}')$ then without loss of generality we have $\mathrm{fst}(a) \neq a_{k+1} = x_A((a_1, b_1) \ldots (a_k, b_k))$ and so w is not consistent with x_A, x_B.

On the other hand if $b \neq (b_{k+1}, b_{k+1}') = \sigma(q_k, (a_k, a_k'))$ then $w \notin \mathcal{L}(\mathcal{T})$. Either way we have that w is not consistent with $\mathcal{T}, x_A$ and x_B. □

Projecting w_0 onto $(\Sigma_B \times \Gamma_B)^\omega$ gives

Corollary 5.7. *Let $\mathcal{T}, x_A$ and x_B be as above. There exists some $w_0 \in (\Sigma_B \times \Gamma_B)^\omega$ such that if $w \in (\Sigma_B \times \Gamma_B)^*$ is consistent with $\mathcal{T}, x_A$ and x_B then $w \leq w_0$.*

So can we have $Y = (\Sigma_B \times \Gamma_B)^\omega$, and $p_{Y|X_A,X_B}(y|x_A, x_B) = 1$ for $y = w_0$ as in Corollary 5.7?

One reason why not is that in Chapter 4 we have assumed that the sets $\mathcal{X}_A, \mathcal{X}_B$ and Y are all finite. A more substantial reason is that it is not at all realistic: it corresponds to Bob being able to conduct an experiment lasting for an infinite time. Moreover it would allow Bob to acquire an infinite (or at least unbounded) amount of information, and it is not clear how this should be interpreted.

For this reason we will consider Bob's interaction with the system not as a single experiment, but as a *family* of experiments, parametrised by the amount of time allowed; that is, by the length of traces which we consider as outcomes. Assuming for the moment that $F = Q$, we then have that the matrix $p_{Y|X_A,X_B}(y|x_A, x_B)$ is stochastic.

Proposition 5.8. *Let $\mathcal{T}$ be an SDFST with $F = Q$, and let $Y = (\Sigma_B \times \Gamma_B)^k$ for some $k \in \mathbb{N}$. Let the matrix $p_{Y|X_A,X_B}(y|x_A, x_B)$ be defined by $p_{Y|X_A,X_B}(y|x_A, x_B) = 1$ if y is compatible with $\mathcal{T}, x_A$ and x_B, and 0 otherwise. Then $p_{Y|X_A,X_B}(y|x_A, x_B)$ is stochastic; that is, we have*

$$\sum_{y \in Y} p_{Y|X_A,X_B}(y|x_A, x_B) = 1$$

for all $x_A \in \mathcal{X}_A$ and $x_B \in \mathcal{X}_B$.

Proof. By Corollary 5.7, we have that for fixed x_A, x_B there is at most one $y \in (\Sigma_B \times \Gamma_B)^k$ which is consistent with $\mathcal{T}, x_A$ and x_B. On the other hand it is clear from the definitions that if $F = Q$ then all prefixes of the infinite word w_0 from Proposition 5.6 are accepted by $\mathcal{T}$. Hence projecting w_0 onto $(\Sigma_B \times \Gamma_B)^k$ gives a suitable y. □

Truncating at length k also means that strategies x_A, x_B can be viewed as drawn from the spaces of functions $(\Sigma_A \times \Gamma_A)^{<k} \to \Sigma_A$ and $(\Sigma_B \times \Gamma_B)^{<k} \to \Sigma_B$ respectively. This means that the spaces $\mathcal{X}_A$ and $\mathcal{X}_B$ of possible strategies for Alice and Bob are also finite.

We are therefore able apply Theorem 4.12 to calculate the information flow as the size of the largest possible set of outcomes that can consistently be seen by Bob, and for convenience we will adopt this as a definition.

Definition 5.9. Let $\mathcal{T}$ be an SDFST over input and output alphabets $\Sigma_A \times \Sigma_B$, and let $\mathcal{X}_A, \mathcal{X}_B$ be the spaces of functions $(\Sigma_A \times \Gamma_A)^* \to \Sigma_A$ and $(\Sigma_B \times \Gamma_B)^* \to \Sigma_B$ respectively. Define

$$I_k = \max_{x_B \in \mathcal{X}_B} \log \left| \left\{ y \in (\Sigma_B \times \Gamma_B)^k \middle| \exists\, x_A \in \mathcal{X}_A : y \text{ is consistent with } \mathcal{T}, x_A \text{ and } x_B \right\} \right|.$$

What about the case where $F \subsetneq Q$? The treatment of this depends on what we consider to be the meaning of a run ending in a non-accepting state. One interpretation is that it represents a catastrophically bad outcome (say, the intruder being detected) which must be avoided. By Proposition 4.7 we may assume that Bob is employing a pure (i.e. non-random) strategy, and so Alice can ensure that non-accepting

runs are avoided by avoiding particular x_A. This means that Definition 5.9 is exactly right for this interpretation.

Another possible interpretation is that a run ending in a non-accepting state produces some kind of 'error' output, where all errors are indistinguishable. This essentially increases the number of possible observations by Bob by either 1 or 0, depending on whether or not the extremal x_B allows for non-accepting runs. This means that the amount of information is either I_k or $\log(1+\exp I_k)$, which we consider to be a trivial difference.

A third possiblity of course is that we reject the very notion of a non-accepting run, and consider only SDFSTs with $F = Q$. Note that many kinds of behaviour which may involve the system going into an 'error' state and producing only a fixed 'dummy' output symbol straightforwardly be modeled as an SDFST with $F = Q$.

Which of these three options the reader considers most satisfactory is, to some extent, a matter of personal taste. However, since as noted above all are modeled adequately by Definition 5.9, that is what we shall adopt as the basic definition for the remainder of this analysis.

Definition 5.9 is in some sense an intensional definition, in the sense that it involves directly considering all possible strategies for Alice and Bob. It will be helpful to have a more extensional version: Definition 5.9 can be recast as

$$I_k = \max_{X \in \mathcal{F}} \log |X|,$$

where $\mathcal{F} \subseteq \mathcal{P}((\Sigma_B \times \Gamma_B)^k)$ is the family of sets X such that there exists some $x_B \in \mathcal{X}_B$ such that $X = \left\{ y \in (\Sigma_B \times \Gamma_B)^k \middle| \exists\, x_A \in \mathcal{X}_A : y \text{ is consistent with } \mathcal{T}, x_A \text{ and } x_B \right\}$. So having an extensional characterisation of I_k amounts to having a condition for a set X to be a member of $\mathcal{F}$.

Theorem 5.10. *Let $\mathcal{T}, \mathcal{X}_A$ and $\mathcal{X}_B$ be as above. Let $\mathcal{F} \subseteq \mathcal{P}\left((\Sigma_B \times \Gamma_B)^*\right)$ be defined by $Y \in \mathcal{F}$ if and only if there exists some $x_B \in \mathcal{X}_B$ such that*

$$Y = \{ y \in (\Sigma_B \times \Gamma_B)^* | \exists\, x_A \in \mathcal{X}_A : y \text{ is consistent with } \mathcal{T}, x_A \text{ and } x_B \}.$$

Let $X \subseteq (\Sigma_B \times \Gamma_B)^*$ be arbitrary. Then $X \subseteq X'$ for some $X' \in \mathcal{F}$ if and only if

(i) $X \subseteq \mathcal{L}(\mathcal{T})|_{(\Sigma_B \times \Gamma_B)^*}$, and

(ii) X does not contain two elements which first differ by an element of Σ_B. That is, we do not have $w_1, w_2 \in X$ such that $w_1 = w(a,b)w'$ and $w_2 = w(a',b')w''$ with $w, w', w'' \in (\Sigma_B \times \Gamma)^*, a, a' \in \Sigma_B$ and $b, b' \in \Gamma_B$ with $a \neq a'$,

where the notation $\mathcal{L}(\mathcal{T})|_{(\Sigma_B \times \Gamma_B)^*}$ *means the projection of* $\mathcal{L}(\mathcal{T}) \subseteq ((\Sigma_A \times \Sigma_B) \times (\Gamma_A \times \Gamma_B))^*$ *onto the set* $(\Sigma_B \times \Gamma_B)^*$.

Proof. The 'only if' direction is straightforward. Part (i) is immediate from the definitions, and for part (ii) we must have $a = x_B(w) = a'$ (for the relevant x_B).

For the 'if' direction, suppose that X satisfies the two conditions in the statement of the theorem. Define the partial function $x : (\Sigma_B \times \Gamma_B)^* \rightharpoonup \Sigma_B$ by $x(w') = a$ whenever $w'(a, b) \leq w$ for some $w \in X$ and some $b \in \Gamma_B$. This is well-defined by condition (ii). Define $x_B : (\Sigma_B \times \Gamma_B)^* \to \Sigma_B$ to be x, extended arbitrarily where x is undefined. We claim that

$$X \subseteq Y = \{y \in (\Sigma_B \times \Gamma_B)^* | \exists\, x_A \in \mathcal{X}_A : y \text{ is consistent with } \mathcal{T}, x_A \text{ and } x_B\}\,.$$

Indeed, let $w \in X$ be arbitrary. Plainly w is consistent with x_B. Since $w \in \mathcal{L}(\mathcal{T})|_{(\Sigma_B \times \Gamma_B)^*}$, there exists some $w' \in \mathcal{L}(\mathcal{T})$ such that $w'|_{(\Sigma_B \times \Gamma_B)^*} = w$. Define the partial function $x' : (\Sigma_A \times \Gamma_A)^* \rightharpoonup \Sigma_A$ by $x'(w'') = a$ whenever $w''(a, b) \leq w'$ for some $b \in \Gamma_A$. Let $x_A : (\Sigma_A \times \Gamma_A)^* \to \Sigma_A$ be an arbitrary total extension of x'. Then plainly w' is consistent with x_A, and is also consistent with x_B since w was. Hence w is consistent with $\mathcal{T}, x_A$ and x_B, as required. □

Truncating to length k, and observing that

$$\max_{X \in \mathcal{F}} \log |X| = \max_{X \subseteq X' \in \mathcal{F}} \log |X|$$

gives

Corollary 5.11. *Let* $\mathcal{T}, \mathcal{X}_A$ *and* $\mathcal{X}_B$ *be as above. Then we have*

$$I_k = \max_{X \in \mathcal{F}'_k} \log |X|,$$

where $\mathcal{F}'_k \subseteq \mathcal{P}\left(\mathcal{L}(\mathcal{T})_{=k}|_{(\Sigma_B \times \Gamma_B)^k}\right)$ *is the collection of sets which do not contain two words which first differ by an element of* Σ_B *(and this has the same meaning as in part (iii) of Theorem 5.10).*

For example, for the system shown in Figure 5.3, we have that

$$\mathcal{L}(\mathcal{T})_{=k}|_{(\Sigma_B \times \Gamma_B)^k} = (\Sigma_B \times \Gamma_B)^k\,,$$

and so $I_k = k \log |\Gamma_B|$.

5.4 Reduction to automata

In this section, we show how to reduce the problem of computing I_k from a problem about transducers to a problem which mentions only automata. The first step is to produce an automaton whose language is in correspondence with Bob's interface with $\mathcal{T}$.

Definition 5.12. Let $\mathcal{T} = (Q, q_0, F, \Sigma_A \times \Sigma_B, \Gamma_A \times \Gamma_B, \delta, \sigma)$ be an SDFST. Define the nondeterministic finite automaton $\mathcal{A}_\mathcal{T} = (Q \cup (Q \times \Gamma_B), q_0, F, \Sigma_B \cup \Gamma_B, \Delta)$, where

$$\Delta(q, a') = \{(\delta(q, (a, a')), \mathrm{snd}(\sigma(q, (a, a'))))|a \in \Sigma_A\}$$

for all $q \in Q$ and $a' \in \Sigma_B$, $\Delta(q, b') = \emptyset$ for all $b' \in \Gamma_B$, and

$$\Delta((q, b'), x) = \begin{cases} \{q\} & \text{if } x = b' \\ \emptyset & \text{otherwise} \end{cases}$$

for all $(q, b') \in Q \times \Gamma_B$ and $x \in \Sigma_B \cup \Gamma_B$.

Informally, we introduce an auxiliary state for each pair $(q, b') \in Q \times \Gamma_B$ to represent the behaviour 'emit the event b' and then go into state q'. For states $q, q' \in Q$ and events $a' \in \Sigma_B, b' \in \Gamma_B$ we have a transition from q to (q', b') if and only there exist some $a \in \Sigma_A$ and $b \in \Gamma_A$ such that $\delta(q, (a, a')) = q'$ and $\sigma(q, (a, a')) = (b, b')$. In the language of CSP, this corresponds to hiding all of Alice's events: that is, the familiar *lazy abstraction* formulation of noninterference.

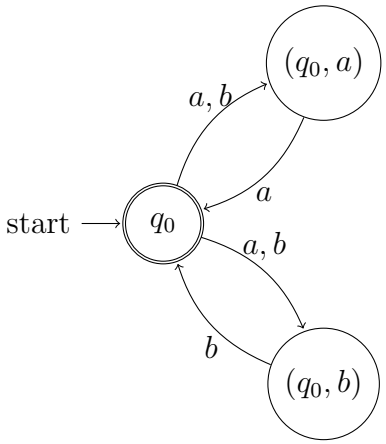

Figure 5.4: Automaton corresponding to the relay system transducer shown in Figure 5.3.

The following lemma is immediate from the definitions, and expresses the fact that the words accepted by $\mathcal{A}_\mathcal{T}$ are in precise correspondence with the words accepted by $\mathcal{T}$, projected onto $\Sigma_B \times \Gamma_B$.

Lemma 5.13. *Let* $f : (\Sigma_B \times \Gamma_B)^* \to (\Sigma_B \cup \Gamma_B)^*$ *be the* flattening *operation defined by* $f((a_1, b_1) \dots (a_k, b_k)) = a_1 b_1 \dots a_k b_k$. *Then we have*

$$\mathcal{L}(\mathcal{A}_\mathcal{T}) = f\left(\mathcal{L}(\mathcal{T})|_{(\Sigma_B \times \Gamma_B)^*}\right).$$

Note that since elements of Σ_B appear at odd-numbered positions in traces of $\mathcal{A}_\mathcal{T}$ and elements of Γ_B appear at even-numbered positions, we may assume without loss of generality that Σ_B and Γ_B are disjoint. Then combining Lemma 5.13 with Corollary 5.11 gives

Theorem 5.14. *Let* $\mathcal{T}$ *be an SDFT as above such that* Σ_B *and* Γ_B *are disjoint. Then*

$$I_k = \max_{X \in \mathcal{F}_k} \log |X|,$$

where $\mathcal{F}_k \subseteq \mathcal{P}\left(\mathcal{L}\left(\mathcal{A}_\mathcal{T}\right)_{=2k}\right)$ *is the collections of sets which do not contain two words which first differ by an element of* Σ_B*; that is, for* $X \in \mathcal{F}_k$ *we do not have* $w_1, w_2 \in X$ *with* $w_1 = waw', w_2 = wa'w''$, *with* $w, w', w'' \in (\Sigma_B \cup \Gamma_B)^*$ *and* $a \neq a' \in \Sigma_B$.

Note that an alternative notation for this theorem (and, *mutatis mutandis*, Corollary 5.11) would be to define a single family $\mathcal{F} \subseteq \mathcal{P}\left((\Sigma_B \cup \Gamma_B)^*\right)$ consisting of the sets which do not contain words first differing on an element of Σ_B, and then say that

$$I_k = \max_{X \in \mathcal{F}} \log |X \cap \mathcal{L}(\mathcal{A}_\mathcal{T})_{=2k}|.$$

We have therefore reduced computing the information flow permitted by a deterministic interactive system to an instance of a more general problem over finite automata, which we call the Σ-*deterministic subset growth* problem.

Definition 5.15. Let Σ, Γ be disjoint finite sets. A set $X \subseteq (\Sigma \cup \Gamma)^*$ is Σ-*deterministic* if it does not contain two words which first differ by an element of Σ; that is, we do not have $w_1, w_2 \in X$ with $w_1 = waw', w_2 = wa'w''$, with $w, w', w'' \in (\Sigma \cup \Gamma)^*$ and $a \neq a' \in \Sigma$.

For a nondeterministic finite automaton $\mathring{A}$ over alphabet $\Sigma \cup \Gamma$, define

$$D_k(\mathcal{A}) = \max_{X \in \mathcal{F}_k} |X|,$$

where $\mathcal{F}_k$ consists of the Σ-deterministic subsets of $\mathcal{L}(\mathcal{A})_{=k}$.

Problem 5.16 (Σ-deterministic subset growth)**.** *Given a nondeterministic finite automaton* $\mathcal{A}$ *over* $\Sigma \cup \Gamma$*, determine the growth rate of* $D_k(\mathcal{A})$.

Of course, the statement of this problem is somewhat informal, in that the meaning of 'determine the growth rate' is not precisely specified. This is in some sense inevitable, considering that $D_k(\mathcal{A})$ is an infinite collection of values. The precise interpretation of the problem will emerge from the results we obtain.

We may also view Problem 5.16 as a direct representation of an information flow problem, which we refer to as the *locked room model*. We consider that Alice is locked inside a room, in possession of a secret which she wishes to convey to Bob outside. Alice and Bob are able to communicate by passing messages, drawn from alphabets Γ and Σ respectively, subject to the constraint that the transcript of their communication must lie in some fixed language L.

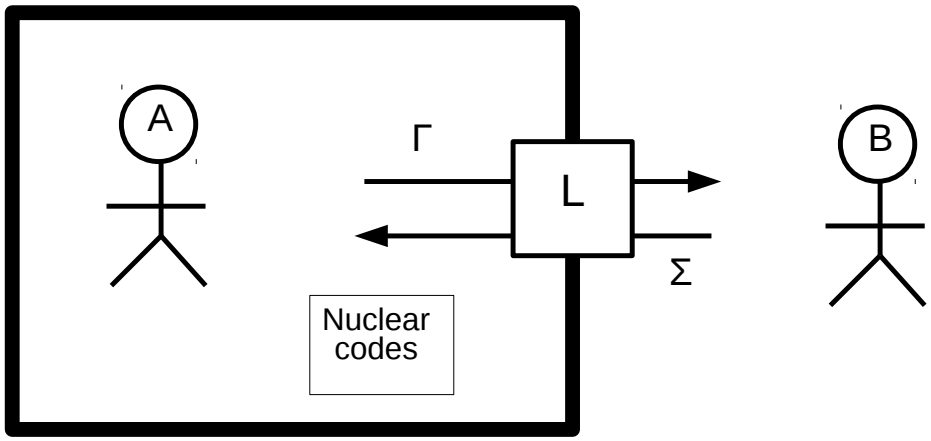

Figure 5.5: The locked room model

Bob must choose a strategy, which is a map $f : (\Sigma \cup \Gamma)^* \to \Gamma$, and a set of transcripts is consistent with the strategy if and only if whenever wa is a prefix of an element of the set, with $w \in (\Sigma \cup \Gamma)^*$ and $a \in \Sigma$ we have $a = f(w)$. The information obtained by Bob is the logarithm of the size of the set of transcripts consistent with his strategy.

Now we have that a set of transcripts is a subset of the set arising from some strategy if and only if it is Σ-deterministic, so computing the information flow to Bob is precisely Problem 5.16.

Note that this model assumes that Alice is able to choose between sending a message herself or waiting for one from Bob; if we prefer not to allow this then the condition becomes that X is valid if any two elements of X first differ by an element of Γ; the difference between this and the definition of Σ-deterministic given above is that it does not allow $waw', wbw'' \in X$, for $w, w', w'' \in (\Sigma \cup \Gamma)^*$, $a \in \Sigma$ and $b \in \Gamma$. We

shall see that the theory developed to handle Problem 5.16 is equally able to handle this choice.

5.5 Antichains

The purpose of this section is to show that Problem 5.16 can itself be reduced to a rather natural combinatorial one.

Definition 5.17. Let X be a set, and let $\leq$ be a partial order on X. Then the *lexicographic* order induced by $\leq$ on X^*, denoted $\preceq$, is defined by

$$\begin{aligned} \forall\, w \in X^* : \epsilon \preceq w \text{ (and } w \not\preceq \epsilon \text{ if } w \neq \epsilon\text{), and} \\ \forall\, x, y \in X, w, w' \in X^* : xw \preceq yw' \text{ if and only if either } x < y \\ \text{or } x = y \text{ and } w \preceq w'. \end{aligned}$$

Observe that $\preceq$ defines a partial order. Indeed, suppose that $w_1, w_2 \in X^*$ are of minimum total length such that $w_1 \preceq w_2, w_2 \preceq w_1$ but $w_1 \neq w_2$. Trivially if $w_1 = \epsilon$ then also $w_2 = \epsilon = w_1$ (and vice versa). Otherwise we have $w_1 = xw, w_2 = yw'$, and either $x < y$ or $x = y$ and $w \preceq w'$, and on the other hand either $y < x$ or $y = x$ and $w' \preceq w$. Hence we have $y = x$ and both $w \preceq w'$ and $w' \preceq w$, so by induction $w = w'$. Hence $w_1 = w_2$, a contradiction, so indeed $\preceq$ is antisymmetric.

Similarly suppose that $w_1, w_2, w_3 \in X^*$ are of minimum total length such that $w_1 \preceq w_2$ and $w_2 \preceq w_3$ but $w_1 \not\preceq w_3$. Since $w_1 \not\preceq w_3$ we have $w_1 \neq \epsilon$, and plainly $w_1 \neq w_2$ and $w_2 \neq w_3$ and so $w_2, w_3 \neq \epsilon$. Write $w_1 = xw, w_2 = yw'$ and $w_3 = zw''$. If $x < y$ then (since $y \leq z$) we have $x < z$ and so $w_1 \preceq w_3$. Similarly if $y < z$ then (since $x \leq y$) we have $x < x$ so $w_1 \preceq w_3$. Hence we have $x = y = z$ and $w \preceq w'$ and $w' \preceq w''$. But then by induction we have $w \preceq w''$ and so $w_1 \preceq w_3$, a contradiction. Hence indeed $\preceq$ is transitive and so (since we have also shown it is antisymmetric, and it is trivially reflexive) it is a partial order.

The study of partially ordered sets is often concerned with *chains* (sets wherein any two elements are comparable) and *antichains* (sets where no two elements are comparable).

Definition 5.18. Let X be a partially ordered set, partially orderd by $\leq$. A set $Y \subseteq X$ is a *chain* if for any $x, y \in Y$ we have $x \leq y$ or $y \leq x$. Y is an *antichain* if for any $x, y \in Y$ such that $x \leq y$ we have $x = y$. Let $Y \subseteq X$ be an antichain of maximum size. Then $|Y|$ is the *width* of X, denoted $w(X)$.

An example of the relevance of the width of a partially ordered set to its structure, is given by the celebrated theorem of Robert Dilworth [22].

Theorem 5.19 (Dilworth, 1950)**.** *Let X be a partially ordered set. Let k be minimal such that $X = Y_1 \cup \ldots \cup Y_k$ with each Y_k a chain. Then $k = w(X)$.*

The relevance of these ideas to Problem 5.16 is established by the following theorem.

Theorem 5.20. *Let Σ, Γ be disjoint sets. Define the partial order $\leq$ on $\Sigma \cup \Gamma$ by setting $\leq|_\Sigma$ to be an arbitrary linear order on Σ, and setting $x \not\leq y, y \not\leq x$ for all $x \in \Gamma$ and all $y \in \Sigma \cup \Gamma$ with $y \neq x$.*

Let $X \subseteq (\Sigma \cup \Gamma)^k$ be arbitrary. Then X is Σ-deterministic if and only if it is an antichain with respect to the lexicographic order induced by $\leq$.

Proof. If $w_1, w_2 \in (\Sigma \cup \Gamma)^k$ first differ by an element of Σ, say $w_1 = waw'$, $w_2 = wa'w''$ with $a \neq a' \in \Sigma$. Then without loss of generality $a < a'$, so $w_1 \preceq w_2$. Conversely, if $w_1 \preceq w_2$, then write $w_1 = wxw', w_2 = wyw''$ for some $x \neq y \in \Sigma \cup \Gamma$. But then we must have $x < y$, and hence $x, y \in \Sigma$ and so w_1, w_2 first differ by an element of Σ. □

We have thus reduced Problem 5.16 to the problem of calculating the growth rate of the width of a regular language, with respect to this partial order, a special case of the following problem.

Problem 5.21 (Antichain growth for NFA)**.** *Given a nondeterministic finite automaton $\mathcal{A}$ over a finite partially ordered set $(\Sigma, \leq)$, determine the growth rate of $w\left(\mathcal{L}(\mathcal{A})_{=k}\right)$, with respect to the lexicographic order.*

Note that to handle the variant of the 'locked room' problem discussed above where we do not allow Alice to choose whether to allow Bob to communicate, and thus view sets as consistent if their elements first differ by an element of Γ, we simply expand the partial order $\leq$ by setting (say) $a \leq b$ for every $a \in \Sigma, b \in \Gamma$.

We conclude this section with a diagram showing the stages of the reduction we have established for deterministic interactive finite-state systems from mutual information in the information-theoretic sense to antichains in regular languages.

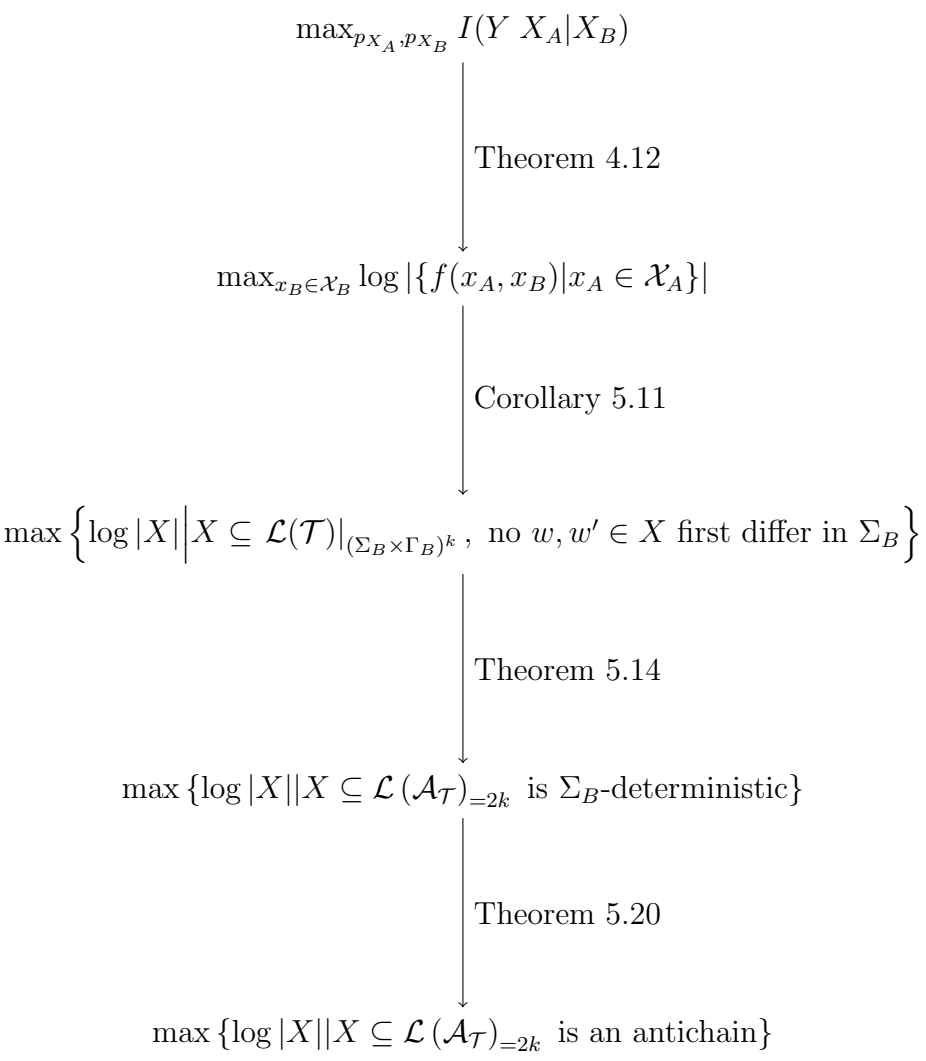

Figure 5.6: The structure of Chapters 4 and 5.

5.6 Asynchronous systems

In this section, we consider the case of an asynchronous system; that is, a system which accepts a sequence of commands coming from *either* Alice *or* Bob, instead of recieving at each time-step a command from both.

The specification of the system itself is straightforward: an SDFST where as before the output alphabet Γ is of the form $\Gamma_A \times \Gamma_B$, but now the input alphabet Σ is of the form $\Sigma_A \sqcup \Sigma_B$ (the disjoint union of Σ_A and Σ_B). This precisely matches the definition of Goguen-Meseguer (G-M).

A more challenging problem is to determine what we consider Alice and Bob to be able to observe about the system, what we consider to be 'strategies', and how the system resolves these into a run. In terms of observations, we have a clear choice

between allowing Alice and Bob to observe the fact that the other has taken an action (which has not changed their output), and not doing so. In the spirit of G-M, we prefer not to do so, which means (see Theorem 2.4) that we allow at least Bob to see only stutter traces.

There is a technical problem with allowing Alice to see only stutter traces, which becomes apparent when we consider the next problem of defining strategies. As before, we should require Alice and Bob at each moment to choose the next event they wish to perform. But what should determine which of the two happens? Perhaps one natural view is that the system nondeterministically chooses one of the two possibilities. But at the moment we are modelling only deterministic systems and so this is not an option. The most conservative solution, then, is to allow Alice to choose between inserting an event and waiting for Bob to perform one.

However, suppose that under this model Alice chooses to allow Bob to take an action. If none of Bob's actions changes the output to Alice then she will never see a change, and so will always just allow Bob to act and never take any further actions herself. This is clearly undesirable. Consequently we should allow Alice to observe full traces, or, equivalently to see that Bob has communicated an event even if the output to her has not changed. The asymmetry of this definition may seem unsatisfying, but as before we will find that Alice in some sense drops out of the picture.

A strategy for Bob will be a function

$$x_B : (\Sigma_B \sqcup \Gamma_B)^* \to \Sigma_B.$$

Note that the domain of this function is $(\Sigma_B \sqcup \Gamma_B)^*$ rather than $(\Sigma_B \times \Gamma_B)^*$ because Bob may see many outputs between consecutive inputs.

However, Alice must also have the ability to allow Bob to take an action, which we represent by a special 'action' $\perp$, and so we define a strategy for Alice as a function

$$x_A : (\Sigma_A \sqcup \Gamma_A)^* \to \Sigma_A \sqcup \{\perp\}.$$

Definition 5.22. We say that a word

$$w = (a_1, (b_1, b_1')) \dots (a_k, (b_k, b_k')) \in (\Sigma \times \Gamma)^* = ((\Sigma_A \sqcup \Sigma_B) \times (\Gamma_A \times \Gamma_B))^*$$

is *consistent* with SDFT $\mathcal{T}$ and strategies x_A, x_B if $w \in \mathcal{L}(\mathcal{T})$ and for all $1 \leq i \leq k$ we have that if $a_i \in \Sigma_A$ then

$$a_i = x_A(\phi((a_1, (b_1, b_1')) \dots (a_{k-1}, (b_{k-1}, b_{k-1}'))))),$$

and if $a_i \in \Sigma_B$ then

$$a_i = x_B(\psi((a_1, (b_1, b'_1)) \dots (a_{k-1}, (b_{k-1}, b'_{k-1}))))$$

and

$$x_A(\phi((a_1, (b_1, b'_1)) \dots (a_{k-1}, (b_{k-1}, b'_{k-1})))) = \perp.$$

In the above ϕ is the obvious map given by flattening and then removing all elements of Σ_B and Γ_B; concretely, if we have

$$w = (a_0^1, (b_0^1, b_0'^1)) \dots (a_0^{n_0}, (b_0^{n_0}, b_0'^{n_0}))(a_1, (b_1, b'_1))(a_1^1, (b_1^1, b_1'^1)) \dots (a_1^{n_1}, (b_1^{n_1}, b_1'^{n_1})) \dots \\ (a_k, (b_k, b'_k))(a_k^1, (b_k^1, b_k'^1)) \dots (a_k^{n_k}, (b_k^{n_k}, b_k'^{n_k})),$$

with $a_1, \dots, a_k \in \Sigma_A$ and $a_i^j \in \Sigma_B$ for all i, j, then

$$\phi(w) = b_0^1 \dots b_0^{n_0} a_1 b_1 b_1^1 \dots b_1^{n_1} \dots a_k b_k b_k^1 \dots b_k^{n_k}.$$

On the other hand ψ is defined by flattening, removing all elements of Σ_A and Γ_A, and then taking the stutter trace; that is, removing repeated elements of Γ_A; again, concretely if

$$w = (a_0^1, (b_0^1, b_0'^1)) \dots (a_0^{n_0}, (b_0^{n_0}, b_0'^{n_0}))(a_1, (b_1, b'_1))(a_1^1, (b_1^1, b_1'^1)) \dots (a_1^{n_1}, (b_1^{n_1}, b_1'^{n_1})) \dots \\ (a_k, (b_k, b'_k))(a_k^1, (b_k^1, b_k'^1)) \dots (a_k^{n_k}, (b_k^{n_k}, b_k'^{n_k})),$$

where now we have $a_1, \dots, a_k \in \Sigma_B$ and $a_i^j \in \Sigma_A$ for all i, j, then

$$\psi(w) = \text{stutt}(b_0'^1 \dots b_0'^{n_0} a_1 b_1 b_1^1 \dots b_1^{n_1} \dots a_k b_k b_k^1 \dots b_k^{n_k}),$$

where stutt : $(\Sigma_B \sqcup \Gamma_B)^* \to (\Sigma_B \sqcup \Gamma_B)^*$ is the function that removes consecutive duplicate letters.

We will take the set Y of outcomes visible to Bob to be the set of stutter traces of length k, and we will have $p_{Y|X_A,X_B}(y|x_A, x_B) = 1$ if y is compatible with x_A and x_B and 0 otherwise. As before there is a unique stutter trace of length k compatible with fixed x_A, x_B.

We can thus apply Theorem 4.12 to conclude that the maximum information flow is the logarithm of the size of the maximum set $X \subseteq Y$ such that there is some single x_B and for each $y \in X$ some x_A such that y is compatible with x_A, x_B.

We have a theorem analagous to Theorem 5.10, giving a condition for a set to be consistent with a fixed strategy x_B.

Theorem 5.23. *Let $\mathcal{F} \subseteq \mathcal{P}((\Sigma_B \sqcup \Gamma_B)^*)$ be defined by $Y \in \mathcal{F}$ if and only if there exists some x_B such that*

$$Y = \{y \in (\Sigma_B \sqcup \Gamma_B)^* | \exists\, x_A : y \text{ is consistent with } x_A, x_B.\}\,.$$

Let $X \subseteq (\Sigma_B \sqcup \Gamma_B)^*$ be arbitrary. then $X \subseteq X'$ for some $X' \in \mathcal{F}$ if and only if

(i) $X \subseteq \psi(\mathcal{L}(\mathcal{T}))$, where ψ is defined as in Definition 5.22, and

(ii) X does not contain two elements which first differ by an element of Σ_B. That is, we do not have $w_1, w_2 \in X$ such that $w_1 = waw'$ and $w_2 = wa'w''$ with $w, w', w'' \in (\Sigma_B \sqcup \Gamma_B)^*$ and $a \neq a' \in \Sigma_B$.

Proof. The 'only if' direction is straightforward: we must have $a = x_B(w) = a'$.

Conversely, suppose that X satisfies the two conditions in the statement of the theorem. Define the partial function $x : (\Sigma_B \sqcup \Gamma_B)^* \rightharpoonup \Sigma_B$ by $x(w') = a$ whenever $w'a \leq w$ for some $w \in X$. This is well-defined by condition (ii). Define $x_B : (\Sigma_B \sqcup \Gamma_B)^* \to \Sigma_B$ to be x, extended arbitrarily where x is undefined. We claim that

$$X \subseteq Y = \{y \in (\Sigma_B \sqcup \Gamma_B)^* | \exists\, x_A : y \text{ is consistent with } x_A, x_B\}\,.$$

Indeed, let $w \in X$ be arbitrary. Plainly w is consistent with x_B. Since $w \in \psi(\mathcal{L}(\mathcal{T}))$, there exists some $w' \in \mathcal{L}(\mathcal{T})$ such that $\psi(w') = w$. Define the partial function $x' : (\Sigma_A \sqcup \Gamma_A)^* \rightharpoonup \Sigma_A \sqcup \{\bot\}$ by $x'(w'') = a$ whenever $w''a \leq w'$, and $x'(w'') = \bot$ if $w''b \leq w'$ for some $b \in \Gamma_B$. Let $x_A : (\Sigma_A \times \Gamma_A)^* \to \Sigma_A \sqcup \{\bot\}$ be an arbitrary total extension of x'. Then plainly w' is consistent with x_A, and is also consisitent with x_B since w was. Hence w is consistent with x_A, x_B as required. □

We also have an analogue of Definition 5.12 and Lemma 5.13, producing an automaton accepting $\psi(\mathcal{L}(\mathcal{T}))$. Essentially we produce stutter traces by recording the previous output, and suppressing further instances of the same letter. It will be more convenient to specify an NFA with silent transitions, so that the transition relation is a map $\Delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q)$. There is a trivial construction to remove the silent transitions while preserving the accepted language: for each state $q \in Q$, let X_q be the set of states reachable from q via only silent transitions. Then set $\Delta'(q, a) = \bigcup_{q' \in X_q} \Delta(q', a)$ for all $a \in \Sigma$. Note that this does not add any states and so does not increase the size of the automaton beyond at worst a quadratic factor.

Definition 5.24. Let $\mathcal{T} = (Q, q_0, F, \Sigma_A \sqcup \Sigma_B, \Gamma_A \times \Gamma_B, \delta, \sigma)$ be an SDFST. Define the nondeterministic finite automaton $\mathcal{A}_\mathcal{T} = (\{0,1\} \times Q \times \Gamma_B \cup \{q_0\}, q_0, F, \Sigma_B \sqcup \Gamma_B, \Delta)$, as follows. For all $q \in Q, b \in \Gamma_B$ and $a \in \Sigma_B$ we have

$$\Delta((0,q,b),a) = \{(1, \delta(q,a), \sigma(q,a))\}.$$

For all $q \in Q$ and $b \in \Gamma_B$ we have $\sigma(q,a) \neq b$ then

$$\begin{aligned}\Delta((0,q,b),\epsilon) = \{(1, \delta(q,a), \sigma(q,a)) | a \in \Sigma_A, \sigma(q,a) \neq b\} \\ \cup \{(0, \delta(q,a), b) | a \in \Sigma_A, \sigma(q,a) = b\}.\end{aligned}$$

For all $q \in Q$ and $b \in \Gamma_B$ we have

$$\Delta((1,q,b),b) = \{(0,q,b)\}.$$

Finally, we have

$$\Delta(q_0, \epsilon) = \{(1, \delta(q_0,a), \sigma(q_0,a)) | a \in \Sigma_A\},$$

and for all $a \in \Sigma_B$ we have

$$\Delta(q_0, a) = \{(1, \delta(q_0,a), \sigma(q_0,a)\}.$$

All other values of Δ are $\emptyset$.

We thus have that the information flow permitted by an asynchronous system $\mathcal{T}$ in traces of length k is the logarithm of the size of the largest Σ_B-deterministic set in $\mathcal{L}(\mathcal{A}_\mathcal{T})_{=k}$, so the asynchronous case is also reduced to Problem 5.16, and hence to Problem 5.21.

5.7 Conclusions

The principal accomplishments of this chapter are to have defined information flow for deterministic interactive systems that can be modelled as finite-state transducers, and to have reduced this problem to that of computing the size of the maximum antichains in regular languages specified as NFA. We will show how to solve this problem in Chapter 6.

Chapter 6

Antichains

6.1 Introduction

In this chapter we consider the problem of estimating the sizes of antichains in a variety of partially ordered structures. The main motivation is to solve Problem 5.21 (which requires us to compute the size of maximum antichains in regular languages), and thereby compute information flow for deterministic interactive systems modelled as finite state machines. However, we also consider the problem to be of independent mathematical interest, and so we also study some other structures, namely context-free languages and tree regular languages.

Computing the size of the largest antichain (set of mutually incomparable elements) is the 'central' problem in the extremal combinatorics of partially ordered sets (posets) [68]. In addition to some general theory [36], it has attracted study for a variety of specific sets, beginning with Sperner's Theorem on subsets of $\{1, \ldots, n\}$ ordered by inclusion [64, 13, 50], and for random posets [12]. The size of the largest antichain in a poset L is called the *width* of L.

In this work we study languages (regular or context-free) over finite partially ordered alphabets, with the lexicographic partial order. Since such languages will in general contain infinite antichains, we study the sets $L_{=n}$ of words of length n, and ask how the width of $L_{=n}$ grows with n; we call this the *antichain growth* rate of L.

In Section 6.2 we set out basic definitions of the lexicographic order, antichains and antichain growth. We then develop some of the attractive theory of antichains in this poset, which will be used later.

In Section 6.3 we consider the case of regular languages. We draw heavily on and generalise the ideas of [26], which shows a dichotomy theorem for the growth rate of regular and context-free languages (equivalent to our problem in the special case of the trivial partial order in which all elements are incomparable). We show a dichotomy

theorem between polynomial and exponential antichain growth (Theorem 6.16), and that there is a polynomial time algorithm to distinguish the two cases (Theorem 6.18). This substantially resolves Problem 5.21: this shows that the information flow in time n is either $\Theta(n)$ (which is clearly 'dangerous') or is $O(\log n)$ (which is in some sense 'safe', since the secret could be guessed at this rate in any case), and we can determine in polynomial time which is the case.

In Section 6.4 we consider the problem of computing the antichain growth more precisely. We first observe that there is a simple dynamic programming algorithm to compute the size of the largest antichain in $L_{=n}$, where n is given in unary and the language is specified as a DFA. We then describe an algorithm to compute the order of polynomial antichain growth for a language specified as a DFA. Finally we show that in fact this algorithm also extends to languages specified by NFA.

It is interesting to observe the similarity in flavour between the work of these two sections and the work of Weber and Seidl in [67], in which they study the degree of *ambiguity* of an NFA, which is the maximum number of accepting runs corresponding to a single word of length n. In particular, in Section 4 of [67] they show that this has either polynomial or exponential growth, and give a criterion which is reminiscent of the criterion we shall obtain in Theorem 6.16. They also give an algorithm to compute the order of polynomial growth, in a somewhat similar fashion to the algorithm we obtain in Theorem 6.28.

Note, however, that so far as we can tell there is no logical relationship between the two problems: there is no obvious way to record runs in such a way that antichains are sets of distinct runs corresponding to a single word. On the other hand, it seems unlikely that the antichains problem can be reduced to the ambiguity problem: for instance for DFA the ambiguity is trivial, whereas the antichain growth problem is not.

In Section 6.5, we illustrate the theory of this chapter and the preceding two chapters by applying it to two example systems: a relay system and an 'interrupt' system for a shared resource. We find that the relay system has exponential antichain growth, whereas the interrupt system has polynomial antichian growth. We then apply the algorithm of Section 6.4 to compute the order of polynomial growth for the interrupt system.

In Section 6.6 we move to considering context-free languages. We show a similar dichotomy theorem (Theorem 6.34). However, contrary to both the regular language case and the context-free language growth problem, there is no algorithm to distinguish between polynomial and exponential antichain growth (Theorem 6.40).

In Section 6.7, we generalise the definition of the lexicographic order to extend to tree languages, and show that there is a trichotomy theorem between polynomial, exponential and doubly exponential antichain growth. This also entails as a corollary that there is a similar trichotomy for language growth rate of regular tree languages, which we believe to be an original result.

6.2 Languages, lexicographic order and antichains

In this section we recall the definition of the lexicographic order, and of antichains, which are the primary objects of study of this chapter. For convenience we will need to make a slight change of notation from the previous chapter: we previously used the symbol $\leq$ to denote the lexicographic order on words, whereas we will now use it to denote the prefix relation on words.

We will then develop some of the basic theory of antichains in the lexicographic order: in particular, we establish the effects of prefixing, postfixing, concatenation and the Kleene star operator. We then make the definitions that will be used for the main results of this chapter: exponential antichain growth, and polynomial antichain growth. Although these definitions are determined by the need to match problem 5.21, we give some consideration to possible alternative definitions.

Finally, we prove the most substantial result of this section (Lemma 6.12), which shows that the weaker notion of *quasiantichain* suffices to distinguish exponential from polynomial antichain growth.

Definition 6.1. Let Σ be a finite alphabet equipped with a partial order $\leq$. Then the lexicographic partial order induced by $\leq$ on Σ^* is the relation $\preceq$ given by

(i) $\epsilon \preceq w$ for all $w \in \Sigma^*$ (where ϵ is the empty word), and

(ii) For any $x, y \in \Sigma, w, w' \in \Sigma^*$, we have $xw \preceq yw'$ if and only if either $x < y$ or $x = y$ and $w \preceq w'$.

If words x and y are comparable in this partial order we write $x \sim y$. If x is a prefix of y we write $x \leq y$. It is important to emphasise that $\leq$ is used to denote the partial order on Σ and the prefix order on Σ^*, but *not* the lexicographic order on Σ^*.

For a totally ordered alphabet (such as the Latin A–Z), this is a total order, and is equal to the usual dictionary order, since we adopt via (i) the dictionary convention that prefixes are listed first, so, for instance, 'cat' appears before 'catharsis', since cat $\leq$ catharsis.

In general, the relationship between two words w and w' is determined as follows: if $w \leq w'$ then $w \preceq w'$, and similarly vice versa. Otherwise, find the first position at which w and w' differ; if the letters the two words have in this position are comparable then the words are comparable, in the same order. Otherwise they are incomparable, and we write $w \not\sim w'$. So, for instance, if we have an alphabet where $a < b < c < \ldots$ and $1 < 2 < 3 < \ldots$ but letters and digits are incomparable then we have c3po $\preceq$ r2d2, but $42 \not\sim$ fortytwo.

A critical property of this partial order, which we will use many times below, is that if two words are incomparable then adding anything onto the end of either one cannot change this: if $w_1 \not\sim w_2$ then $w_1 w_1' \not\sim w_2 w_2'$ for all w_1', w_2'. On the other hand, if two words are comparable then this can be for either of two reasons. It could be that one is a prefix of the other, in which case appending further letters could make the two words incomparable, or alternatively reverse their order. However, if neither is a prefix of the other (and so they first differ by two comparable letters) then adding further letters cannot change their order.

For a language L, we will often write $L_{=n}$ to denote the set $\{w \in L \mid |w| = n\}$ (with corresponding definitions for $L_{<n}$, etc.), and $|L|_{=n}$ for $|L_{=n}|$.

The main subject of this chapter is *antichains*, that is sets of words which are mutually incomparable. It will sometimes be useful also to consider *quasiantichains*,[1] which are sets of words which are incomparable except that the set may include prefixes.

Definition 6.2. A language L is an *antichain* if for every $l_1, l_2 \in L$ with $l_1 \neq l_2$ we have $l_1 \not\sim l_2$. A language L is a *quasiantichain* if for every $l_1, l_2 \in L$ we have either $l_1 \leq l_2$, $l_2 \leq l_1$ or $l_1 \not\sim l_2$.

It is easy to see that the property of being an antichain is preserved by the operations of prefixing, postfixing and concatenation.

Lemma 6.3 (Prefixing)**.** *Let w, w_1, w_2 be any words. Then $w_1 \sim w_2$ if and only if $ww_1 \sim ww_2$. Hence for any language L, wL is an antichain (respectively quasiantichain) if and only if L is an antichain (quasiantichain).*

Lemma 6.4 (Postfixing)**.** *Let w, w_1, w_2 be any words. Then $w_1 \sim w_2$ if $w_1 w \sim w_2 w$. Hence for any language L, Lw is an antichain if L is an antichain.*

[1] Note that this is not a standard term.

Lemma 6.5 (Concatenation)**.** *Let w_1, w_2, w_1', w_2' be any words such that $w_1 \not\preceq w_2$ and $w_2 \not\preceq w_1$. Then $w_1 w_1' \sim w_2 w_2'$ if and only if $w_1 \sim w_2$. Hence if L_1 and L_2 are antichains then $L_1 L_2$ is an antichain.*

Clearly the property of being an antichain is not preserved by Kleene star, since L^* will contain prefixes for any non-empty L. The best we can hope for is that L^* is a quasiantichain.

Lemma 6.6 (Kleene star)**.** *Let L be an antichain. Then L^* is a quasiantichain.*

Proof. Suppose $w_1 \sim w_2$ with $w_1, w_2 \in L^*, w_1 \not\preceq w_2$ and $w_2 \not\preceq w_1$ with $|w_1 + w_2|$ minimal. Say $w_i = w_i' w_i''$ with $w_i' \in L, w_i'' \in L^*$. By minimality we have $w_1' \neq w_2'$, and since L is an antichain we also have $w_1' \not\sim w_2'$. Hence by the concatenation lemma $w_1' w_1'' \not\sim w_2' w_2''$, a contradiction. □

Ultimately we are going to care about the size of antichains inside particular languages. Since these will often be unbounded, we choose to ask about the rate of growth; that is, if $L_1, L_2, L_3, \ldots \subseteq L$ are antichains such that L_i consists of words of length i, how quickly can $|L_i|$ grow with i? We will call $\bigcup_i L_i$ an *antichain family* and ask whether it grows exponentially, polynomially, etc.

Definition 6.7. A language L is an *antichain family* if for each n the set $L_{=n}$ of words in L of length n is an antichain.

Definition 6.8. A language L is *exponential* (or *has exponential growth*) if there exists some $\epsilon > 0$ (the *order* of exponential growth) such that

$$\limsup_{n\to\infty} \frac{|L|_{=n}}{2^{\epsilon n}} > 0.$$

L is *polynomial* (or *has polynomial growth*) if there exists some k such that

$$\limsup_{n\to\infty} \frac{|L|_{=n}}{n^k} < \infty.$$

If $0 < \limsup_{n\to\infty} \frac{|L|_{=n}}{n^k} < \infty$ then we say that L has polynomial growth of order k.

For notational convenience, we will sometimes later adopt the convention that a language L which is finite (and so $\limsup_{n\to\infty} \frac{|L|_{=n}}{n^k} = 0$ for all k) has polynomial growth of order -1.

A reasonable alternative choice of notation would have been to define the quantity w_n to be the size of the largest antichain consisting of words of length n, and then ask about the growth of the series $w_1, w_2, \ldots$. This is clearly equivalent to the definitions we have given above.

Note that we will sometimes use other characterisations that are clearly equivalent; for instance L has exponential growth if and only if there is some ϵ such that $|L|_{=n} > 2^{\epsilon n}$ infinitely often. We will sometimes refer to a language which is not polynomial as 'super-polynomial', or as having 'growth beyond all polynomial orders'. Of course there exist languages whose growth rates are neither polynomial nor exponential; for instance $|L|_{=n} = \Theta(2^{\sqrt{n}})$.

Definition 6.9. A language L *has exponential antichain growth* if there is an exponential antichain family $L' \subseteq L$. L *has polynomial antichain growth* if for every antichain family $L' \subseteq L$ we have that L' is polynomial.

Note that we could have chosen (when formulating the general mathematical problem, without the constraint of matching Problem 5.21) to define exponential antichain growth as containing an exponential antichain (rather than an exponential antichain family). We will eventually see (Corollary 6.17) that for regular languages the two notions are equivalent. However, for general languages they are not; indeed the following lemma shows that the two possible definitions are not equivalent even for context-free languages.

Proposition 6.10. *There exists a context-free language L such that L has exponential antichain growth but all antichains in L are finite.*

Proof. Let $\Sigma = \{a, b, 0, 1\}$ with $<= \{(a, b)\}$. Let

$$L = \bigcup_{n=1}^{\infty} L_n = \bigcup_{n=1}^{\infty} a^{n-1} b \{0, 1\}^n.$$

Then each L_n is an antichain of size 2^n consisting of words of length $2n$, but we have $L_1 > L_2 > L_3 > \ldots$ so any antichain is a subset of L_k for some k and hence is finite (the notation $L_1 > L_2$ means that for any $w_1 \in L_1$ and $w_2 \in L_2$ we have $w_2 \preceq w_1$).

Plainly L is a context-free language, establishing the result. □

Note that it is clear by either the Myhill-Nerode Theorem or the Pumping Lemma that the language L constructed in the proof of Proposition 6.10 is not regular.

This example also justifies our choice of definition: a language which contains antichains of unboundedly large (but finite) size should surely be viewed as having 'large' antichains rather than 'small'.

We observed above that Kleene star does not preserve the property of being an antichain. We conclude this section by establishing Lemma 6.12, which addresses this problem; if our goal is to find a large antichain, it suffices to find a large quasiantichain (where the precise meaning of 'large' is having exponential growth).

As a preliminary, we observe the trivial fact that taking finite unions does not change the polynomial or exponential growth character of languages.

Lemma 6.11. *Let $L_1, L_2, \ldots, L_k$ be languages, such that $\bigcup_{i=1}^{k} L_i$ has exponential growth of order ϵ (respectively super-polynomial growth). Then L_i has exponential growth of order ϵ (respectively super-polynomial growth) for some i.*

Proof. Suppose that $\cup_{i=1}^{k} L_i$ has exponential growth of order ϵ. Then we have

$$0 < \limsup_{n\to\infty} \frac{|L|_{=n}}{2^{\epsilon n}} \leq \sum_{i=1}^{k} \limsup_{n\to\infty} \frac{|L_i|_{=n}}{2^{\epsilon n}},$$

and hence we have $\limsup_{n\to\infty} \frac{|L_i|_{=n}}{2^{\epsilon n}} > 0$ for some i.

Similarly, suppose that L has growth beyond all polynomial orders. Then for every m we have

$$\infty = \limsup_{n\to\infty} \frac{|L|_{=n}}{n^m} \geq \max_{i=1,\ldots,k} \limsup_{n\to\infty} \frac{|L_i|_{=n}}{n^m},$$

and hence there is some i_m such that $\limsup_{n\to\infty} \frac{|L_{i_m}|_{=n}}{n^m} = \infty$. Now by the pigeon-hole principle there must be some i such that $i = i_m$ for arbitrarily large m, and so L_i has growth beyond all polynomial orders. □

We are now ready to prove Lemma 6.12. We do this by constructing an exponential prefix-free subset of the exponential quasiantichain, which will therefore be an exponential antichain. We do this by a Ramsey-style argument[2]: always maintaining the invariant of exponential growth, at each step we pick a fixed word w of length k,

[2]So named by analogy to the proof of the infinite Ramsey theorem, that any edge colouring in n colours of an infinite complete graph has an infinite monochromatic subgraph. Pick a single vertex v_0, and by the pigeon-hole principle choose a colour $k_0 \in [n]$ such that $c(v_0, v) = k_0$ for infinitely many v. Then throw away all vertices v such that $c(v_0, v) \neq k_0$, preserving the invariant that the graph is infinite. Repeat this process to obtain an infinite sequence of vertices $v_0, v_1, v_2, \ldots$ and colours $k_0, k_1, k_2, \ldots$ such that for any $i < j$ we have $c(v_i, v_j) = k_i$. We must have some k such that $k_i = k$ for infinitely many i, say on the infinite set $I \subseteq \mathbb{N}$. Then we have that v_I is an infinite monochromatic subgraph.

throw away that word if it is in the set, and also throw away all longer words of which w is *not* a prefix. We will see that by Lemma 6.11 it is always possible to choose w such that this process preserves the invariant.

Lemma 6.12. *Let L be an exponential quasiantichain. Then there exists an exponential antichain $L' \subseteq L$.*

Proof. Suppose that L has exponential growth, that is that $|L|_{=n} > 2^{\epsilon n}$ infinitely often for some ϵ. We will construct a prefix-free set $S \subset \Sigma^*$ such that $S \cap L$ has exponential growth. We will construct a sequence of sets $S_0 \supseteq S_1 \supseteq S_2 \supseteq \ldots$ (and associated integers $n_0 < n_1 < n_2 < \ldots$ and reals $\epsilon_0 > \epsilon_1 > \epsilon_2 > \ldots > \epsilon'$ for initially chosen $0 < \epsilon' < \epsilon$) such that the intersection of the S_i is the desired set S. In particular we will maintain the invariant that each $S_i \cap L$ has $|S_i \cap L|_{=n} > 2^{\epsilon_i n}$ infinitely often.

Let $S_0 = \Sigma^*$ and let $n_0 = 0$. To produce S_{i+1}, note that by the invariant we can choose some $n = n_{i+1} > n_i$ such that $|S_i \cap L|_{=n} > 2^{\epsilon_i n}$. Now $S_i \cap L$ has exponential growth of order ϵ_i, hence so does $(S_i \cap L)_{>n}$. Now

$$(S_i \cap L)_{>n} = \bigcup_{w \in \Sigma^n} (S_i \cap L) \cap w\Sigma^+,$$

which is a finite union. Hence by Lemma 6.11 we have that $(S_i \cap L) \cap w\Sigma^+$ has exponential growth of order ϵ_i for some $w = w_{i+1} \in \Sigma^n$. Thus taking any ϵ_{i+1} with $\epsilon' < \epsilon_{i+1} < \epsilon_i$ we have that $|(S_i \cap L) \cap w_{i+1}\Sigma^+|_{=n} > 2^{\epsilon_{i+1} n}$ infinitely often. Now let

$$S_{i+1} = S_i \cap \left(\Sigma^{\leq n_i} \cup (\Sigma^n \setminus w_{i+1}) \cup w_{i+1}\Sigma^+\right).$$

Informally, to form S_{i+1} we leave intact the part of S_i consisting of words of length n_i or shorter. To this we add all the words of length n in S_i apart from w_{i+1}, and all the words of length $> n$ which have w_{i+1} as a prefix. Since $S_i \cap w_{i+1}\Sigma^+ \subseteq S_{i+1}$ we clearly preserve the exponential growth invariant.

We must now show that S is prefix free and that it has exponential intersection with L. Note that the set of word lengths in S is $\{n_0, n_1, n_2, \ldots\}$, and also that

$$S_{=n_i} = (S_i)_{=n_i}.$$

So

$$\begin{aligned} |S \cap L|_{=n_i} &= |S_i \cap L|_{=n_i} \\ &\geq |S_{i-1} \cap L|_{=n_i} - 1 \\ &> 2^{\epsilon_{i-1} n} - 1 \\ &> 2^{\epsilon' n} - 1, \end{aligned}$$

where the first inequality is by the construction of S_i from S_{i-1} (up to a single word of length n_i is removed, namely w_i), the second is by the definition of n_i and the third is by the definition of ϵ_{i-1}. Hence $S \cap L$ has exponential growth of order at least ϵ'.

To show that S is prefix free, we show that S_i has no pair $w < w'$ such that $|w| = n_i$. Indeed, by the definition of S_i we must have on the one hand that $w \neq w_i$ but on the other that $w' \in w_i\Sigma^+$, and so $w \not\leq w'$. Since $S \subseteq S_i$ for all i and S only contains words of length n_i for some i, we have that S is prefix-free. □

6.3 Regular languages

The dichotomy between polynomial and exponential language growth for regular languages has been independently discovered at least six times (see citations in [26]), in each case based on the fact that a regular language L has polynomial growth if and only if L is *bounded* (that is, $L \subseteq w_1^* \dots w_k^*$ for some $w_1, \dots, w_k$); otherwise L has exponential growth.

In [26], Gawrychowski, Krieger, Rampersad and Shallit describe a polynomial time algorithm for determining whether a language is bounded. The key idea is to consider the sets L_q of words which can be generated beginning and ending at state q. L is bounded if and only if for every q we have that L_q is *commutative* (that is, that $L_q \subseteq w^*$ for some w), and this can be checked in polynomial time.

In this section, we generalise this idea to the problem of antichain growth by showing that L has polynomial antichain growth if and only if L_q is a chain for every q, and otherwise L has exponential antichain growth. This is sufficient to establish the dichotomy theorem (Theorem 6.16). To give an algorithm for distinguishing the two cases (Theorem 6.18), we show how to produce an automaton whose language is empty if and only if L_q is a chain (roughly speaking the automaton accepts pairs of incomparable words in L_q).

Before proving the main theorems, we will first show (Lemma 6.13) that if L_1 and L_2 have polynomial antichain growth then so does L_1L_2. Moreover if the rates of polynomial growth of L_1 and L_2 are at most k_1 and k_2 respectively then the rate of polynomial growth of L_1L_2 is at most k_1k_2.

Lemma 6.13. *Let L_1, L_2 be languages with polynomial antichain growth of order at most k_1 and k_2 respectively. Then L_1L_2 has polynomial antichain growth of order at most $k_1 + k_2 + 1$.*

Proof. Let C_1, C_2 be such that for any antichain family $L \subseteq L_i$ we have $|L|_n < C_i n^{k_i}$ for all n. We have

$$(L_1 L_2)_{=n} = \bigcup_{i=0}^{n} (L_1)_{=i} (L_2)_{=n-i},$$

and so it suffices to prove that each $(L_1)_{=i} (L_2)_{=n-i}$ contains antichains of size at most proportional to $n^{k_1+k_2}$.

Let $L \subseteq (L_1)_{=i} (L_2)_{=n-i}$ be an antichain. Then by the concatenation lemma we have that $\{w \in (L_1)_{=i} | ww' \in L \text{ for some } w'\}$ is an antichain, and hence it has size at most $C_1 i^{k_1}$. On the other hand, by the prefixing lemma we have that the set $\left\{w' \in (L_2)_{=n-i} \middle| ww' \in L\right\}$ is an antichain, and hence it has size at most $C_2 n^{k_2}$. Since

$$L = \bigcup_{w \in (L_1)_{=i}} \left\{ww' \middle| w' \in (L_2)_{=n-i}, ww' \in L\right\},$$

we have that

$$\begin{aligned} |L| &\leq |\{w \in (L_1)_{=i} | ww' \in L \text{ for some } w'\}| \times \max_w \left|\left\{w' \in (L_2)_{=n-i} \middle| ww' \in L\right\}\right| \\ &\leq C_1 n^{k_1} C_2 n^{k_2} \\ &= C_1 C_2 n^{k_1+k_2}, \end{aligned}$$

as required. □

We are now ready to prove the main theorem, generalising the condition for polynomial language growth (that L_q is commutative for every q) to one for polynomial antichain growth: that L_q is a chain for every relevant q.

Definition 6.14. A state q of an automaton $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$ is *accessible* if q is reachable from q_0 and *co-accessible* if F is reachable from q.

Definition 6.15. Let $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$ be an NFA. Then for each $q_1, q_2 \in Q$, the automaton $\mathcal{A}_{q_1,q_2} \triangleq (Q, \Sigma, \Delta, q_1, \{q_2\})$.

Theorem 6.16. *Let $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$ be an NFA over a partially ordered alphabet. Then*

(i) $\mathcal{L}(\mathcal{A})$ has polynomial antichain growth if and only if $\mathcal{L}(\mathcal{A}_{q,q})$ is a chain for every accessible and co-accessible state q, and

(ii) if $\mathcal{L}(\mathcal{A})$ does not have polynomial antichain growth then it contains an exponential antichain (and hence has exponential antichain growth).

Proof. Suppose that $w_1, w_2 \in \mathcal{L}(\mathcal{A}_{q,q})$ with $w_1 \not\sim w_2$ and q accessible and co-accessible, so $w \in \mathcal{L}(\mathcal{A}_{q_0,q})$ and $w' \in \mathcal{L}(\mathcal{A}_{q,q'})$ for some w, w' and some $q' \in F$. Now by the Kleene star Lemma we have that $(w_1+w_2)^*$ is an exponential quasiantichain and so by Lemma 6.12 there is an exponential antichain $L' \subseteq (w_1 + w_2)^*$. Then by the Prefixing and Postfixing Lemmas we have that $wL'w' \subseteq L$ is an exponential antichain.

For the converse, we proceed by induction on $|Q|$. Let $Q' = Q \setminus \{q_0\}, F' = F \setminus \{q_0\}$ and $\Delta'(q, a) = \Delta(q, a) \setminus \{q_0\}$ for all $q \in Q', a \in \Sigma$. For any $q \in Q'$, let $\mathcal{A}'_q = (Q', \Sigma, \Delta', q, F')$. Then by the inductive hypothesis we have that $\mathcal{L}(\mathcal{A}'_q)$ has polynomial antichain growth. Also, since $\mathcal{L}(\mathcal{A}_{q_0,q_0})$ is a chain it has polynomial (in particular constant) antichain 'growth'. Now we have

$$\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}_{q_0,q_0}) \cup \bigcup_{q \in Q'} \bigcup_{a \in \Delta(q_0,q)} L_{q_0} a \mathcal{L}(\mathcal{A}'_q).$$

By Lemma 6.13, each $L_{q_0} a \mathcal{L}(\mathcal{A}'_q)$ also has polynomial antichain growth, and hence by Lemma 6.11 so does the finite union. □

A trivial restatement of part (ii) of the theorem shows that the two possible definitions of antichain growth are equivalent

Corollary 6.17. *Let L be a regular language. Then L has exponential (respectively super-polynomial) antichain growth if and only if L contains an exponential (respectively super-polynomial) antichain.*

Using Theorem 6.16 we can produce an algorithm for distinguishing the two cases.

Theorem 6.18. *There exists a polynomial time algorithm to determine whether the language of a given NFA $\mathcal{A}$ has exponential antichain growth.*

Proof. First remove all states which are not accessible and co-accessible (trivial flood fill: for instance, to compute the set of accessible states, initialise the set $X = \{q_0\}$ and then repeatedly add states to X if they can be reached by a transition from a state in X), to give $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$. We will now check for each state q whether $\mathcal{L}(\mathcal{A}_{q,q})$ is a chain.

Let Σ' denote the alphabet $\{x' | x \in \Sigma\}$ (that is, an alphabet of fresh letters of the same size as Σ). Let $\mathcal{A}'$ be the automaton corresponding to $\mathcal{A}$ over Σ'. Let $\mathcal{B} = (\Sigma \cup \{s_0, s_1\}, \Sigma \cup \Sigma', \widetilde{\Delta}, s_0, \{s_1\})$ be an NFA, where s_0, s_1 are fresh and $\widetilde{\Delta}$ is given

by (for all $a \in \Sigma$)

$$\begin{aligned}
\widetilde{\Delta}(s_0, a) &= \{a\},\\
\widetilde{\Delta}(a, a') &= \{s_0\},\\
\widetilde{\Delta}(a, b') &= \{s_1\} \text{ for all } b \text{ with } a \not\leq b \text{ and } b \not\leq a,\\
\widetilde{\Delta}(s_1, a) &= \widetilde{\Delta}(s_1, a') = \{s_1\},
\end{aligned}$$

and all other sets empty.

Then $\mathcal{B}$ has two important properties. Firstly every word accepted by $\mathcal{B}$ is a shuffle of two words w_1 and w_2', where $w_1, w_2 \in \Sigma^*$ such that $w_1 \not\sim w_2$ and w_2' is w_2 over the primed alphabet (intuitively, the two words are equal for the part where s_0 is visited, and then they first differ by two incomparable letters). Secondly, for every $w_1 \not\sim w_2$ we have that the perfect shuffle of w_1 and w_2 is accepted by $\mathcal{B}$ (that is, if $w_1 = a_1 a_2 \dots a_k, w_2 = b_1 b_2 \dots b_{k'}$ and WLOG $k < k'$ then $a_1 b_1' a_2 b_2' \dots a_k b_k' b_{k+1}' \dots b_{k'}'$ is accepted by $\mathcal{B}$).

Hence $\mathcal{A}_{q,q}$ is a chain if and only if $(\mathcal{A}_{q,q} \,|||\, \mathcal{A}'_{q,q}) \cap \mathcal{B}$ is empty, which can be checked in polynomial time (where $|||$ is the interleaving operator, which can be realised by a product construction). □

Note that in fact it suffices to check a single representative of each strongly connected component of $\mathcal{A}$ since if q is reachable from q' then it is clear that $\mathcal{L}(\mathcal{A}_{q,q})$ is a chain if and only if $\mathcal{L}(\mathcal{A}_{q',q'})$ is. Indeed, suppose that $w_1 \not\sim w_2 \in \mathcal{L}(\mathcal{A}_{q,q})$, $w \in \mathcal{L}(\mathcal{A}_{q,q'})$ and $w' \in \mathcal{L}(\mathcal{A}_{q',q})$. Then $w'w_1w \not\sim w'w_2w \in \mathcal{L}(\mathcal{A}_{q',q'})$. Hence if $\mathcal{L}(\mathcal{A}_{q',q'})$ is a chain then so is $\mathcal{L}(\mathcal{A}_{q,q})$, and similarly vice versa.

6.4 Precise growth rates

Theorem 6.18 shows that we can tell whether the language of a given NFA has polynomial or exponential antichain growth, but what if we want more precise information than this? For instance, we might ask about the order of polynomial or exponential growth. Alternatively, we might ask for given n what is the largest antichain consisting of words of length n.

If the language is specified as an NFA there appear to be formidable difficulties with all of these questions, since even the special case of the discrete order (that is, the simpler question of language growth) is highly non-trivial. In the first place, it is well-known that the question of determining whether a given NFA over the alphabet $\Sigma = \{0, 1\}$ generates the language Σ^*, and hence whether we have $|L|_k = 2^k$, is

PSPACE-complete. Therefore unless P=PSPACE there cannot be an algorithm to efficiently compute the precise language (or, *a fortiori*, antichain) growth behaviour of a given NFA.

Even the easier problem of computing the order of exponential growth appears to be unsolved for NFA.[3] The order of polynomial growth is solved for NFA in [26] (although not in the original preprint version [39]). This is possible essentially because having polynomial growth is a very constraining property: in particular for each state we have that any path from that state to itself must generate a word in w^* for some single fixed w. This allows us to keep track of the ambiguity that appears (i.e. the extent to which different paths can generate the same word) in a relatively simple fashion. Unfortunately the property of having polynomial antichain growth does not have such useful consequences: in particular, even though $L_{q,q}$ is a chain for each q, this can still be extremely complex.

What if we consider the language to be given as a DFA? (Note that in the information flow setting the overhead involved in determising the relevant automaton corresponds to the amount of 'hidden state' in the system invisible to Bob.) For fixed n given in unary, there is a straightforward dynamic programming algorithm to compute the size of the larges antichain consisting of words of length n.

For any $q \in Q$ and $k \in \mathbb{N}$, let $f(q,k)$ be the size of the largest antichain consisting of words of length k that can be accepted starting at state q (and let $f(q,0) = 1$ if $q \in F$ and 0 otherwise). We have the recurrence relation

$$f(q,k) = \max_{X \in \mathcal{F}} \sum_{x \in X} f(\delta(q,x), k-1),$$

where $\mathcal{F} \subseteq \mathcal{P}(\Sigma)$ is the set of nonempty antichains in Σ. We can use this to compute $f(q_0, n)$ in polynomial time, which is the size of the largest antichain consisting of words of length n. If we wish to calculate the largest antichain consisting of words of length *at most* n, then the recurrence relation becomes

$$f(q,k) = \max\left(1, \max_{X \in \mathcal{F}} \sum_{x \in X} f(\delta(q,x), k-1)\right).$$

What about computing the rate of exponential or polynomial antichain growth for DFA? Unfortunately, for exponential growth, the only methods of which we are aware for computing the language growth rate of DFA are based on the Perron-Fobenius theorem—the growth rate is r^n, where r is the Perron-Frobenius eigenvalue of the

[3] See [35], in which we posed this question on MathOverflow and received no answer; furthermore no solution is claimed or referred to in [26].

transition matrix (the Perron-Frobenius eigenvalue is the eigenvalue of maximum modulus, which is non-negative real for a square non-negative matrix). It is not clear how to apply these essentially algebraic methods to the antichain growth problem.

On the other hand, for polynomial growth there is a polynomial time algorithm to compute the order of polynomial growth where the language is specified as a DFA, as we now proceed to show. We will then show that in fact this algorithm also works for NFA. We will assume throughout without loss of generality that all states are accessible and co-accessible.

Definition 6.19. Let $\mathcal{A} = (Q, q_0, F, \Sigma, \delta)$ be a DFA over a partially ordered alphabet. Let $G_{\mathcal{A}} = (Q, E)$ be the directed graph with vertex-set Q such that $(q, q') \in E$ if and only if $q \xrightarrow{w} q'$ for some $w \in \Sigma^*$.

Let $G'_{\mathcal{A}} = (Q, E')$ be the directed graph with $(q, q') \in E'$ if and only if there exist words $w \not\sim w' \in \Sigma^*$ such that $q \xrightarrow{w} q$ and $q \xrightarrow{w'} q'$.

We will generally omit the subscript $\mathcal{A}$s from now on, where this will not cause confusion.

Note that by Theorem 6.16, we have that G' is a directed acyclic graph (DAG) if and only if $\mathcal{L}(\mathcal{A})$ has polynomial antichain growth.

Note also that by a similar argument to the proof of Theorem 6.18, the graph G' can be computed in polynomial time. Clearly G can be computed in polynomial time using a flood fill.

Definition 6.20. Let $\mathcal{A} = (Q, q_0, F, \Sigma, \delta)$ be a DFA with polynomial antichain growth. For a directed path $P = q_0 q_1 \dots q_l$ (not necessarily simple) in $G_{\mathcal{A}}$, let

$$D(P) = |\{i \in \{0, \dots, l-1\} | (q_i, q_{i+1}) \in E(G'_{\mathcal{A}})\}| + \begin{cases} 1 \text{ if } |L_{q_m, q_l}| = \infty \\ 0 \text{ otherwise.} \end{cases},$$

where $m = \max\{i+1 | (q_i, q_{i+1}) \in G'_{\mathcal{A}}\}$ if this exists, and 0 otherwise.

Observe that if $|L_{q_m, q_l}| = \infty$ then we have $ww'^*w'' \subseteq L_{q_m, q_l}$ for some w, w', w''.

Lemma 6.21. *Let $\mathcal{A} = (Q, q_0, F, \Sigma, \delta)$ be a DFA with polynomial antichain growth. Let $\mathcal{P}$ be the set of directed paths from q_0 to an element of F. The quantity*

$$D_{\mathcal{A}} = \max_{P \in \mathcal{P}} D(P)$$

is well-defined and can be computed in polynomial time.

Proof. To show that $D_{\mathcal{A}}$ is well-defined, observe that no directed cycle in G contains an edge in G'. Indeed, suppose that $q_1 q_2 \dots q_1$ is a directed cycle in G, with $(q_1, q_2) \in E(G')$. Then we have $q_1 \xrightarrow{w} q_1$ and $q_1 \xrightarrow{w'} q_2$ for some $w \not\sim w' \in \Sigma^*$. Also we have $q_2 \xrightarrow{w''} q_1$ for some $w'' \in \Sigma^*$. But then $q_1 \xrightarrow{w'w''} q_1$ and $w'w'' \not\sim w$ by the Postfixing Lemma, contradicting polynomial antichain growth of $\mathcal{L}(\mathcal{A})$. Hence $D(P)$ is bounded.

For a polynomial time algorithm, first expand G and G' by adding a sink vertex v_f for each $f \in F$. For each q such that $|L_{q,f}| = \infty$ put $(q, f) \in E(G)$ and $(q, f) \in E(G')$. Then add a further vertex v with $(f, v) \in E(G)$ and $(v_f, v) \in E(G)$ for all $f \in F$. Then $D_{\mathcal{A}}$ is precisely the maximum number of edges of G' contained in a directed path from q_0 to v in G.

Form the graph G'' on vertex-set $Q \cup \{v\}$ by $(v_1, v_2) \in E(G'')$ if and only if there is a path from v_1 to v_2 in G containing a single edge of G'. Then we have that G'' is a DAG (by the first observation), and $D_{\mathcal{A}}$ is the longest path from q_0 to v in G'', which can be found by a simple dynamic programming algorithm. □

Lemma 6.22. *Let $\mathcal{A} = (Q, q_0, F, \Sigma, \delta)$ be a DFA with polynomial antichain growth. Then $\mathcal{L}(\mathcal{A})$ has polynomial antichain growth of order at least $D_{\mathcal{A}} - 1$.*

Proof. Let $P = q_0 q_1 \dots q_l$ be a path with $D(P) = D_{\mathcal{A}}$. Let $i_1, \dots, i_k$ be such that $(q_{i_j}, q_{i_j+1}) \in E(G'_{\mathcal{A}})$ for all j. Let $w_1, \dots, w_k, w'_1 \dots, w'_k, w \in \Sigma^*$ be such that $w_j \not\sim w'_j$ for all j, $q_{i_j} \xrightarrow{w_j} q_{i_j}$ for all j, $q_{i_j} \xrightarrow{w'_j} q_{i_{j+1}}$ for all $j < k$, $q_{i_k} \xrightarrow{w'_k} q_l$, and $q_0 \xrightarrow{w} q_{i_1}$.

Suppose that $|L_{q_m,q_l}| = \infty$ (with m defined as in Definition 6.20), and let $w', w'', w''' \in \Sigma^*$ be such that $w'w''^*w''' \subseteq L_{q_m,q_l}$. Then $L = ww_1^*w'_1w_2^*w'_2 \dots w_k^*w'_kw'w''^*w'''$ is an antichain with polynomial growth of order $k = D_{\mathcal{A}} - 1$. Similarly if $L_{q_m,q_l} < \infty$, then $L = ww_1^*w'_1w_2^*w'_2 \dots w_k^*w'_k$ is an antichain with polynomial growth of order $k - 1 = D_{\mathcal{A}} - 1$. □

We will now prove the upper bound. Our strategy will be to classify words by the edges of G' they visit. We first show a preliminary lemma, which bounds the antichain growth from regions between edges of G'.

Lemma 6.23. *Let $q_1, q_2 \in Q$, and let $L \subseteq L_{q_1,q_2}$ be the set of words such that no edges of G' appear in the runs corresponding to elements of L. Then L has antichain growth of order at most 0.*

Proof. Without loss of generality we may assume that $\mathcal{A}$ does not have any transitions labelled by more than a single letter (by introducing additional states if necessary; in particular we can set $Q' = Q \times \Sigma$ and ensure that $\delta'(q, x) \in Q \times \{x\}$ for all $x \in \Sigma$).

We will show that L cannot contain two incomparable words that correspond after removal of loops to the same sets of simple paths in G.[4] Since G is finite and hence contains only finitely many simple paths, this suffices to establish the result.

Suppose that $w_1 \not\sim w_2$ correspond to the same simple path P. Suppose that the first point of divergence of w_1 and w_2 is at state q; that is, that $w_1 = wx_1w_1'$ and $w_2 = wx_2w_2'$ with $x_1 \neq x_2 \in \Sigma$ and $q_1 \xrightarrow{w} q$ (see Figure 6.1). Without loss of generality we may assume that q and $\delta(q, x_1)$ lie on P.

Since the path for w_2 corresponds to P after removal of cycles, we must have that $w_2' = w_2''w_2'''$ with $q \xrightarrow{x_2w_2''} q$ and $q \xrightarrow{w_2'''} q_2$. But $w_1 \not\sim w_2$ and $x_1 \neq x_2$ so $x_1 \not\sim x_2$ and so $x_1 \not\sim x_2w_2''$. Hence $(q, \delta(q, x_1) \in G'$, which is a contradiction. □

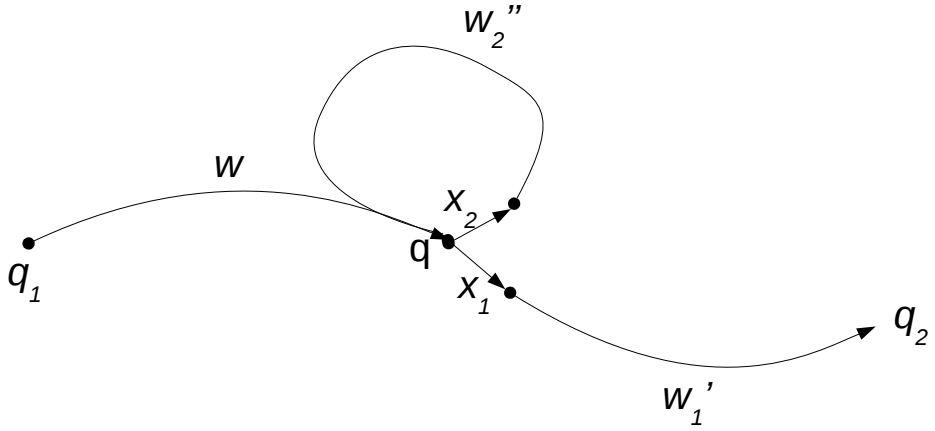

Figure 6.1: The proof of Lemma 6.23

Lemma 6.24. *Let $\mathcal{A} = (Q, q_0, F, \Sigma, \delta)$ be a DFA with polynomial antichain growth. Then $\mathcal{L}(\mathcal{A})$ has polynomial antichain growth of order at most $D_{\mathcal{A}} - 1$.*

Proof. We may assume without loss of generality that there is only a single accepting state, say q_f.

We classify words by the edges of G' that appear in their accepting runs. We shall show that the set words corresponding to a fixed sequence P of G'-edges has antichain growth of order at most $D(P)$ (where $D(P) = |P| - 1$ or $|P|$ depending on whether the set of accepted words beginning at the last vertex of P is finite). Since the number of relevant G'-edge sequences is finite (recalling that no edge of G' is contained in a directed cycle in G and so no G'-edge can appear more than once), this will suffice to establish the result.

[4]Note that since removal of loops may be done in many different ways, a single path may correspond to multiple simple paths. We are asserting that L cannot contain two incomparable words which correspond to precisely the same *sets* of simple paths.

Let $(q_1, q_1'), \ldots, (q_k, q_k')$ be a set of G'-edges. Then the set L of words which have this sequence of G'-edges in their run is given by

$$L = L'_{q_0,q_1} X_1 L'_{q_1',q_2} X_2 L'_{q_2',q_3} \ldots X_k L'_{q_k',q_f},$$

where $X_i = \{x \in \Sigma \mid \delta(q_i, x) = q_i'\}$ and $L'_{q,q'} \subset L_{q,q'}$ is the set of words whose runs do not include edges of G'.

The X_i are finite and hence have antichain growth of order -1. By Lemma 6.23 the $L'_{q_i',q_{i+1}}$ and also L'_{q_0,q_1} and L'_{q_k',q_f} have antichain growth of order at most 0. Moreover if L_{q_k',q_f} is finite then so is $L'_{q_k',q_f} \subseteq L_{q_k',q_f}$ and so it has antichain growth of order -1. The result follows by Lemma 6.13. □

Combining Lemmas 6.21, 6.22 and 6.24 yields

Theorem 6.25. *Let $\mathcal{A} = (Q, q_0, F, \Sigma, \delta)$ be a DFA with polynomial antichain growth. Then $\mathcal{L}(\mathcal{A})$ has polynomial antichain growth of order exactly $D_\mathcal{A} - 1$, which can be computed in polynomial time.*

We now show how to extend this algorithm to the case of NFA. Note that $D_\mathcal{A}$ as defined above is well-defined for NFA just as for DFA, and that the algorithm to compute it in polynomial time is equally applicable. It therefore remains to show that for NFA we also have that if $\mathcal{A}$ has polynomial antichain growth then it has antichain growth of order exactly $D_\mathcal{A} - 1$.

We do this by showing (Lemma 6.27) that $D_\mathcal{A}$ depends only on the language $\mathcal{L}(\mathcal{A})$, so that if $\mathcal{A}$ and $\mathcal{A}'$ are NFA with $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$ then $D_\mathcal{A} = D_{\mathcal{A}'}$. Have shown this we then consider $\mathcal{A}'$ to be the determinisation of $\mathcal{A}$. This is a DFA with $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$, and by Theorem 6.25 we have that $\mathcal{L}(\mathcal{A}')$ has polynomial antichain growth of order $D_{\mathcal{A}'} - 1 = D_\mathcal{A} - 1$.

We will first show (Lemma 6.26) that if $L = v_0 w_1^* v_1 w_2^* v_2 \ldots w_k^* v_k \subseteq \mathcal{L}(\mathcal{A})$ then there exists a single sequence of states $q_1, q_2, \ldots, q_k$ which essentially realises L (that is, up to various cyclic shifts and offsets we have $v_i \in \mathcal{L}(\mathcal{A}_{q_i,q_{i+1}})$ and $w_i^* \in \mathcal{L}(\mathcal{A}_{q_i,q_i})$).

Lemma 6.26. *Let $\mathcal{A} = (Q, q_0, F, \Sigma, \Delta)$ be an NFA such that $v_0 w_1^* v_1 w_2^* v_2 \ldots w_k^* v_k \subseteq \mathcal{L}(\mathcal{A})$. Then then there exists a sequence of states $q_1, q_2, \ldots, q_{k+1}$ and integers $m_1, m_2, \ldots m_k$, $m_1', m_2', \ldots, m_k'$ and $n_1, n_2, \ldots, n_k$ such that*

(i) $v_0 w_1^{m_1} \in \mathcal{A}_{q_0,q_1}$ and $w_k^{m_k'} v_k \in \mathcal{L}(\mathcal{A}_{q_k,F})$,

(ii) for all $0 < i < k$ we have $w_i^{m_i'} v_i w_{i+1}^{m_{i+1}} \in \mathcal{L}(\mathcal{A}_{q_i,q_{i+1}})$, and

(iii) for all $0 < i \leq k$ *we have* $w_i^{n_i} \in \mathcal{L}(\mathcal{A}_{q_i,q_i})$.

Proof. Consider an accepting run for $v_0 w_1^{|Q|+1} v_1 w_2^{|Q|+1} v_2 \dots w_k^{|Q|+1} v_k \in \mathcal{L}(\mathcal{A})$, and write $q(s)$ for the state reached in this run after the word s. By the pigeon-hole principle, we must have $q(v_0 w_1^{m_1}) = q(v_0 w^{m_1+n_1} = q_1$ (say) for some $m_1 \geq 0$ and some $n_1 > 0$ with $m_1 + n_1 \leq |Q| + 1$. Let $m_1' = |Q| + 1 - m_1 - n_1$. Similarly for each i we have $q(v_1 w_1^{|Q|+1} v_2 \dots w_i^{m_i}) = q(v_1 w_1^{|Q|+1} v_2 \dots w_i^{m_i+n_i}) = q_i$ (say) for some $m_i \geq 0$ and $n_i > 0$ with $m_i + n_i \leq |Q| + 1$. Let $m_i' = |Q| + 1 - m_i - n_i$. Then these q_i, m_i, m_i' and n_i give the result. □

Lemma 6.27. *Let* $\mathcal{A}$ *and* $\mathcal{A}'$ *be NFA with* $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$. *Then* $D_{\mathcal{A}} = D_{\mathcal{A}'}$.

Proof. Let $\mathcal{A} = (Q, q_0, F, \Sigma, \Delta)$ and $\mathcal{A}' = (Q', q_0', F', \Sigma, \Delta')$.

Suppose that $D_{\mathcal{A}'} = k$. Then by an identical argument to the proof of Lemma 6.22 we have that $v_0 w_1^* v_1 w_2^* v_2 \dots w_k^* v_k \subseteq \mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$ for some $v_0, \dots, v_k, w_1, \dots, w_k \in \Sigma^*$ with $w_i \not\sim v_i$. Then by Lemma 6.26 there exists a sequence of states $q_1, q_2, \dots, q_{k+1} \in Q$ and integers $m_1, m_2, \dots, m_k, m_1', m_2', \dots m_k'$ and $n_1, n_2, \dots, n_k$ such that (i)–(iii) in the statement of the lemma hold. Now since $w_i \not\sim v_i$ we have $w_i^{n_i} \not\sim w_i^{m_i'} v_i w_{i+1}^{m_{i+1}}$ and so

$$D_{\mathcal{A}} \geq k = D_{\mathcal{A}'}.$$

Similarly $D_{\mathcal{A}'} \geq D_{\mathcal{A}}$, and hence $D_{\mathcal{A}} = D_{\mathcal{A}'}$. □

Theorem 6.28. *Let* $\mathcal{A}$ *be an NFA with polynomial antichain growth. Then* $\mathcal{L}(\mathcal{A})$ *has polynomial antichain growth of order exactly* $D_{\mathcal{A}} - 1$.

Proof. Let $\mathcal{A}'$ be the powerset determinisation of $\mathcal{A}$, so $\mathcal{A}'$ is a DFA with $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$. By Theorem 6.25, $\mathcal{L}(\mathcal{A}')$ has polynomial antichain growth of order exactly $D_{\mathcal{A}'} - 1$, and by Lemma 6.27 we have $D_{\mathcal{A}'} = D_{\mathcal{A}}$. □

6.5 Examples

In this section we illustrate the theory of the preceding three chapters by applying it to two example systems: the relay system depicted in Figure 5.3 (reproduced for convenience as Figure 6.2), and an 'interrupt' system that allows Alice to interrupt Bob's actions for a single contiguous period.

6.5.1 Relay system

Recall that the relay system allows Alice and Bob to input a letter from $\{a, b\}$ at each step, and outputs to each the letter selected by the other. The transducer corresponding to this system is shown in Figure 5.3. We reproduce this below as Figure 6.2, but since for later work we assume that Σ_B and Γ_B are disjoint, we will write Bob's inputs (and Alice's outputs) using primed symbols.

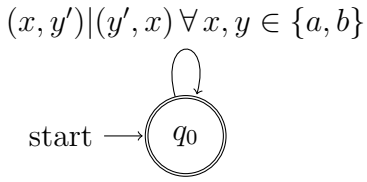

Figure 6.2: A relay system.

The automaton corresponding to this system, which we will denote by $\mathcal{A}$, was discussed in Chapter 5, and shown in Figure 5.4, reproduced for convenience as Figure 6.3 (again with Σ_A and Γ_B written using primed symbols).

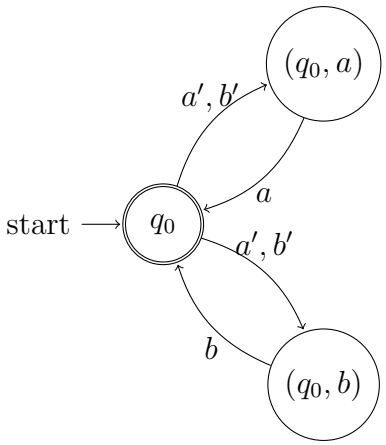

Figure 6.3: Automaton corresponding to the relay system transducer shown in Figure 6.2.

Now the question is whether $\mathcal{L}(\mathcal{A})$ has polynomial or exponential antichain growth with respect to the lexicographic order, where $\Sigma_B \cup \Gamma_B = \{a', b', a, b\}$ is ordered by $a' < b'$, and other elements incomparable.

Now q_0 is accessible and coaccessible, and we have $a'a \not\sim a'b \in \mathcal{L}(\mathcal{A}_{q,q})$, and hence by Theorem 6.16 we have that $\mathcal{L}(\mathcal{A})$ has exponential antichain growth and so the system has linear information flow.

6.5.2 Interrupt system

This system is intended to represent a resource shared between Alice and Bob. At each step Bob can transmit a', signifying that he wishes to use the resource, or b', signifying that he does not. If he asks to use the resource then he will receive either an a, signifying that he was successful, or a b, signifying that he was not. If he did not try to use the system then he always receives a b. Similarly Alice can transmit a if she wishes to use the system or b if she does not, and receives acknowledgements a' and b' for success and failure respectively.

Initially, Bob has priority over the use of the system, and for as long as Alice transmits b he retains it. However, as soon as Alice seeks to use the system by transmitting an a she obtains priority and retains it for as long as she uses it continuously. As soon as she transmits a b priority shifts back to Bob, who retains it for the remainder of the execution.

This system is depicted in Figure 6.4 (where missing arguments mean that the input from that user is ignored).

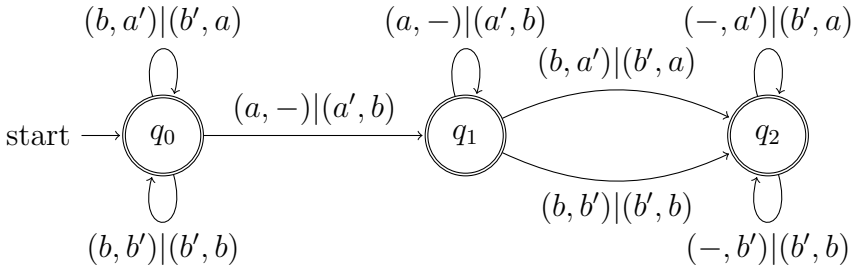

Figure 6.4: An interrupt system.

We can now apply Definition 5.12 to construct the corresponding automaton $\mathcal{A}$. This is shown in Figure 6.5.

As before our task is to determine whether $\mathcal{L}(\mathcal{A})$ has polynomial or exponential antichain growth with respect to the lexicographic order, where $\Sigma_B \cup \Gamma_B = \{a', b', a, b'\}$ is ordered by $a' < b'$ and other elements incomparable.

Now we have that $\mathcal{L}(\mathcal{A}_{q_0,q_0}) = (a'a + b'b)^*$ is a chain, and hence (by the remark following Theorem 6.18) so are $\mathcal{L}(\mathcal{A}_{(q_0,a),(q_0,a)})$ and $\mathcal{L}(\mathcal{A}_{(q_0,b),(q_0,b)})$. Similarly $\mathcal{L}(\mathcal{A}_{q_1,q_1}) = (a'b + b'b)^*$ is a chain and hence so is $\mathcal{L}(\mathcal{A}_{(q_1,b),(q_1,b)})$. Finally $\mathcal{L}(\mathcal{A}_{q_2,q_2}) = (a'a + b'b)^*$ is also a chain and hence so are $\mathcal{L}(\mathcal{A}_{(q_2,a),(q_2,a)})$ and $\mathcal{L}(\mathcal{A}_{(q_2,b),(q_2,b)})$.

Hence by Theorem 6.16, we have that this $\mathcal{L}(\mathcal{A})$ has polynomial antichain growth and so the system has logarithmic information flow.

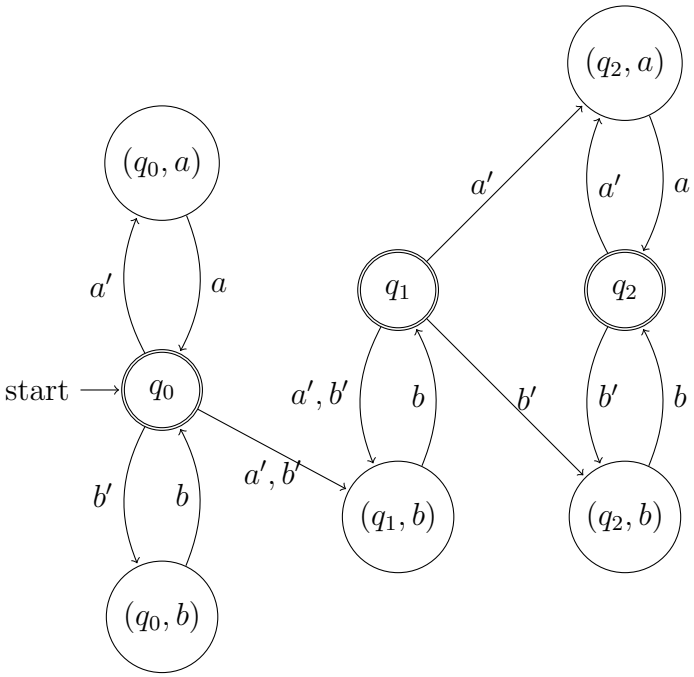

Figure 6.5: Automaton corresponding to the interrupt system transducer shown in Figure 6.4.

What about the order of polynomial growth? We have that $(q_0, q_1) \in E(G'_{\mathcal{A}})$, since $a'a \in \mathcal{L}(\mathcal{A}_{q_0,q_0})$ and $a'b \in \mathcal{L}(\mathcal{A}_{q_0,q_1})$. Similarly $(q_1, q_2) \in E(G'_{\mathcal{A}})$ since $a'b \in \mathcal{L}(\mathcal{A}_{q_1,q_1})$ and $a'a \in \mathcal{L}(\mathcal{A}_{q_1,q_2})$. Also $|\mathcal{L}(\mathcal{A}_{q_1,q_2})| = \infty$, so $D_{\mathcal{A}} \geq 3$.

On the other hand since $\mathcal{L}(\mathcal{A})$ has polynomial antichain growth we have that $G'_{\mathcal{A}}$ does not contain any edges within strongly connected components of $G_{\mathcal{A}}$ and so $D_{\mathcal{A}} \leq 3$. Hence $D_{\mathcal{A}} = 3$ and so by Theorem 6.28 it has polynomial growth of order 2. This means that in time n Bob learns approximately $2 \log n$ bits of information. Note that this makes intuitive sense: Alice can choose when to start using the resource and when to stop, which she can do in $\binom{n}{2} = \Theta(n^2)$ ways.

We conclude this section by considering the information flow from Bob to Alice for this system. Again we can apply Definition 5.12 (*mutatis mutandis*) to compute the automaton corresponding to Alice's view of the system. This is shown in Figure 6.6.

The partial order on $\Sigma_A \cup \Gamma_A = \{a, b, a', b'\}$ is $a < b$ and all other elements incomparable. Now we have that $\mathcal{L}(\mathcal{A}) = (bb')^*(aa')^+(bb')^+$ is a chain, and so there is no information flow from Bob to Alice.

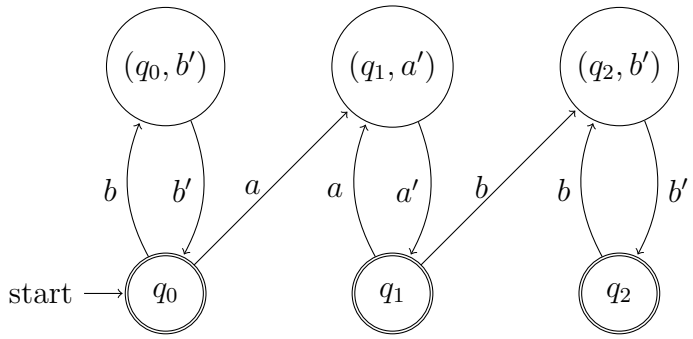

Figure 6.6: Automaton corresponding to Alice's view of the interrupt system transducer shown in Figure 6.4.

6.6 Context-free languages

In [30], Ginsburg and Spanier show (Theorem 5.1) that a context-free grammar G generates a bounded language if and only if the sets $L_A(G)$ and $R_A(G)$ are commutative for all non-terminals A, where L_A and R_A are respectively the sets of possible w and u in productions $A \stackrel{*}{\Rightarrow} wAu$. They also give an algorithm to decide this (which [26] improves to be in polynomial time).

We generalise this to our problem by showing that G generates a language with polynomial antichain growth if and only $L_A(G)$ and also the sets $R_{A,w}(G)$ of possible u for each fixed w are chains, and that otherwise $\mathcal{L}(G)$ has exponential antichain growth. However, we will show that the problem of distinguishing the two cases is undecidable, by reduction from the CFG intersection emptiness problem.

Except where otherwise specified, we will assume all CFGs have starting symbol S and that all nonterminals are accessible and co-accessible: for any nonterminal A we have $S \stackrel{*}{\Rightarrow} uAu'$ for some $u, u' \in \Sigma^*$ and $A \stackrel{*}{\Rightarrow} v$ for some $v \in \Sigma^*$.

Definition 6.29. Let G be a context-free grammar (CFG) over Σ. Then for any nonterminal A let

$$L_A(G) = \{w \in \Sigma^* | \, \exists \, u \in \Sigma^* : A \stackrel{*}{\Rightarrow} wAu\}.$$

Lemma 6.30. *Let G be a CFG over Σ and A some nonterminal such that $L_A(G)$ is not a chain. Then $\mathcal{L}(G)$ contains an exponential antichain.*

Proof. Since $L_A(G)$ is not a chain, we have w_1, w_2, u_1, u_2 with $w_1 \not\sim w_2$ such that $A \stackrel{*}{\Rightarrow} w_1 A u_1$ and $A \stackrel{*}{\Rightarrow} w_2 A u_2$. Now A is accessible and co-accessible so also $S \stackrel{*}{\Rightarrow} uAu'$ and $A \stackrel{*}{\Rightarrow} v$ for some $u, u', v \in \Sigma^*$.

Hence

$$uw_{i_1}w_{i_2}\dots w_{i_k}vu_{i_k}u_{i_{k-1}}\dots u_1u' \subseteq \mathcal{L}(G),$$

for any $i_1i_2\dots i_k \in \{1,2\}^*$. Write $\phi : (w_1 + w_2)^* \to (u_1 + u_2)^*$ for the map $w_{i_1}w_{i_2}\dots w_{i_k} \mapsto u_{i_k}u_{i_{k-1}}\dots u_{i_1}$ (with any ambiguity resolved arbitrarily).

Now $\{w_{i_1}w_{i_2}\dots w_{i_k}|i_1\dots i_k \in \{1,2\}^*\} = (w_1+w_2)^*$ is a quasiantichain by Lemma 6.6, clearly it is exponential and hence by Lemma 6.12 it contains an exponential antichain L. By the Concatenation Lemma we have that $L' = \{lv\phi(l)|l \in L\}$ is an antichain, and it is exponential because there is a bijection between L and L' such that the length of each word in L' exceeds the length of the corresponding word in L by a factor of at most $\frac{|v|+\max(|u_1|,|u_2|)}{\min(|w_1|,|w_2|)}$. By the Prefixing and Postfixing Lemmas we have that $uL'u' \subseteq \mathcal{L}(G)$ is an exponential antichain. □

Definition 6.31. Let G be a CFG over Σ. Then for any nonterminal A and any $w \in \Sigma^*$, let

$$R_{A,w}(G) = \{u \in \Sigma^*|A \overset{*}{\Rightarrow} wAu\}.$$

Lemma 6.32. *Let G be a CFG over Σ, A some nonterminal and $w \in \Sigma^*$ such that $R_{A,w}(G)$ is not a chain. Then $\mathcal{L}(G)$ has exponential antichain growth.*

Proof. We have $v, w, u, u' \in \Sigma^*$ and $u_1 \not\sim u_2 \in \Sigma^*$ such that $S \overset{*}{\Rightarrow} uAu'$, $A \overset{*}{\Rightarrow} v$, $A \overset{*}{\Rightarrow} wAu_1$ and $A \overset{*}{\Rightarrow} wAu_2$. Let

$$L_i = uw^{2i}v(u_1u_2 + u_2u_1)^iu'.$$

Then L_i is an antichain and $\bigcup_{i=1}^{\infty} L_i$ is an exponential antichain family. □

Lemma 6.33. *Let G be a CFG over Σ such that $L_A(G)$ and $R_{A,w}(G)$ are chains for all nonterminals A and all $w \in \Sigma^*$. Then $\mathcal{L}(G)$ has polynomial antichain growth.*

Proof. We proceed by induction on the number of nonterminals which appear on the right hand side of productions in G. Let A be a nonterminal, and let G' be the CFG obtained from G by deleting all productions mentioning A on the right hand side and changing the starting state to A. Let $L' = \mathcal{L}(G')$. Then by the inductive hypothesis L' has polynomial antichain growth; say any antichain family $L \subseteq L'$ has $|L|_{\leq k} < Ck^N$ for some fixed C, N. If A is not the starting state, let G'' be the CFG obtained from G by deleting all productions mentioning A, and let $L'' = \mathcal{L}(G'')$ (otherwise let $L'' = \emptyset$). By the inductive hypothesis L'' also has polynomial antichain growth. Now we have

$$\mathcal{L}(G) \subseteq L'' \cup \left(L_A(G)L' \bigcup_{w\in\Sigma^*} R_{A,w} \right).$$

By Lemma 6.11 it suffices to prove that $L_A(G)L' \bigcup_{w\in\Sigma^*} R_{A,w}$ has polynomial antichain growth.

Let $L \subseteq L_A(G)L' \bigcup_{w\in\Sigma^*} R_{A,w}$ be an antichain family. Now since $L_A(G)$ is a chain and $L_{=k}$ is an antichain, we have

$$L_{=k} \subseteq \bigcup_{i=0}^{k} w_i L' R_{A,w_i},$$

for some $w_0 < w_1 < w_2 < \ldots < w_k$ with $|w_k| = k$ (recall that $<$ is defined on Σ^* as meaning strict prefix).

Since R_{A,w_i} is a chain and $L_{=k}$ is an antichain we cannot have $w_i l u, w_i l u' \in L_{=k}$ for any $l \in L'$ and $u \neq u' \in R_{A,w_i}$. Hence for each i there exists some function ϕ and $\widetilde{L} \subseteq L'$ such that

$$L_{=k} \cap w_i L' R_{A,w_i} = \{w_i l \phi(l) | l \in \widetilde{L}\}.$$

Now since $L_{=k}$ is an antichain we have that $\widetilde{L}$ is a quasiantichain and in particular an antichain family, and since also $\widetilde{L} \subseteq L'_{\leq k}$ we have that $|\widetilde{L}| < Ck^N$. Hence

$$|L_{=k} \cap w_i L' R_{A,w_i}| \leq |\widetilde{L}| < Ck^N,$$

and so

$$|L_{=k}| < (k+1)Ck^N < Ck^{N+2}$$

for sufficiently large k. □

Combining these three lemmas gives:

Theorem 6.34. *Let L be a context-free language. Then either L has exponential antichain growth or L has polynomial antichain growth.*

We now show that the problem distinguishing the two cases is undecidable, by reduction from the CFG intersection emptiness problem. In fact, it is undecidable even to determine whether a given CFG generates a chain.

Definition 6.35. CFG-Intersection is the problem of determining whether two given CFGs have non-empty intersection. CFG-Chain is the problem of determining whether the language generated by a given CFG is a chain. CFG-ExpAntichain is the problem of determining whether the language generated by a given CFG has exponential antichain growth.

Lemma 6.36. CFG-Intersection *is undecidable.*

Proof. [29], Theorem 4.2.1. □

Lemma 6.37. *There is a polynomial time reduction from* CFG-Intersection *to* CFG-Chain.

Proof. Let G_1, G_2 be arbitrary CFGs over alphabet Σ. Let $\widetilde{\Sigma} = \Sigma \cup \{0, 1\}$, with an arbitrary linear order on Σ, and $\Sigma < 0, \Sigma < 1$ but 0 and 1 incomparable. Let $\widetilde{G}$ be a CFG such that

$$\mathcal{L}(\widetilde{G}) = (\mathcal{L}(G_1)0) \cup (\mathcal{L}(G_2)1)$$

(which can trivially be constructed with polynomial blowup). Then $\mathcal{L}(\widetilde{G})$ is a chain if and only if $G_1 \cap G_2 = \emptyset$. □

Lemma 6.38. *Let L be a prefix-free chain. Then L^* is a chain.*

Proof. Let $lw \not\sim l'w'$ be a minimum-length counterexample with $l, l' \in L$ and $w, w' \in L^*$. By minimality and the Prefixing Lemma we have that $l \neq l'$. Then by the Concatenation Lemma since L is prefix-free we have that $l \not\sim l'$, which is a contradiction. □

Lemma 6.39. *There is a polynomial time reduction from* CFG-Chain *to* CFG-ExpAntichain.

Proof. Let G be a CFG over a partially ordered alphabet Σ. Let $\widetilde{\Sigma} = \Sigma \cup \{0\}$, with $\Sigma < 0$. Let $\widetilde{G}$ be a CFG such that

$$\mathcal{L}(\widetilde{G}) = (\mathcal{L}(G)0)^*.$$

We claim that $\mathcal{L}(\widetilde{G})$ has exponential antichain growth if and only if $\mathcal{L}(G)$ is not a chain.

Indeed, suppose that $l_1 \not\sim l_2 \in \mathcal{L}(G)$. Then $l_1 0 \not\sim l_2 0$ and so by Lemmas 6.6 and 6.12 we have that $(l_1 0 + l_2 0)^* \subseteq \mathcal{L}(\widetilde{G})$ contains an exponential antichain.

Conversely, suppose that $\mathcal{L}(G)$ is a chain. Then $\mathcal{L}(G)0$ is a prefix-free chain and so by Lemma 6.38 we have that $\mathcal{L}(\widetilde{G})$ is a chain. □

Combining these lemmas gives:

Theorem 6.40. *The problems* CFG-Chain *and* CFG-ExpAntichain *are undecidable.*

6.7 Tree automata

In this section, we generalise the definition of the lexicographic ordering to tree languages, and prove a trichotomy theorem: regular tree languages have antichain growth which is either polynomial, exponential or doubly exponential.

Notation and definitions (other than for the lexicographic ordering) are taken from [17], to which the reader is referred for a more detailed treatment.

Definition 6.41. Let $\mathcal{F}$ be a finite set of function symbols of arity ≥ 0, and $\mathcal{X}$ a set of variables. Write $\mathcal{F}_p$ for the set of function symbols of arity p. Let $T(\mathcal{F}, \mathcal{X})$ be the set of terms over $\mathcal{F}$ and $\mathcal{X}$. Let $T(\mathcal{F})$ be the set of *ground terms* over $\mathcal{F}$, which is also the set of *ranked ordered trees* labelled by $\mathcal{F}$ (with rank given by arity as function symbols).

For example, the set of ordered binary trees is $T(\mathcal{F})$, where $\mathcal{F} = \{f, g, c\}$ and f has arity 2, g arity 1 and c arity 0.

Note that this generalises the definition of finite words over an alphabet Σ, by taking $\mathcal{F} = \Sigma \cup \{\epsilon\}$, giving each $a \in \Sigma$ arity one and ϵ arity zero.

A term t is *linear* if no free variable appears more than once in t. A linear term mentioning k free variables is a *k-ary context*.

Definition 6.42. Let $\mathcal{F}$ be equipped with a partial order $\leq$. Then the lexicographic partial order induced by $\leq$ on $T(\mathcal{F})$ is the relation $\preceq$ defined as follows: for any $f \in \mathcal{F}_p, f' \in \mathcal{F}_q$ and any $t_1, \ldots, t_p \in T(\mathcal{F})$ and $t'_1, \ldots, t'_q \in T(\mathcal{F})$ we have $f(t_1, \ldots, t_p) \preceq f'(t'_1, \ldots, t'_q)$ if and only if either $f < f'$ or $f = f'$ and $t_i \preceq t'_i$ for all i.

Note that this generalises Definition 6.1, by taking $\epsilon \leq a$ for all $a \in \Sigma$.

As before we will write $t \sim t'$ if $t, t' \in T(\mathcal{F})$ are related by the lexicographic order; the definitions of *chain* and *antichain* are as before. To quantify antichain growth we need a notion of the size of a tree. The measure we will use will be *height*:

Definition 6.43. The *height* function $h : T(\mathcal{F}, \mathcal{X}) \to \mathbb{N}$ is defined by $h(x) = 0$ for all $x \in \mathcal{X}$, $h(t) = 1$ for all $t \in \mathcal{F}_0$ and $h(t(t_1, \ldots, t_n)) = 1 + \max(h(t_1, \ldots, t_n))$ for all $t \in \mathcal{F}_n$ $(n \geq 1)$ and $t_1, \ldots, t_n \in T(\mathcal{F}, \mathcal{X})$. For a language L, the set $\{t \in L | h(t) = k\}$ is denoted $L_{=k}$.

For example, taking the earlier example of binary trees, ground terms of height 3 include $f(f(c, c), f(c, c))$, $f(c, f(c, c))$ and $g(f(c, c))$.

We say that L has *doubly exponential* antichain growth if there is some ϵ such that the maximum size antichain in $L_{=n}$ exceeds $2^{2^{\epsilon n}}$ infinitely often.

Definition 6.44. A *nondeterministic finite tree automaton* (NFTA) over $\mathcal{F}$ is a tuple $\mathcal{A} = (Q, \mathcal{F}, Q_f, \Delta)$ where Q is a set of unary states, $Q_f \subseteq Q$ is a set of final states, and Δ a set of transition rules of type

$$f(q_1(x_1), \ldots, q_n(x_n)) \to q(f(x_1, \ldots, x_n)),$$

for $f \in \mathcal{F}_n$, $q, q_1, \ldots, q_n \in Q$ and $x_1, \ldots, x_n \in \mathcal{X}$. The *move relation* $\underset{\mathcal{A}}{\to}$ is defined by applying a transition rule possibly inside a context and possibly with substitutions for the x_i. The reflexive transitive closure of $\underset{\mathcal{A}}{\to}$ is denoted $\overset{*}{\underset{\mathcal{A}}{\to}}$.

A tree $t \in T(\mathcal{F})$ is *accepted* by $\mathcal{A}$ if there is some $q \in Q_f$ such that $t \overset{*}{\underset{\mathcal{A}}{\to}} q(t)$. The set of trees accepted by $\mathcal{A}$ is denoted $\mathcal{L}(\mathcal{A})$.

Again this generalises the definition of an NFA: put in transitions $\epsilon \to q(\epsilon)$ for all accepting states q, $a(q(x)) \to q'(a(x))$ whenever $q \in \Delta(q', a)$, and set Q_f as the initial state.

The critical idea for the proof is to find the appropriate analogue of L_q. This turns out to be the set P_q of binary contexts such that if the free variables are assigned state q then the root can also be given state q. By analogy to the 'trousers decomposition' of differential geometry,[5] we refer to such a context as a *pair of trousers*.

It turns out that a sufficient condition for L to have doubly exponential antichain growth is for P_q to be non-empty for some q (note that this does not depend on the particular partial order on Σ). On the other hand, if P_q is empty for all q, then there is in a suitable sense no branching and so we have a similar situation to ordinary languages.

Definition 6.45. Let $\mathcal{A} = (Q, \mathcal{F}, Q_f, \Delta)$ be a NFTA and $q \in Q$. A linear term $t \in T(\mathcal{F}, \{x_1, x_2\})$ is a *pair of trousers* with respect to q if x_1, x_2 appear in t and $t[x_1 \leftarrow q(x_1), x_2 \leftarrow q(x_2)] \overset{*}{\underset{\mathcal{A}}{\to}} q(t)$. The set of pairs of trousers with respect to q is denoted $P_q(\mathcal{A})$.

Lemma 6.46. *Let $\mathcal{A} = (Q, \mathcal{F}, Q_f, \Delta)$ be a reduced NFTA. If there exists some $q \in Q$ such that $P_q(\mathcal{A})$ is non-empty, then $\mathcal{L}(\mathcal{A})$ contains a doubly exponential antichain.*

Proof. We will clearly be done if we can find two pairs of trousers t_1, t_2 such that $\sigma_1(t_1) \not\sim \sigma_2(t_2)$ for all substitutions σ_1, σ_2: the set of trees built from them is of doubly exponential size, and any two such trees are comparable only if they are constructed in exactly the same way, i.e. are equal. We produce this pair by first

[5] Also known as the 'pants decomposition'

constructing two incomparable ground terms s_1, s_2 whose roots can be labelled with state q. Having done this we produce t_1 by attaching s_1 to the left leg of our pair of trousers t, and a copy of t to the right leg. For t_2 we do likewise but with s_2 in place of s_1. Since $s_1 \not\sim s_2$ we have that $\sigma_1(t_1) \not\sim \sigma_2(t_2)$ for all substitutions σ_1, σ_2.

Let t be a pair of trousers with respect to q and let s be a ground term with $s \xrightarrow[\mathcal{A}]{*} q(s)$. We claim that there exist incomparable ground terms s_1, s_2 with $s_i \xrightarrow[\mathcal{A}]{*} q(s_i)$.

Indeed, we have that s and $s' = t[x_1 \leftarrow s, x_2 \leftarrow s]$ are ground terms with $s \xrightarrow[\mathcal{A}]{*} q(s)$ and $s' \xrightarrow[\mathcal{A}]{*} q(s')$. Let $s_1 = t[x_1 \leftarrow s, x_2 \leftarrow s']$ and $s_2 = t[x_1 \leftarrow s', x_2 \leftarrow s]$. Now $s_1 \preceq s_2$ only if $s \preceq s'$ and $s' \preceq s$, which is impossible as $s \neq s'$ (since $h(s') > h(s)$). Similarly we have that $s_2 \not\preceq s_1$, as required.

Hence $t_1 = t[x_1 \leftarrow s_1, x_2 \leftarrow t]$ and $t_2 = t[x_1 \leftarrow s_2, x_2 \leftarrow t]$ are pairs of trousers with the property that $\sigma_1(t_1) \not\sim \sigma_2(t_2)$ for all substitutions σ_1, σ_2. It is clear that a doubly exponential antichain can be built from these. □

Lemma 6.47. *Let $\mathcal{A} = (Q, \mathcal{F}, Q_f, \Delta)$ be a reduced NFTA such that $P_q(\mathcal{A}) = \emptyset$ for all $q \in Q$. Then $\mathcal{L}(\mathcal{A})$ has at most exponential growth.*

Proof. We proceed by induction on the number of states appearing on the left of transitions. Without loss of generality we may assume that $Q_f = \{q\}$ for some q (otherwise consider a finite union of automata). Let $t \in \mathcal{L}(\mathcal{A})_{\leq n}$ be any term of height at most n. Say $t = f(t_1, \ldots, t_k)$ for some function symbol f and terms $t_1, \ldots, t_k$. In any accepting run for t, since the root is labelled with q we have that q can appear in at most one subtree, since otherwise we obtain a pair of trousers. Hence for all but at most one value of i we have that $t_i \in \mathcal{L}(\mathcal{A}')$, where $\mathcal{A}'$ is $\mathcal{A}$ with all transitions in which q appears on the left removed, which has at most single exponential language growth by the inductive hypothesis.

Hence we have

$$|\mathcal{L}(\mathcal{A})|_{\leq n} \leq |\mathcal{F}| d |\mathcal{L}(\mathcal{A})|_{\leq n-1} |\mathcal{L}(\mathcal{A}')|^d_{\leq n-1},$$

where d is the maximum arity of symbols in $\mathcal{F}$. Hence $\mathcal{L}(\mathcal{A})$ has at most single exponential language growth. □

In the case where there are no pairs of trousers, the situation is essentially equivalent to ordinary NFA, and so we have a further dichotomy between exponential and polynomial antichain growth. To show this, we define a set equivalent to $L_{q,q}$, and show that we have polynomial growth if it is a chain and exponential growth otherwise.

Definition 6.48. Let $\mathcal{A} = (Q, \mathcal{F}, Q_f, \Delta)$ be a NFTA, and $q \in Q$. Define $\mathcal{L}_q(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}) \cap T(\mathcal{F}, \{x_1\})$ to be the set of unary contexts t such that $t[x_1 \leftarrow q(x_1)] \xrightarrow[\mathcal{A}]{*} q(t)$.

Note that unary contexts are linear terms in which *exactly* one free variable appears, so $\mathcal{L}_q(\mathcal{A})$ does not contain ground terms. Note also that $x_1 \in \mathcal{L}_q(\mathcal{A})$ for any $\mathcal{A}$.

To give meaning to the statement '$\mathcal{L}_q(\mathcal{A})$ is a chain', we must extend the definition of the lexicographic order from the set $T(\mathcal{F})$ of ground terms to the set $T(\mathcal{F}, \{x_1\})$ of unary contexts. We do this by extending the relation $\leq$ on $\mathcal{F}$ to $\mathcal{F} \cup \{x_1\}$ by $x_1 \leq f$ for all $f \in \mathcal{F}$, and extending this to the lexicographic order as before.

Note in particular we have that if $t = \sigma(t')$ for some substitution σ then we have $t' \preceq t$; this corresponds to the notion of prefixes for words. On the other hand, if $t' \preceq t$ then we have that either $t = \sigma(t')$ for some σ (t' is a prefix of t) or otherwise that $\sigma'(t') \preceq \sigma(t)$ for all substitutions σ, σ'. Conversely, if $t \not\sim t'$ then we have that $\sigma(t) \not\sim \sigma'(t')$ for all substitutions σ, σ'; note that this does not hold for contexts of arity greater than 1 (for a similar definition of the lexicographic order).

Lemma 6.49. *Let $\mathcal{A} = (Q, \mathcal{F}, Q_f, \Delta)$ be a reduced NFTA such that $P_q(\mathcal{A}) = \emptyset$ for all q. Then $\mathcal{L}(\mathcal{A})$ has polynomial antichain growth if $\mathcal{L}_q(\mathcal{A})$ is a chain for all q, and otherwise $\mathcal{L}(\mathcal{A})$ has exponential antichain growth.*

Proof. If $\mathcal{L}_q(\mathcal{A})$ is not a chain then let $t_1 \not\sim t_2 \in \mathcal{L}_q(\mathcal{A})$. Since $\mathcal{A}$ is reduced there is a ground term t with $t \xrightarrow[\mathcal{A}]{*} q(t)$ and a unary context t' with $t'(q(x)) \xrightarrow[\mathcal{A}]{*} q'(t)$ for some $q' \in Q_f$. Let the function $\phi : \mathcal{P}(T(\mathcal{F})) \to \mathcal{P}(T(\mathcal{F}))$ be defined by $\phi(X) = \{t_1[x_1 \leftarrow s], t_2[x_1 \leftarrow s] | s \in X\}$, and let $Y = \bigcup_{n=0}^{\infty} \phi^n(\{t\})$. Then the set $\{t'[x_1 \leftarrow s] | s \in Y\} \subseteq \mathcal{L}(\mathcal{A})$ is an antichain and has exponential growth.

Conversely if $\mathcal{L}_q(\mathcal{A})$ is a chain for all q then an argument similar to the upper bound in the proof of Theorem 6.16 shows that $\mathcal{L}(\mathcal{A})$ has polynomial antichain growth.

Once again we proceed by induction on the number of states appearing on the left of transitions, and assume without loss of generality that $Q_f = \{q\}$ for some q. Then for any $t \in \mathcal{L}(\mathcal{A})$ we have that $t = t'[x_1 \leftarrow t'']$ for some $t' \in \mathcal{L}_q(\mathcal{A})$ and $t'' \in \mathcal{L}(\mathcal{A}')$, where $\mathcal{A}'$ is $\mathcal{A}$ with all transitions in which q appears on the left removed, which has polynomial antichain growth by the inductive hypothesis.

For any antichain $L \subseteq \mathcal{L}(\mathcal{A})$, we claim that we have that

$$L \subseteq \{t[x_1 \leftarrow t'] | t' \in \mathcal{L}(\mathcal{A}')\}$$

for some fixed $t \in \mathcal{L}_q(\mathcal{A})$. Indeed, supposing the contrary let $t_1 \neq t_2 \in \mathcal{L}_q(\mathcal{A})$ be contexts such that $t_1[x_1 \leftarrow t_1'], t_2[x_1 \leftarrow t_2'] \in L$ with $t_1 \neq \sigma(t_2), t_2 \neq \sigma(t_1)$ for all

substitutions σ. Since $\mathcal{L}_q(\mathcal{A})$ is a chain we have that (without loss of generality) $t_1 \preceq t_2$ and since t_1 is not a prefix of t_2, we have that $\sigma_1(t_1) \preceq \sigma_2(t_2)$ for all substitutions σ_1, σ_2. In particular we have that $t_1[x_1 \leftarrow t_1'] \preceq t_2[x_1 \leftarrow t_2']$, which is a contradiction since L is an antichain, so the claim is proved.

Hence by induction we have that $\mathcal{L}(\mathcal{A})$ has polynomial antichain growth. □

Combining these lemmas gives

Theorem 6.50. *Let L be a regular tree language over a partially ordered alphabet. Then L has either doubly exponential antichain growth, singly exponential antichain growth, or polynomial antichain growth.*

The special case of the trivial partial order (in which elements are only comparable to themselves) yields the fact that the language growth of any regular tree language is either polynomial, exponential or doubly exponential, which we are not aware of having appeared in the literature.

Corollary 6.51. *Let L be a regular tree language. Then L has either doubly exponential language growth, singly exponential language growth or polynomial language growth.*

Finally, we show that there is a polynomial algorithm to detect doubly exponential growth, by determining whether or not the language of a given NFTA contains a pair of trousers.

Theorem 6.52. *There exists a polynomial time algorithm to determine whether the language of a given NFTA has doubly exponential growth*

Proof. We show how to determine whether $T_{q_0}(\mathcal{A}) = \emptyset$ for fixed q_0.

We proceed similarly to the Reduction Algorithm in [17] (p.25), which iteratively computes the set M of states q such that $t \xrightarrow[\mathcal{A}]{*} q(t)$ for some t. We first iteratively compute the set M' of states q such that there is a unary context $t \in T(\mathcal{F}, \{x_1\})$ such that $t[x_1 \leftarrow q_0] \xrightarrow[\mathcal{A}]{*} q$. We can then iteratively compute the set M'' of states q such that there is a binary context $t \in T(\mathcal{F}, \{x_1, x_2\})$ such that $t[x_1 \leftarrow q_0, x_2 \leftarrow q_0] \xrightarrow[\mathcal{A}]{*} q$. Then $T_{q_0}(\mathcal{A}) \neq \emptyset$ if and only if $q_0 \in M''$.

Concretely, the reduction algorithm from [17] proceeds as follows. Initialise the set $X = \emptyset$. For each transition rule $f(q_1(x_1), \ldots, q_n(x_n)) \to q$ in Δ such that $q_1, \ldots, q_n \in X$, add q to X. Repeat this process until X no longer changes. Then $X = M$ is the set of accessible states.

To compute the set M' of states q such that $t[x_1 \leftarrow q_0] \xrightarrow[\mathcal{A}]{*} q$ for some unary context t, first initialise the set $X' = \{q_0\}$. For each transition rule $f(q_1(x_1), \ldots, q_n(x_n)) \to q$ in Δ such that we have $q_1, \ldots, q_{k-1}, q_{k+1}, \ldots, q_n \in M$ and $q_k \in X'$ for some k, add q to X'. Repeat this until X' no longer changes, and then we have $X' = M'$.

Finally we compute the set M'' of states q such that $t[x_1 \leftarrow q_0, x_2 \leftarrow q_0] \xrightarrow[\mathcal{A}]{*} q$ for some binary context t. First initialise X'' to be the set of states q such that there is some transition rule $f(q_1(x_1), \ldots, q_n(x_n)) \to q$ in Δ where f has arity at least 2, and we have $q_1, \ldots, q_{k-1}, q_{k+1}, \ldots, q_{l-1}, q_{l+1}, \ldots, q_n \in M$ and $q_k, q_l \in M'$ for some $k < l$.

For the iterative step, for each transition rule $f(q_1(x_1), \ldots, q_n(x_n)) \to q$ in Δ such that we have $q_1, \ldots, q_{k-1}, q_{k+1}, \ldots, q_n \in M$ and $q_k \in X''$ for some k, add q to X''. Repeat this until X'' stabilises and then we have $M'' = X''$. □

6.8 Conclusions

The main results of this chapter are Theorem 6.16 and Theorem 6.18, which in combination with Chapters 4 and 5 show that there is a dichotomy between 'safe' (i.e. logarithmic) and 'dangerous' (i.e. linear) information flow, corresponding to polynomial and exponential antichain growth respectively, and an algorithm to tell the difference. For the case of polynomial antichain growth, we have also shown (Theorem 6.28) that there is an algorithm to compute the precise order of polynomial growth, corresponding to the constant factor in the information flow.

We have also studied the attractive topic of antichain growth in some other automatic structures. For context-free languages, we have shown that there is the same dichotomy as for regular languages (Theorem 6.34, but the problem of distinguishing the two cases for a given context-free grammar is undecidable (Theorem 6.40. Finally we studied the question for regular tree languages and showed (Theorem 6.50) that there is a trichotomy between doubly exponential, singly exponential and polynomial antichain growth.

Chapter 7

Model shifting

7.1 Introduction

The study of non-interference and information flow in the context of process algebra has a rich history, some of which is helpfully described by Ryan and Schneider in [60]. In this chapter we show how to extend this work to quantified information flow using the machinery developed in the previous chapters. Because the standard notions of non-interference are inherently dependent on semantic models richer than the traces model, and because the previous chapters have worked exclusively in terms of the languages (that is, sets of traces), it is first necessary to show how to represent the behaviours of a process in the traces model. In fact we are able to show that for a wide class of behavioural models, which in particular encompasses all known finite observational models, there is a construction to achieve this, which we refer to as 'model shifting'. This work has appeared as [46].

A number of different forms of process calculus have been developed for the modeling of concurrent programs, including Hoare's Communicating Sequential Processes (CSP) [34], Milner's Calculus of Communicating Systems (CCS) [47], and the π-calculus [48]. Unlike the latter two, CSP's semantics are traditionally given in behavioural semantic models coarser than bisimulation.

In this chapter, we study finite linear-time observational models for CSP; that is, models where all observations considered can be determined in a finite time by an experimenter who can see the visible events a process communicates and the sets of events it can offer in any stable state. While the experimenter can run the process arbitrarily often, he or she can only record the results of individual finite executions. Thus each behaviour recorded can be deduced from a single finite sequence of events together with the sets of events accepted in stable states during and immediately after this *trace*.

At least six such models have been considered for CSP, but the state-of-the art refinement checking tool, FDR3 [27], currently only supports two, namely *traces* and *failures* (it also supports the failures-divergences model, which is not finite observational).

We present a construction which produces a context $\mathcal{C}$ such that refinement questions in the failures model correspond to trace refinement questions under the application of $\mathcal{C}$. We are able to generalise this to show (Theorem 7.30) that a similar construction is possible not only for the six models which have been studied, but also for any sensible finite observational model (where 'sensible' means that the model can be recognised by a finite-memory computer, in a sense which we shall make precise).

The structure of this chapter is as follows. In Section 7.2 we briefly summarise approaches to non-interference in CSP. In Section 7.3 we show that the usual definition of total non-interference is in some sense model-independent, whereas we will find that quantified non-interference does require consideration of the appropriate semantic model. In Section 7.4 we briefly describe the language of CSP. In Section 7.6 we give a formal definition of a finite observational model, and of the notion of rationality, and in Section 7.7 we prove the main theorem showing that model shifting is possible for rational models. In Section 7.8 we discuss implementation issuses. In Section 7.9 we show how to use this technique to define logarithmic and linear information flow for the finite linear observations model, and remark that by Chapter 6 they can be distinguished in polynomial time.

7.2 Noninterference and CSP

In this section we outline various notions of noninterference that have been considered in the context of CSP, and how to extend them to quantified noninterference.

The notion of *non-deducibility on compositions* was introduced by Focardi and Gorrieri in [25], and is helpfully described by Lowe in [43]. It captures the intuitively plausible idea that the system is secure if the interface exposed to *Lo* is the same no matter what *Hi* is:

Definition 7.1. Process P satisfies *traces non-deducibility on compositions* if and only if for all processes Hi_1, Hi_2 over H, we have

$$(P \underset{H}{\parallel} Hi_1) \setminus H \equiv_{\mathrm{T}} (P \underset{H}{\parallel} Hi_2) \setminus H.$$

Focardi and Gorrieri observe that this definition is inadequate if the user is able to detect refusals; for instance the process

$$P = h \rightarrow (l \rightarrow STOP \sqcap STOP) \,\Box\, l \rightarrow STOP \tag{7.1}$$

satisfies TNDC, but Low can detect whether h has occured based on whether l can be refused.

Equipped as we are with the rich universe of CSP denotational models, we are well-prepared to solve this problem for any choice of model for how the user is able to observe the process (introduced by Focardi for the failures model in [24]):

Definition 7.2. Let $\mathcal{M}$ be a denotational model. Process P satisfies $\mathcal{M}$ *nondeducibility on compositions* if and only if for all processes Hi_1, Hi_2 over H, we have

$$(P \underset{H}{\parallel} Hi_1) \setminus H \equiv_M (P \underset{H}{\parallel} Hi_2) \setminus H.$$

Note that in [43], Lowe takes the view that possible divergence which may (but need not) be introduced by the actions of High should not be considered to give rise to information flow, and consequently introduces lazy abstraction (see Definition 7.3) at this stage, ostensibly to prevent this. We do not believe that it does so, and in any case we do not agree that a situation where High has the option of causing a livelock or not can be considered to be one of noninterference.

However, even with the failures model (or indeed richer models) the notion of nondeducibility on compositions is flawed, for a reason that is most clearly expressed by the well-known *refinement paradox*. Since the statement $P \sqsubseteq_{\mathrm{F}} Q$ means that Q is a valid implementation of P, we should certainly have that if P is secure and $P \sqsubseteq_{\mathrm{F}} Q$ then also Q is secure.

Suppose that $P = CHAOS$. Then certainly P satisfies non-deducibility on compositions, since in particular for any model $\mathcal{M}$ we have that $(P \underset{H}{\parallel} Hi) \setminus H \equiv_M P \setminus H$ for any process Hi. On the other hand, let $LEAK$ be some insecure process. Then since $CHAOS$ is refinement-minimal in all models we have $P \sqsubseteq_M LEAK$.

The solution to this is to adopt the definition of *lazy abstraction*, first introduced by Roscoe, Woodcock and Wulf in [58], which essentially involves considering the combination of the system and the high-level user, and asking whether this involves any nondeterminism visible to the low-level user.

Definition 7.3. The *lazy abstraction* of a process P with respect to a set of events H is denoted $\mathcal{L}_H(P)$ and is defined by

$$\mathcal{L}_H(P) = (P \underset{H}{\parallel} CHAOS_H) \setminus H.$$

If $\mathcal{L}_H(P)$ is deterministic then we say that P is *lazily independent with respect to* H.

One technicality here is how to deal with the situation where $\mathcal{L}_H$ introduces divergence: since the definition of determinism excludes divergence, a process P for which $\mathcal{L}_H(P)$ is divergent will be viewed as insecure. In [58] the authors avoid this by interleaving with RUN_H instead of hiding H, since both of these conceal from the low-level user which H-actions were actually performed by the high-level user. However, this is also not altogether satisfactory: if the system has the option either to livelock or not from the perspective of the low-level user then this ought surely to be viewed as a source of information flow. Since this issue is only tangentially relevant for our purposes, we will make an assumption of fairness and view divergence of $\mathcal{L}_H(P)$ as inconsistent with the security of P.

Note that lazy abstraction is closely analagous to the construction of 5.12, in which we similarly give Alice a nondeterminsitic choice among her actions, and hide these actions from Bob.

7.3 Quantified information flow

How should we quantify information flow in CSP? One option, proposed by Lowe in [42], is essentially based on nondeducibility on compositions: we count the number of different views for Low that can be produced by sutable strategies for High, where views are regarded as 'different' if they can be distinguished by a suitable testing strategy for Low. This suffers from the same problems as the original definition of noninterference on compositions, and so we do not discuss it further.

Instead, we will eventually base a definition on lazy abstraction, similarly to the definitions of the previous chapters. However, we will have to give careful consideration to the appropriate semantic model. For the question of complete noninterference this does not arise, because as the next theorem shows, the definition of determinism does not depend on our choice of model. Recall that one definition of determinism is that the process is divergence-free and $\sqsubseteq_{\mathrm{F}}$-maximal among divergence-free processes. The result is that the property of being $\sqsubseteq_M$-maximal is equivalant among all finite observational models $\mathcal{M}$ with $\mathcal{F} \preceq \mathcal{M}$.

Theorem 7.4. *Let $\mathcal{M}$ be a finite observational model with $\mathcal{F} \preceq \mathcal{M}$. Then P is maximal with respect to $\sqsubseteq_M$ among divergence-free processes if and only if it is maximal with respect to $\mathcal{F}$.*

Recall ([53]) that all finite observational models $\mathcal{M} \neq \mathcal{T}$ have $\mathcal{F} \preceq \mathcal{M}$. Clearly the only $\sqsubseteq_{\mathrm{T}}$-maximal process is $STOP$.

We prove this theorem by defining the *refined-determinisation* of P, denoted $\mathrm{Det}(P)$, by deleting some transitions from P so as to leave essentially a DFA. We will then show that for any model $\mathcal{M}$ we have that P is $\sqsubseteq_M$-maximal if and only if $P \equiv_{\mathrm{F}} \mathrm{Det}(P)$.

Definition 7.5. Let P be a CSP process. The *refined-determinisation* of P, denoted $\mathrm{Det}(P)$, is the process whose LTS is formed from that of P as follows:

(i) For states p of P with at least one τ out-arrow, choose a single τ-arrow, say to state q. Replace all arrows into p with corresponding arrows into q, and delete p.

(ii) For other states, choose for each letter $x \in \Sigma$ a single out-arrow labelled by x to retain and delete all other out-arrows labelled by x.

(iii) Delete all states which are now not reachable from the initial state.

Note that $\mathrm{Det}(P)$ has the property that for each trace s we have that P can be in at most a single state after trace s. We will call processes which have this property *unambiguous*.

Lemma 7.6. *Let P be a process and $\mathcal{M}$ a finite observational model. Then $P \sqsubseteq_M \mathrm{Det}(P)$.*

Proof. It suffices to show this for $\mathcal{M} = \mathcal{FL}$. It is clear that P simulates $\mathrm{Det}(P)$, and that for any state q of $\mathrm{Det}(P)$, the corresponding state of P has the same acceptance-set as q. □

Lemma 7.7. *Let P be a process with $P \equiv_{\mathrm{F}} \mathrm{Det}(P)$. Then for any finite observational model $\mathcal{M}$, we have $P \equiv_M \mathrm{Det}(P)$.*

Proof. By Lemma 7.6 it remains to show that $\mathrm{Det}(P) \sqsubseteq_M P$. Again it suffices to show this for $\mathcal{M} = \mathcal{FL}$. Let $\langle A_0, a_1, A_1, \ldots, A_{n-1}, a_n, A_n\rangle = s^\frown\langle A_{n-1}, a_n, A_n\rangle$ be a minimum-length element of $\mathcal{FL}(P) \setminus \mathcal{FL}(\mathrm{Det}(P))$ (note that WLOG $A_n \neq \bullet$ by divergence-freedom). Then $a_n \in A_{n-1}$ and by minimality $s^\frown\langle A_{n-1}\rangle \in \mathcal{FL}(\mathrm{Det}(P))$, so $s^\frown\langle A_{n-1}, a_n, X\rangle \in \mathcal{FL}(\mathrm{Det}(P))$ for some X. Hence $s^\frown\langle A_{n-1}, a_n, X\rangle \in \mathcal{FL}(P)$, so by determinism of P we have $X = A_n$.↯ □

Lemma 7.8. *Let P be an unambiguous process and $\mathcal{M}$ a finite observational model with $\mathcal{F} \preceq \mathcal{M}$. Then P is $\sqsubseteq_M$-maximal.*

Proof. Suppose $P \sqsubseteq_M Q$. Now by Lemma 7.6 we have $P \sqsubseteq_M Q \sqsubseteq_M \mathrm{Det}(Q)$. But P is deterministic and hence $\sqsubseteq_{\mathrm{F}}$-maximal, so $P \equiv_{\mathrm{F}} \mathrm{Det}(Q)$. Now since P and $\mathrm{Det}(Q)$ are unambiguous, there is a unique state of each that can be reached after any trace s. Hence the finite linear observations are completely determined by the failures and so $P \equiv_{\mathrm{FL}} \mathrm{Det}(Q)$. In particular $\mathrm{Det}(Q) \sqsubseteq_M P$ and so $Q \equiv_M P$. □

Proof of Theorem 7.4. If P is $\sqsubseteq_{\mathrm{F}}$-maximal, then by Lemma 7.6 we have $P \equiv_{\mathrm{F}} \mathrm{Det}(P)$. Hence by Lemma 7.7 we have $P \equiv_M \mathrm{Det}(P)$. By Lemma 7.8, $\mathrm{Det}(P)$ is $\sqsubseteq_M$-maximal and hence so is P.

Conversely, if P is $\sqsubseteq_M$-maximal then $P \equiv_M \mathrm{Det}(P)$ and so $P \equiv_{\mathrm{F}} \mathrm{Det}(P)$. Since $\mathrm{Det}(P)$ is $\sqsubseteq_{\mathrm{F}}$-maximal, so is P. □

For quantified information flow, however, the question of what semantic model we use—that is to say, what behaviours of the system we consider that the low-level user is able to observe—will become important. However, since all of our previous work has been expressed in terms of traces, it will be helpful to be able to represent richer behavioural models in the form of trace-sets. In the following sections of this chapter we prove a general result, which may be of independent interest, which shows (Theorem 7.30) that for all of a very general class of finite observational models (including all the examples which are known), there exists a suitable context $\mathcal{C}_{\mathcal{M}}$ such that the $\mathcal{M}$-behaviours of P are in correspondence with the traces of $\mathcal{C}_{\mathcal{M}}[P]$.

7.4 The CSP language

We provide a brief outline of the language, largely taken from [53]; the reader is encouraged to consult [54] for a more comprehensive treatment.

Throughout, Σ is taken to be a finite nonempty set of communications that are visible and can only happen when the observing environment permits via handshaken communication. The actions of every process are taken from $\Sigma \cup \{\tau\}$, where τ is the invisible internal action that cannot be prevented by the environment. Note that the usual treatment of CSP permits sequential composition by including another unpreventable event $\checkmark$ to represent termination; this adds slight complications to each model and we omit it for simplicity. It could be added back without any significant alteration to the results of this chapter.

The constant processes of CSP are

- *STOP* which does nothing—a representation of deadlock.
- **div** which performs (only) an infinite sequence of internal τ actions—a representation of divergence or livelock.
- *CHAOS* which can do anything except diverge.

The prefixing operator introduces communication:

- $a \rightarrow P$ communicates the event a before behaving like P.

There are two forms of binary choice between a pair of processes:

- $P \sqcap Q$ lets the process decide to behave like P or like Q: this is *nondeterministic* or *internal* choice.
- $P \ \Box \ Q$ offers the environment the choice between the initial Σ-events of P and Q. If the one selected is unambiguous then it continues to behave like the one chosen; if it is an initial event of both then the subsequent behaviour is nondeterministic. The occurence of τ in one of P and Q does *not* resolve the choice (unlike CCS +). This is *external* choice.

We only have a single parallel operator in our core language since all the usual ones of CSP can be defined in terms of it as discussed in Chapter 2 etc. of [54].

- $P \underset{X}{\parallel} Q$ runs P and Q in parallel, allowing each of them to perform any action in $\Sigma \setminus X$ independently, whereas actions in X must be synchronised between the two.

There are two operators that change the nature of a process's communications.

- $P \setminus X$, for $X \subseteq \Sigma$, *hides* X by turning all P's X-actions into τs.
- $P[\![R]\!]$ applies the *renaming* relation $R \subseteq \Sigma \times \Sigma$ to P: if $(a, b) \in R$ and P can perform a, then $P[\![R]\!]$ can perform b. The domain of R must include all visible events used by P. Renaming by the relation $\{(a, b)\}$ is denoted $[\![a/b]\!]$.

There is another operator that allows one process to follow another:

- $P \Theta_A Q$ behaves like P until an event in the set A occurs, at which point P is shut down and Q is started. This is the *throw* operator.

The final CSP construct is *recursion*: this can be single or mutual (including mutual recursions over infinite parameter spaces), can be defined by systems of equations or (in the case of single recursion) in line via the notation $\mu\, p.P$, for a term P that may include the free process identifier p. Recursion can be interpreted operationally as having a τ-action corresponding to a single unwinding. Denationally, we regard P as a function on the space of denotations, and interpret $\mu\, p.P$ as the least fixed point of this function.

We also make use of the *interleaving* operator $|||$, which allows processes to perform actions independently and is equivalent to $\underset{\emptyset}{\|}$, and the process RUN_X, which always offers every element of the set X and is defined by $RUN_X = \Box_{x \in X}\, x \to RUN_X$.

7.4.1 Priority

The prioritisation operator is discussed in detail in Chapter 20 of [54]. It allows us to specify an ordering on the set of visible events Σ, and prevents lower-priority events from occuring whenever a higher-priority event or τ is available.

The operator described in [54] as implemented in FDR3 [27] is parametrised by three arguments: a process P, a partial order $\leq$ on the event set Σ, and a subset $X \subseteq \Sigma$ of events that can occur when a τ is available. We require that all elements of X are maximal with respect to $\leq$. Writing $\mathit{initials}(P) \subseteq \Sigma \cup \{\tau\}$ for the set of events that P can immediately perform, and extending $\leq$ to a partial order on $\Sigma \cup \{\tau\}$ by adding $y \leq \tau\, \forall\, y \in \Sigma \setminus X$, we define the operational semantics of prioritise as follows:

$$\frac{P \xrightarrow{a} P' \wedge \forall\, b \neq a . a \leq b \Rightarrow b \notin \mathit{initials}(P)}{\text{prioritise}(P, \leq, X) \xrightarrow{a} \text{prioritise}(P', \leq, X)}\ (a \in \Sigma \cup \{\tau\}).$$

Note that prioritise is not compositional over denotational models other than the most precise model $\mathcal{FL}$, so we think of it as an optional addition to CSP rather than an integral part of it; when we refer below to particular types of observation as giving rise to valid models for CSP, we will mean CSP without priority.

7.5 Example: the failures model

We first demonstrate our construction using the *failures* model: we will produce a context $\mathcal{C}$ such that for any processes P, Q, we have that Q refines P in the failures model if and only $\mathcal{C}[Q]$ refines $\mathcal{C}[P]$ in the traces model.

7.5.1 The traces and failures models

The *traces* model $\mathcal{T}$ is familiar from automata theory, and represents a process by the set of (finite) strings of events it is able to accept. Thus each process is associated (for fixed alphabet Σ) to a subset of Σ^* the set of finite words over Σ.

The *failures* model $\mathcal{F}$ also records sets X of events that the process is able to stably refuse after a trace s (that is, the process is able after trace s to be in a state where no τ events are possible, and where the set of initial events does not meet X). Thus a process is associated to a subset of $\Sigma^* \times (\mathcal{P}(\Sigma) \cup \{\bullet\})$, where $\bullet$ represents the absence of a recorded refusal set.[1] Note that recording $\bullet$ does not imply that there is no refusal to observe, simply that we have not observed stability. The observation of the refusal $\emptyset$ implies that the process can be stable after the present trace, whereas $\bullet$ does not.

In any model $\mathcal{M}$, we say that Q *$\mathcal{M}$-refines* P, and write $P \sqsubseteq_M Q$, if the set associated to Q is a subset of that corresponding to P.

7.5.2 Model shifting for the failures model

The construction is as follows:

Lemma 7.9. *For each finite alphabet Σ there exists a context $\mathcal{C}$ (over an expanded alphabet) such that for any processes P and Q we have that $P \sqsubseteq_F Q$ if and only if $\mathcal{C}[P] \sqsubseteq_T \mathcal{C}[Q]$.*

Proof. **Step 1:** We use priority to produce a process (over an expanded alphabet) that can communicate an event x' if and only if the original process P is able to stably refuse x.

This is done by expanding the alphabet Σ to $\Sigma \cup \Sigma'$ (where Σ' contains a corresponding primed event for every event in Σ), and prioritising with respect to a partial order which prioritises each x over the corresponding x'. Recall that the definition of the priority operator means that this also causes τ to be promoted over the primed events.

We must also introduce an event *stab* to signify stability without requiring any refusals to be possible. This is necessary in order to be able to record an empty refusal set. Let the partial order $\leq_1$ be defined by $x' <_1 x \,\forall\, x \in \Sigma$, and let the context $\mathcal{C}_1$ be defined by

$$\mathcal{C}_1[P] = \text{prioritise}(P \,|||\, RUN_{\Sigma' \cup \{stab\}}, \leq_1, \Sigma).$$

[1] This is equivalent to the standard presentation in which a process is represented by a subset of Σ^* and one of $\Sigma^* \times \mathcal{P}(\Sigma)$: the trace component is just $\{s : (s, \bullet) \in \mathcal{P}\}$.

This process has a state ξ' for each state ξ of P, where ξ' has the same unprimed events (and corresponding transitions) as ξ. Furthermore ξ' can communicate x' just when ξ is stable and can refuse X, and *stab* just when ξ is stable.

Step 2: We now recall that the definition of the failures model only allows a refusal set to be recorded at the *end* of a trace, and is not interested in (so does not record) what happens after the refusal set.

We gain this effect by using a regulator process to prevent a primed event (or *stab*) from being followed by an unprimed event. Let

$$\begin{aligned} UNSTABLE &= \Box_{x \in \Sigma} x \to UNSTABLE \\ &\quad \Box \Box_{x \in \Sigma' \cup \{stab\}} x \to STABLE \\ STABLE &= \Box_{x \in \Sigma' \cup \{stab\}} x \to STABLE, \end{aligned}$$

and define $\mathcal{C}$ by

$$\mathcal{C}[P] = \mathcal{C}_1[P] \underset{\Sigma \cup \Sigma' \cup \{stab\}}{\|} UNSTABLE.$$

A trace of $\mathcal{C}[P]$ consists of: firstly, a trace s of P; followed by, if P can after s be in a stable state, then for some such state σ_0 any string formed from the events that can be refused in σ_0, together with *stab*. The lemma clearly follows. □

It is clear that any such context must involve an operator that is not compositional over traces, for otherwise we would have $P \sqsubseteq_{\mathrm{T}} Q$ implies $\mathcal{C}[P] \sqsubseteq_{\mathrm{T}} \mathcal{C}[Q]$, which is equivalent to $P \sqsubseteq_{\mathrm{F}} Q$, and this is not true for general P and Q (consider for instance $P = a \to STOP$, $Q = (a \to STOP) \sqcap STOP$). It follows that only contexts which like ours involve priority can achieve this.

7.6 Semantic models

In order to generalise this construction to arbitrary finite observational semantic models, we must give formal definitions not only of particular models but of the very notion of a finite observational model.

7.6.1 Finite observations

We consider only models arising from *finite linear observations*. Intuitively, we postulate that we are able to observe the process performing a finite number of visible actions, and that where the process was stable (unable to perform a τ) immediately

before an action, we are able to observe the *acceptance set* of actions it was willing to perform.

Note that we are unable to finitely observe *instability*: the most we are able to record from an action in an unstable state is that we did not *observe* stability. Thus in any context where we can observe stability we can also fail to observe it by simply not looking.

We take models to be defined over finite alphabets Σ, and take an arbitrary ordering on each finite Σ to be *alphabetical*.

The most precise finite observational model is that considering all finite linear observations, and is denoted $\mathcal{FL}$:

Definition 7.10. The set of *finite linear observations* over an alphabet Σ is

$$\mathcal{FL}_\Sigma := \{\langle A_0, a_1, A_1, \ldots, A_{n-1}, a_n, A_n\rangle : n \in \mathbb{N}, a_i \in \Sigma, A_i \subseteq \Sigma \text{ or } A_i = \bullet\},$$

where the a_i are interpreted as a sequence of communicated events, and the A_i denote stable acceptance sets, or in the case of $\bullet$ failure to observe stability. Let the set of such observations corresponding to a process P be denoted $\mathcal{FL}_\Sigma(P)$.

(Sometimes we will drop the Σ and just write $\mathcal{FL}(P)$).

More formally, $\mathcal{FL}(P)$ can be defined inductively; for instance

$$\mathcal{FL}(P \,\Box\, Q) := \{\langle A \cup B\rangle\hat{}\alpha, \langle A \cup B\rangle\hat{}\beta : \langle A\rangle\hat{}\alpha \in \mathcal{FL}(P), \langle B\rangle\hat{}\beta \in \mathcal{FL}(Q)\}$$

(where $X \cup \bullet := \bullet$ for any set X). See Section 11.1.1 of [54] for further details.

Observe that $\mathcal{FL}$ has a natural partial order corresponding to extensions (where $\alpha\hat{}\langle\bullet\rangle\hat{}\beta$ and $\alpha\hat{}\langle A\rangle$ are both extended by $\alpha\hat{}\langle A\rangle\hat{}\beta$ for any set A and any α and β). Note that for any process P we have that $\mathcal{FL}(P)$ is downwards-closed with respect to this partial order.

7.6.2 Finite observational models

We consider precisely the models which are derivable from the observations of $\mathcal{FL}$, which are well-defined in the sense that they are compositional over CSP syntax (other than priority), and which respect extension of the alphabet Σ.

Definition 7.11. A finite observational *pre-model* $\mathcal{M}$ consists for each (finite) alphabet Σ of a set of *observations*, $\mathrm{obs}_\Sigma(\mathcal{M})$, together with a relation $\mathcal{M}_\Sigma \subseteq \mathcal{FL}_\Sigma \times$

$\mathrm{obs}_\Sigma(\mathcal{M})$. The representation of a process P in $\mathcal{M}_\Sigma$ is denoted $\mathcal{M}_\Sigma(P)$, and is given by

$$\mathcal{M}_\Sigma(P) := \mathcal{M}_\Sigma(\mathcal{FL}_\Sigma(P)) = \{y \in \mathrm{obs}_\Sigma(\mathcal{M}) : \exists\, x \in \mathcal{FL}_\Sigma(P).(x, y) \in \mathcal{M}_\Sigma\}.$$

For processes P and Q over alphabet Σ, if we have $\mathcal{M}_\Sigma(Q) \subseteq \mathcal{M}_\Sigma(P)$ then we say Q *$\mathcal{M}$-refines* P, and write $P \sqsubseteq_M Q$.

(As before we will sometimes drop the Σ).

Note that this definition is less general than if we had defined a pre-model to be any equivalence relation on $\mathcal{P}(\mathcal{FL}_\Sigma)$. For example, the equivalence relating sets of the same cardinality has no corresponding pre-model. Definition 7.11 agrees with that sketched in [54].

Without loss of generality, $\mathcal{M}_\Sigma$ does not identify any elements of $\mathrm{obs}_\Sigma(\mathcal{M})$; that is, we have $\mathcal{M}_\Sigma^{-1}(x) = \mathcal{M}_\Sigma^{-1}(y)$ only if $x = y$ (otherwise quotient by this equivalence relation). Subject to this assumption, $\mathcal{M}_\Sigma$ induces a partial order on $\mathrm{obs}_\Sigma(\mathcal{M})$:

Definition 7.12. The partial order *induced by* $\mathcal{M}_\Sigma$ *on* $\mathrm{obs}_\Sigma(\mathcal{M})$ is given by: $x \leq y$ if and only if for all $b \in \mathcal{M}_\Sigma^{-1}(y)$ there exists $a \in \mathcal{M}_\Sigma^{-1}(x)$ with $a \leq b$.

Observe that for any process P it follows from this definition that $\mathcal{M}(P)$ is downwards-closed with respect to this partial order (since $\mathcal{FL}(P)$ is downwards-closed).

Definition 7.13. A pre-model $\mathcal{M}$ is *compositional* if for all CSP operators $\bigoplus$, say of arity k, and for all processes $P_1, \ldots, P_k$ and $Q_1, \ldots, Q_k$ such that $\mathcal{M}(P_i) = \mathcal{M}(Q_i)$ for all i, we have

$$\mathcal{M}\left(\bigoplus_{i=1}^{k} P_i\right) = \mathcal{M}\left(\bigoplus_{i=1}^{k} Q_i\right).$$

This means that the operator defined on processes in $\mathrm{obs}(\mathcal{M})$ by taking the push-forward of $\bigoplus$ along $\mathcal{M}$ is well-defined: for any sets $X_1, \ldots, X_k \subseteq \mathrm{obs}(\mathcal{M})$ which correspond to the images of CSP processes, take processes $P_1, \ldots, P_k$ such that $X_i = \mathcal{M}(P_i)$, and let

$$\bigoplus_{i=1}^{k} X_i = \mathcal{M}\left(\bigoplus_{i=1}^{k} P_i\right).$$

Definition 7.13 says that the result of this does not depend on the choice of the P_i.

Note that it is not necessary to require the equivalent of Definition 7.13 for recursion in the definition of a model, because of the following lemma which shows that least fixed point recursion is automatically well-defined (and formalises some arguments given in [54]):

Lemma 7.14. *Let $\mathcal{M}$ be a compositional pre-model. Let $\mathcal{C}_1, \mathcal{C}_2$ be CSP contexts, such that for any process P we have $\mathcal{M}(\mathcal{C}_1[P]) = \mathcal{M}(\mathcal{C}_2[P])$. Let the least fixed points of $\mathcal{C}_1$ and $\mathcal{C}_2$ (viewed as functions on $\mathcal{P}(\mathcal{FL})$ under the subset order) be P_1 and P_2 respectively. Then $\mathcal{M}(P_1) = \mathcal{M}(P_2)$.*

Proof. Using the fact that CSP contexts induce Scott-continuous functions on $\mathcal{P}(\mathcal{FL})$ (see [34], Section 2.8.2), the Kleene fixed point theorem gives that $P_i = \bigcup_{n=0}^{\infty} \mathcal{C}_i^n(\bot)$. Now any $x \in \mathcal{M}(P_1)$ is in the union taken up to some finite N, and since finite unions correspond to internal choice, and $\bot$ to the process **div**, we have that the unions up to N of $\mathcal{C}_1$ and $\mathcal{C}_2$ agree under $\mathcal{M}$ by compositionality. Hence $x \in \mathcal{M}(P_2)$, so $\mathcal{M}(P_1) \subseteq \mathcal{M}(P_2)$. Similarly $\mathcal{M}(P_2) \subseteq \mathcal{M}(P_1)$. □

Definition 7.15. A pre-model $\mathcal{M}$ is *extensional* if for all alphabets $\Sigma_1 \subseteq \Sigma_2$ we have that $\mathrm{obs}_{\Sigma_1}(\mathcal{M}) \subseteq \mathrm{obs}_{\Sigma_2}(\mathcal{M})$, and $\mathcal{M}_{\Sigma_2}$ agrees with $\mathcal{M}_{\Sigma_1}$ on $\mathcal{FL}(\Sigma_1) \times \mathrm{obs}_{\Sigma_1}(\mathcal{M})$.

Definition 7.16. A pre-model is a *model* if it is compositional and extensional.

In this setting, we now describe the five main finite observational models coarser than $\mathcal{FL}$: traces, failures, revivals, acceptances and refusal testing.

The traces model

The coarsest model measures only the *traces* of a process; that is, the sequences of events it is able to accept. This corresponds to the language of the process viewed as a nondeterministic finite automaton (NFA).

Definition 7.17. The *traces* model, $\mathcal{T}$, is given by

$$\mathrm{obs}_\Sigma(\mathcal{T}) = \Sigma^*, \ \mathcal{T}_\Sigma = \mathit{trace}_\Sigma$$

where *trace* is the equivalence relation which relates the observation $\langle A_0, a_1, A_1, \ldots, a_n, A_n \rangle$ to the string $a_1 \ldots a_n$.

Failures

The traces model gives us information about what a process is *allowed* to do, but it in some sense tells us nothing about what it is *required* to do. In particular, the process *STOP* trace-refines any other process.

In order to specify liveness properties, we can incorporate some information about the events the process is allowed to refuse, begining with the *failures* model. Intuitively, this captures traces s, together with the sets of events the process is allowed to stably refuse after s.

Definition 7.18. The *failures* model, $\mathcal{F}$, is given by

$$\mathrm{obs}_\Sigma(\mathcal{F}) = \Sigma^* \times (\mathcal{P}(\Sigma) \cup \{\bullet\}),\ \mathcal{F}_\Sigma = \mathit{fail}_\Sigma,$$

where fail_Σ relates the observation $\langle A_0, \ldots, a_n, A_n\rangle$ to all pairs $(a_1 \ldots a_n, X)$, for all $X \subseteq \Sigma \setminus A_n$ if $A_n \neq \bullet$, and for $X = \bullet$ otherwise.

Revivals

The next coarsest model, first introduced in [53], is the *revivals* model. Intuitively this captures traces s, together with sets X that can be stably refused after s, and events a (if any) that can then be accepted.

Definition 7.19. The *revivals* model, $\mathcal{R}$, is given by

$$\mathrm{obs}_\Sigma(\mathcal{R}) = \Sigma^* \times (\mathcal{P}(\Sigma) \cup \{\bullet\}) \times (\Sigma \cup \{\bullet\}),\ \mathcal{R}_\Sigma = \mathit{rev}_\Sigma,$$

where rev_Σ relates the observation $\langle A_0, a_1, \ldots, a_{n-1}, A_{n-1}, a_n, A_n\rangle$ to

(i) the triples $(a_1 \ldots a_{n-1}, X, a_n)$, for all $X \subseteq \Sigma \setminus A_{n-1}$ if $A_{n-1} \neq \bullet$ and for $X = \bullet$ otherwise, and

(ii) the triples $(a_1 \ldots a_n, X, \bullet)$, for all $X \subseteq \Sigma \setminus A_n$ if $A_n \neq \bullet$ and for $X = \bullet$ otherwise.

A finite linear observation is related to all triples consisting of: its initial trace; a stable refusal that could have been observed, or $\bullet$ if the original observation did not observe stability; and optionally (part (i) above) a single further event that can be accepted.

Acceptances

All the models considered up to now refer only to sets of refusals, which in particular are closed under subsets. The next model, *acceptances* (also known as 'ready sets'), refines the previous three and also considers the precise sets of events that can be stably accepted at the ends of traces.

Definition 7.20. The *acceptances* model, $\mathcal{A}$, is given by

$$\mathrm{obs}_\Sigma(\mathcal{A}) = \Sigma^* \times (\mathcal{P}(\Sigma) \cup \{\bullet\}),\ \mathcal{A}_\Sigma = \mathit{acc}_\Sigma,$$

where acc_Σ relates the observation $\langle A_0, a_1, \ldots, a_n, A_n\rangle$ to the pair $(a_1 \ldots a_n, A_n)$.

Refusal testing

The final model we consider is that of *refusal testing*, first introduced in [51]. This refines $\mathcal{F}$ and $\mathcal{R}$ by considering an entire history of events and stable refusal sets. It is incomparable to $\mathcal{A}$, because it does not capture precise acceptance sets.

Definition 7.21. The *refusal testing* model, $\mathcal{RT}$, is given by

$$\mathrm{obs}_\Sigma(\mathcal{RT}) = \{\langle X_0, a_1, X_1, \ldots, a_n, X_n\rangle : n \in \mathbb{N}, a_i \in \Sigma, X_i \subseteq \Sigma \text{ or } X_i = \bullet\}$$

$$\mathcal{RT}_\Sigma = rt_\Sigma,$$

where rt_Σ relates the observation $\langle A_0, \ldots, a_n, A_n\rangle$ to $\langle X_0, \ldots, a_n, X_n\rangle$, for all $X_i \subseteq \Sigma \setminus A_i$ if $A_i \neq \bullet$, and for $X_i = \bullet$ otherwise.

7.6.3 Rational models

We will later on wish to consider only models $\mathcal{M}$ for which the correspondence between $\mathcal{FL}$-observations and $\mathcal{M}$ observations is decidable by a finite memory computer. We will interpret this notion as saying the the relation $\mathcal{M}_\Sigma$ corresponds to the language accepted by some finite state automaton. In order to do this, we must first decide how to convert elements of $\mathcal{FL}_\Sigma$ to words in a language. We do this in the obvious way (the reasons for using fresh variables to represent the A_i will become apparent in Section 7.7).

Definition 7.22. The *canonical encoding* of $\mathcal{FL}_\Sigma$ is over the alphabet $\Xi := \Sigma \cup \Sigma'' \cup Sym$, where $\Sigma'' := \{a'' : a \in \Sigma\}$ and $Sym = \{\langle, \rangle, `, ', \bullet\}$.[2] It is given by the representation in Definition 7.10, where sets A_i are expressed by listing the elements of Σ'' corresponding to the members of A_i in alphabetical order. We denote this encoding by $\phi_\Sigma : \mathcal{FL}_\Sigma \to \Xi^*$.

We now define a model to be *rational* (borrowing a term from automata theory) if its defining relation can be recognised (when suitably encoded) by some nondeterministic finite automaton.

Definition 7.23. A model $\mathcal{M}$ is *rational* if for every alphabet Σ, there is some finite alphabet Θ and a map $\psi_\Sigma : \mathrm{obs}_\Sigma(\mathcal{M}) \to \Theta^*$, such that there is a (nondeterministic) finite automaton $\mathcal{A}$ recognising $\{(\phi_\Sigma(x), \psi_\Sigma(y)) : (x, y) \in \mathcal{M}_\Sigma\}$, and such that ψ_Σ is *order-reflecting* (that is, $\psi_\Sigma(x) \leq \psi_\Sigma(y)$ only if $x \leq y$), with respect to the prefix partial order on Θ^*, and the partial order induced by $\mathcal{M}_\Sigma$ on $\mathrm{obs}_\Sigma(\mathcal{M})$.

[2]Note that this somewhat unsatisfactory notation denotes a set of four elements: the angle brackets $\langle$ and $\rangle$, the comma , and the symbol $\bullet$.

What does it mean for an automaton to 'recognise' a relation?

Definition 7.24. For alphabets Σ and T, a relation $\mathcal{R} \subseteq \Sigma^* \times T^*$ is *recognised* by an automaton $\mathcal{A}$ just when:

(i) The event-set of $\mathcal{A}$ is $\mathrm{left}.\Sigma \cup \mathrm{right}.T$, and

(ii) For any $s \in \Sigma^*, t \in T^*$, we have $s\mathcal{R}t$ if and only if there is some interleaving of $\mathrm{left}.s$ and $\mathrm{right}.t$ accepted by $\mathcal{A}$.

Note that recognisability in the sense of Definition 7.24 is easily shown to be equivalent to the common notion of recognisability by a *finite state transducer* given for instance in [61], but the above definition is more convenient for our purposes. Note also that $\mathcal{FL}$ itself (viewing $\mathcal{FL}_\Sigma$ as the diagonal relation) is trivially rational.

Lemma 7.25. *The models $\mathcal{T}, \mathcal{F}, \mathcal{R}, \mathcal{A}$ and $\mathcal{RT}$ are rational.*

Proof. By inspection of Definitions 7.17–7.21. We take $\Theta = \Sigma \cup \Sigma' \cup \Sigma'' \cup Sym$, with Σ'' and the expression of acceptance sets as in the canonical encoding of $\mathcal{FL}$, and refusal sets expressed in the corresponding way over $\Sigma' := \{a' : a \in \Sigma\}$. □

Note that not all relations are rational. For instance, the 'counting relation' mapping each finite linear observation to its length is clearly not rational. We do not know whether the additional constraint of being a finite observational model necessarily implies rationality; however, no irrational models are known. We therefore venture the following conjecture:

Conjecture 7.26 (Rationality of finite observational models)**.** *Let $\mathcal{M}$ be a finite observational model. Then $\mathcal{M}$ is rational.*

7.7 Model shifting

We now come to the main substance of this chapter: we prove results on 'model shifting', showing that there exist contexts allowing us to pass between different semantic models and the basic traces model. The main result is Theorem 7.30, which shows that this is possible for any rational model.

7.7.1 Model shifting for $\mathcal{FL}$

We begin by proving the result for the finest model, $\mathcal{FL}$. We show that there exists a context $\mathcal{C}_{\mathcal{FL}}$ such that for any process P, the finite linear observations of P correspond to the traces of $\mathcal{C}_{\mathcal{FL}}(P)$.

Lemma 7.27 (Model shifting for $\mathcal{FL}$)**.** *For every alphabet Σ, there exists a context $\mathcal{C}_{\mathcal{FL}}$ over alphabet $T := \Sigma \cup \Sigma' \cup \Sigma'' \cup \{done\}$, and an order-reflecting map $\pi : \mathcal{FL}_\Sigma \to T^*$ (with respect to the extension partial order on $\mathcal{FL}_\Sigma$ and the prefix partial order on T^*) such that for any process P over Σ we have $\mathcal{T}(\mathcal{C}_{\mathcal{FL}}[P]) = \text{pref}(\pi(\mathcal{FL}(P)))$ (where $\text{pref}(X)$ is the prefix-closure of the set X).*

Proof. We will use the unprimed alphabet Σ to denote communicated events from the original trace, and the double-primed alphabet Σ'' to denote stable acceptances. Σ' will be used in an intermediate step to denote refusals, and *done* will be used to distinguish $\emptyset$ (representing an empty acceptance set) from $\bullet$ (representing a failure to observe anything).

Step 1: We first produce a process which is able to communicate events x'_i, just when the original process can stably refuse the corresponding x_i. Define the partial order $\leq_1 = \langle x' <_1 x : x \in \Sigma \rangle$, which prevents refusal events when the corresponding event can occur.

Let the context $\mathcal{C}_1$ be given by

$$\mathcal{C}_1[X] = \text{prioritise}(X \;|||\; RUN_{\Sigma'}, \leq_1, \Sigma).$$

Note that the third argument prevents primed events from occurring in unstable states.

Step 2: We now similarly introduce acceptance events, which can happen in stable states when the corresponding refusal can't.

Similarly define the partial order $\leq_2 = \langle x'' <_2 x' : x \in \Sigma \rangle$, which prevents acceptance events when the corresponding refusal is possible. Let the context $\mathcal{C}_2$ be defined by

$$\mathcal{C}_2[X] = \text{prioritise}(\mathcal{C}_1[X] \;|||\; RUN_{\Sigma''}, \leq_2, \Sigma).$$

Step 3: We now ensure that an acceptance set inferred from a trace is a complete set accepted by the process under examination. This is most straightforwardly done by employing a regulator process, which can either accept an unprimed event or accept the alphabetically first refusal or acceptance event, followed by a refusal or

acceptance for each event in turn. In the latter case it then communicates a *done* event, and returns to its original state.

The *done* event is necessary in order to distinguish between a terminal $\emptyset$, which can have a *done* after the last event, and a terminal $\bullet$, which cannot (observe that a $\emptyset$ cannot occur other than at the end). Finally, we hide the refusal events.

Let a and z denote the alphabetically first and last events respectively, and let succ x denote the alphabetical successor of x. Define the processes

$$\begin{aligned}
UNSTABLE &= \Box_{x\in\Sigma}\, x \to UNSTABLE \\
&\qquad \Box\, a' \to STABLE(a) \,\Box\, a'' \to STABLE(a) \\
STABLE(x) &= x' \to STABLE(\text{succ } x) \,\Box\, x'' \to STABLE(\text{succ } x) \qquad (x \neq z) \\
STABLE(z) &= done \to UNSTABLE,
\end{aligned}$$

and let

$$\mathcal{C}_{\mathcal{FL}}[X] = \left(\mathcal{C}_2[X] \underset{\Sigma\cup\Sigma'\cup\Sigma''}{\|} UNSTABLE\right) \setminus \Sigma'.$$

Step 4: We now complete the proof by defining the function π inductively as follows:

$$\begin{aligned}
\pi(s\hat{}\langle\bullet\rangle) &= \pi(s) \\
\pi(s\hat{}\langle x\rangle) &= \pi(s)\hat{}\langle x\rangle \\
\pi(s\hat{}\langle A = \{x_1,\ldots,x_k\}\rangle) &= \pi(s)\hat{}\langle x_1''\ldots x_k'' done\rangle,
\end{aligned}$$

where without loss of generality the x_i are listed in alphabetical order.

It is clear that this is order-reflecting, and by the construction above satisfies $\mathcal{T}(\mathcal{C}_{\mathcal{FL}}[P]) = \text{pref}(\pi(\mathcal{FL}(P)))$. □

This result allows us to translate questions of $\mathcal{FL}$-refinement into questions of trace refinement under $\mathcal{C}_{\mathcal{FL}}$, as follows:

Corollary 7.28. *For $\mathcal{C}_{\mathcal{FL}}$ as in Lemma 7.27, and for any processes P and Q, we have $P \sqsubseteq_{\text{FL}} Q$ if and only if $\mathcal{C}_{\mathcal{FL}}[P] \sqsubseteq_{\text{T}} \mathcal{C}_{\mathcal{FL}}[Q]$.*

Proof. Certainly if $\mathcal{FL}(Q) \subseteq \mathcal{FL}(P)$ then $\mathcal{T}(\mathcal{C}_{\mathcal{FL}}[Q]) = \text{pref}(\pi(\mathcal{FL}(Q))) \subseteq \text{pref}(\pi(\mathcal{FL}(P))) = \mathcal{T}(\mathcal{C}_{\mathcal{FL}}[P])$ and so $\mathcal{C}_{\mathcal{FL}}[P] \sqsubseteq_{\text{T}} \mathcal{C}_{\mathcal{FL}}[Q]$.

Conversely, suppose there exists $x \in \mathcal{FL}(Q) \setminus \mathcal{FL}(P)$. Then since $\mathcal{FL}(P)$ is downwards-closed, we have $x \not\leq y$ for all $y \in \mathcal{FL}(P)$. Since π is order-reflecting, we have correspondingly $\pi(x) \not\leq \pi(y)$ for all $y \in \mathcal{FL}(P)$. Hence $\pi(x) \notin \text{pref}(\pi(\mathcal{FL}(P)))$, so $\text{pref}(\pi(\mathcal{FL}(Q))) \not\subseteq \text{pref}(\pi(\mathcal{FL}(P)))$. □

7.7.2 Model shifting for rational observational models

We now have essentially all we need to prove the main theorem. We record a folk result, that any NFA can be implemented as a CSP process (up to prefix-closure, since trace-sets are prefix-closed but regular languages are not):

Lemma 7.29 (Implementation for NFA)**.** *Let $\mathcal{A} = (\Sigma, Q, \delta, q_0, F)$ be a (nondeterministic) finite automaton. Then there exists a CSP process $P_{\mathcal{A}}$ such that* $\mathrm{pref}(L(\mathcal{A})) = \mathrm{pref}(\mathcal{T}(P_{\mathcal{A}}))$.

Proof. Trivial construction. See Chapter 7 of [52]. □

Theorem 7.30 (Model shifting for rational models)**.** *For every rational model $\mathcal{M}$, there exists a context $\mathcal{C}_{\mathcal{M}}$ such that for any process P we have $\mathcal{T}(\mathcal{C}_{\mathcal{M}}[P]) = \mathrm{pref}(\psi(\mathcal{M}(P)))$.*

Proof. Let $\mathcal{A}$ be the automaton recognising $(\phi \times \psi)(\mathcal{M})$ (as from Definition 7.23), and let $P_{\mathcal{A}}$ be the corresponding process from Lemma 7.29.

We first apply Lemma 7.27 to produce a process whose traces correspond to the finite linear observations of the original process, prefixed with left: let $\mathcal{C}_{\mathcal{FL}}$ be the context from Lemma 7.27, and let the context $\mathcal{C}_1$ be defined by

$$\mathcal{C}_1[X] = \mathcal{C}_{\mathcal{FL}}[X][\![^{\mathrm{left}.x}/_x]\!].$$

We now compose in parallel with $P_{\mathcal{A}}$, to produde a process whose traces correspond to the $\mathcal{M}$-observations of the original process. Let $\mathcal{C}_2$ be defined by

$$\mathcal{C}_2[X] = \left(\left(\mathcal{C}_1[X] \underset{\{|\mathrm{left}|\}}{\|} P_{\mathcal{A}}\right) \setminus \{|\mathrm{left}|\}\right)[\![^{x}/_{\mathrm{right}.x}]\!].$$

Then the traces of $\mathcal{C}_2[X]$ are precisely the prefixes of the images under ψ of the observations corresponding to X, as required. □

By the same argument as for Corollary 7.28, we have

Corollary 7.31. *For any rational model $\mathcal{M}$, let $\mathcal{C}_{\mathcal{M}}$ be as in Theorem 7.30. Then for any processes P and Q, we have $P \sqsubseteq_M Q$ if and only if $\mathcal{C}_{\mathcal{M}}[P] \sqsubseteq_{\mathrm{T}} \mathcal{C}_{\mathcal{M}}[Q]$.*

7.8 Implementation

We demonstrate the technique by implementing contexts with the property of Corollary 7.31; source code may be found at [1].

For the sake of efficiency we work directly rather than using the general construction of Theorem 7.30. The context `C1` introduces refusal events and a `stab` event, which can occur only when the corresponding normal events can be refused. This implements the refusal testing model, and the context `CF` which allows only normal events optionally followed by some refusals (and `stab`) implements the failures model.

This is however suboptimal over large alphabets, in the typical situation where most events are refused most of the time. FDR3's inbuilt failures refinement checking is able to compare *acceptance* sets (checking that the acceptances of the specification are a subset of those of the implementation), which are typically smaller than the refusal sets.

The context `C'` introduces acceptance events which can occur only in stable states where the corresponding refusal cannot, and then blocks all refusals. The problem then is: how to check that the acceptances of the *specification* are a subset of those of the *implementation*, despite the fact that trace refinement checks for inclusion the other way?

The answer is to use priority to prevent the `stab` event from happening while acceptances are still available, so that `CFImpl'` is able to communicate only its *precise* acceptance sets. We then form `CFSpec'` by parallel composition with RUN for all the acceptance events, so that `CFSpec'` can communicate any *supersets* of its acceptance set.

Similar constructions with slightly different restrictions on the permissible sequences of events produce efficient processes for the revivals and refusal testing models. For the acceptances model, we just want to check for inclusion of the implementation's acceptance sets in those of the specification, so the context `CFImpl'` works for both the specification and the implementation; finite linear observations works similarly with failures replaced by refusal testing.

7.8.1 Testing

We test this implementation by constructing processes which are first distinguished by the failures, revivals, refusal testing and acceptance models respectively (the latter two being also distinguished by the finite linear observations model). The processes, and the models which do and do not distinguish them, are shown in Table 7.1 (recall

the precision hierarchy of models: $\mathcal{T} \leq \mathcal{F} \leq \mathcal{R} \leq \{\mathcal{A}, \mathcal{RT}\} \leq \mathcal{FL}$). The correct results are obtained when these checks are run in FDR3 with the implementation described above.

Specification	Implementation	Passes	Fails
$a \rightarrow \mathbf{div}$	$a \rightarrow STOP$	$\mathcal{T}$	$\mathcal{F}$
$((a \rightarrow \mathbf{div}) \,\Box\, \mathbf{div}) \sqcap STOP$	$a \rightarrow \mathbf{div}$	$\mathcal{F}$	$\mathcal{R}$
$(a \rightarrow \mathbf{div}) \sqcap (\mathbf{div} \triangle (a \rightarrow STOP))$	$a \rightarrow STOP$	$\mathcal{R}, \mathcal{A}$	$\mathcal{RT}, \mathcal{FL}$
$(a \rightarrow STOP) \sqcap (b \rightarrow STOP)$	$(a \rightarrow STOP) \,\Box\, (b \rightarrow STOP)$	$\mathcal{R}, \mathcal{RT}$	$\mathcal{A}, \mathcal{FL}$

Table 7.1: Tests distinguishing levels of the model precision heirachy. $\triangle$ is the *interrupt* operator; see [54] for details.

7.8.2 Performance

We assess the performance of our simulation by running those examples from Table 1 of [28] which involve refinement checks (as opposed to deadlock- or divergence-freedom assertions), and comparing the timings for our construction against the time taken by FDR3's inbuilt failures refinement check (since $\mathcal{F}$ is the only model for which we have a point of comparison between a direct implementation and the methods developed in this chapter). Results are shown in Table 7.2, for both the original and revised contexts described above; the perfomance of the $\mathcal{FL}$ check is also shown. As may be seen, performance is somewhat worse but not catastrophically so. Note however that these processes involve rather small alphabets; performance is expected to be worse for larger alphabets.

7.8.3 Example: Conflict detection

We illustrate the usefulness of richer semantic models than just traces and failures by giving a sample application of the revivals model. Suppose that we have a process P consisting of the parallel composition of two sub-processes Q and R. The failures model is able to detect when P can refuse all the events of their shared alphabet, or deadlock in the case when they are synchronised on the whole alphabet. However, it is unable to distinguish between the two possible causes of this: it may be that one of the composands is able to refuse the entire shared alphabet, or it may be that each accepts some events from the shared alphabet, but the acceptances of Q and R are disjoint. We refer to the latter situation as a 'conflict'. The absence of conflict (and similar situations) is at the core of a number of useful ways of proving deadlock-freedom for networks of processes running in parallel [56].

	Inbuilt $\mathcal{F}$			`CF`			`CF'`			`FL`		
Input File	$\|S\|$	$\|\Delta\|$	$T(s)$	$\|S\|$	$\|\Delta\|$	$T(s)$	$\|S\|$	$\|\Delta\|$	$T(s)$	$\|S\|$	$\|\Delta\|$	$T(s)$
`inv`	21M	220M	23	21M	220M	78	21M	220M	125	21M	220M	145
`nspk`	6.9M	121M	22	6.3M	114M	73	4.1M	72M	55	5.4M	97M	92
`swp`	24M	57M	16	30M	123M	61	43M	76M	107	42M	93M	131

Table 7.2: Experimental results comparing the performance of our construction with FDR3's inbuilt failures refinement check. $|S|$ is the number of states, $|\Delta|$ is the number of transitions, T is the time (in seconds), and M indicates millions.

The revivals model can be used to detect conflicts. For a process $P = Q \ {}_X\|_Y \ R$, we introduce a fresh event a to represent a generic event from the shared alphabet, and form the process $P' = Q' \ {}_{X'}\|_{Y'} \ R'$, where $Q' = Q[\![\{(x,x),(x,a) : x \in X\}]\!]$, $X' = X \cup \{a\}$, and similarly for R' and Y'. Conflicts of P now correspond to revivals $(s, X \cap Y, a)$, where s is a trace not containing a.

7.9 Information flow

How do we use this to define and calculate quantified information flow? We will consider information flow in the $\mathcal{FL}$ model, where we assume that the attacker is able to see the precise acceptance sets at each step.

We must decide on a model of how the experimenter interacts with the system. The obvious choice is to say that at each step the system presents the experimenter with an acceptance set, and the experimenter chooses a single event to occur. We are therefore essentially interested in computing the antichain growth of the language

$$\mathcal{T}(\mathcal{C}_{\mathcal{FL}}[\mathcal{L}_H(P)]),$$

where actions chosen by the experimenter are comparable (so Σ is linearly ordered), but acceptances offered by the system are incomparable (so $\Sigma'' \cup \{done\}$ is given the discrete order). However, we should measure 'length' not as the length of the word in $(\Sigma \cup \Sigma'' \cup \{done\})^*$, but rather just as the number of Σ-events that have occurred.

One decision we have to make is how to handle the possibility of actions occuring in unstable states, or before stability has been observed. One option would be simply to prohibit these, but this is not satisfactory because we should be able to model systems involving timeouts. The most significant question is whether we should consider information flow to result from the situation where the system is able to choose between unstably offering a or unstably offering b. In the context of complete noninterference this problem does not arise because if a can be unstably offered then it can never be stably refused and so if the system is deterministic then the state where b was offered must resolve (since the system is divergence-free) to a state stably accepting both a and b.

We choose to resolve this problem by saying that the system (rather than the experimenter) makes the choice as to whether the experimenter is able to offer an event before stability (so Σ and $\Sigma'' \cup \{done\}$ are incomparable), but the experimenter is able to offer only a single event in such a case, so $s\hat{}\langle xa\rangle$ and $s\hat{}\langle xb\rangle$ are not consistent

with a single experimental strategy for any $a \neq b, x \in \Sigma$. This yields the following definition of information flow.

Definition 7.32. Let P be a process. The $\mathcal{FL}$-information flow permitted by P in time k, denoted I_k, is given by

$$I_k = \max_{X \in \mathcal{F}_k} \log |X|,$$

where $\mathcal{F}_k \subseteq \mathcal{P}((\Sigma \cup \Sigma'' \cup \{done\})^*)$ is defined by $X \in \mathcal{F}_k$ if and only if

(i) for all $s \in X$ we have length $(s|_\Sigma) \leq k$, and

(ii) X is an antichain with respect to the lexicographic partial order induced by setting Σ to be linearly ordered, and otherwise setting all the elements of $\Sigma \cup \Sigma'' \cup \{done\}$ to be incomparable.

For fixed Σ we have that

$$\frac{1}{2|\Sigma| + 1}\text{length}(s) \leq \text{length}\,(s|_\Sigma) \leq \text{length}(s),$$

and so I_k has linear (respectively logarithmic) growth if and only if $\mathcal{T}(\mathcal{C}_{\mathcal{FL}}[\mathcal{L}_H(P)])$ has exponential (respectively polynomial) antichain growth.

Proposition 7.33. *Let P be a process. The $\mathcal{FL}$-information flow I_k permitted by P has logarithmic growth if and only if the language*

$$\mathcal{T}(\mathcal{C}_{\mathcal{FL}}[\mathcal{L}_H(P)])$$

has polynomial antichain growth with respect to the partial order induced by setting Σ to be linearly ordered, and otherwise setting all the elements of $\Sigma \cup \Sigma'' \cup \{done\}$ to be incomparable. Otherwise I_k has linear growth.

The results of the previous chapter show that logarithmic and linear information flow are the only possibilities, and that there is a polynomial time algorithm to distinguish the two cases.

7.10 Conclusions

The result of Theorem 7.30 shows that the expressibility of all finite observational (rational) models can in some sense be simulated by the traces model using the priority operator. This provides a practical method of testing refinement over models

that FDR does not directly support. While any such model could be implemented directly in the program itself, we have shown this is not necessary. This also serves to further demonstrate the power and usefulness of the priority operator (see also the previous work of Roscoe on the expressiveness of CSP with priority [55] and on 'slow abstraction' [57]).

Note that this type of construction can be used more generally. Firstly, it seems likely that the construction can be extended to non-finite models; for instance to reduce failures-divergences tests to traces-divergences, or infinite-traces-failures-divergences to infinite-traces-divergences.

Secondly, the construction does not use the requirement that a model be compositional. This means that it will work for any rational set of observable behaviours, such as the singleton failures semantics presented in [7]. The techniques described here can also be used to support the Timed Failures model of Timed CSP in FDR3 [6].

The limitation to rational models is from a theoretical point of view rather unsatisfactory, although it may be of little practical significance since all known models (and probably all models one would be likely to come up with) are clearly rational. However, Conjecture 7.26 remains of interest since a resolution in either direction would undoubtedly yield insight into the structure of the 'clouds' of models lying above $\mathcal{R}$ set out in [53].

We have shown how to use this technique in combination with the results of Chapter 6 to quantify information flow in the finite linear observations model. The question of how to define quantified information flow in other models is more subtle: for instance, in the refusal testing model, the experimenter can be shown any subset of the complement of the true acceptance set, but presumably the choice of which set she is shown should not be conisdered to give rise to information flow. In some sense we believe that this reflects the fact that there is no clear model of interaction tightly associated with these models, and so it is not obvious that the question is of great practical significance.

Chapter 8

Conclusions and future work

8.1 Conclusions

We believe that we have accomplished the main goal of this thesis: we have shown how to define information flow for interactive deterministic systems, and how to characterise it as essentially 'safe' or 'dangerous'. A number of questions remain for future research. As we have mentioned above, there is still some question regarding the appropriate information-theoretic measure to use for these purposes. We believe that we have made a contribution to this question but it is far from settled.

We have shown that in the case of polynomial antichain growth there is a polynomial time algorithm to calculate the precise order of that growth, but the corresponding question for exponential growth is wide open. It is remarkable that so far as we can tell there is no known efficient algorithm even to calculate the order of exponential *language* growth for an NFA, and we hope that at least this easier question may be resolved in the future.

Two very substantial issues that are not covered by our work are the problems of calculating information flow for probabilistic systems, and that of calculating information flow for systems with multiple users (which we will remark also encompasses the case of nondeterministic systems). We discuss these two questions in the following sections.

8.2 Probabilistic systems

Throughout Chapter 5 we have considered only deterministic systems. A natural topic for further work is to deal with probabilistic systems. Just as we have modelleded determinsitic systems as deterministic finite state transducers, the obvious choice to model probabilistic systems is as *probabilistic finite state transducers* (also known

as stochastic FSTs). The definition below is slightly less general than the standard definition, but is convenient for our purposes.

Definition 8.1. A *probabilistic finite state transducer* is a tuple $T = (Q, q_0, \Sigma, \Gamma, \Delta)$, where Q is a finite set of *states*, $q_0 \in Q$ is the *initial state* and $\Delta : Q \times \Sigma \times \Gamma \times Q \to \mathbb{R}^{\geq 0}$ is the *transition function*, such that for all $q \in Q$ and all $a \in \Sigma$ we have

$$\sum_{b \in \Gamma, q' \in Q} \Delta(q, a, b, q') = 1.$$

We interpret $\Delta(q, a, b, q')$ as the probability that on recieving the input a in state q, the system outputs b and moves to state q'.

As before we require that Σ and Γ are of the form $\Sigma_A \times \Sigma_B$ and $\Gamma_A \times \Gamma_B$ respectively, where Σ_A and Γ_A are the input and output alphabets for Alice, and Σ_B and Γ_B are the input and output alphabets for Bob. As before a strategy for Alice is a function $x_A : (\Sigma_A \times \Gamma_A)^* \to \Sigma_A$ and a strategy for Bob is a funtion $x_B : (\Sigma_B \times \Gamma_B)^* \to \Sigma_B$ (recalling that we may assume that any randomness in the strategies for Alice and Bob is determined at the start, so that they are making a probabilistic choice among deterministic strategies).

For fixed x_A and x_B, the output Y produced to Bob after n steps is a sequence $y \in (\Sigma_B \times \Gamma_B)^n$, where $y = ((a_1, b_1), \ldots, (a_n, b_n))$ occurs with probability

$$\sum_{q_1, \ldots, q_n \in Q} \prod_{i=1}^{n} \Delta(q_{i-1}, a_i, b_i, q_i).$$

We are now able to apply the theory of Chapter 4 to define information flow with respect to our chosen measure of information (either mutual information, or $I_{\mathrm{LSE}}, I'_{\mathrm{LSE}}$ or I''_{LSE} as defined in Chapter 4). By Proposition 4.7 and Proposition 4.22 we have that information flow will be maximised by a pure strategy for Bob. On the other hand, we certainly do not have an equivalent to Theorem 4.12 (extended to $I_{\mathrm{LSE}}, I'_{\mathrm{LSE}}, I''_{\mathrm{LSE}}$ by Theorem 4.24), which says that Alice will maximise information flow to Bob by maximising the size of the set of possible outputs he can see, and making these outputs equiprobable.

Computing the value of any of these quantities remains a challenge for future research; we suggest that a reasonable starting point might be [18], which studies the problem of computing the relative entropy of probabilistic automata.

8.3 Multi-agent systems

In Chapters 4–7 we have considered systems with only two agents, Alice and Bob. In general, however, there may be more than this. Some situations can straightforwardly be modeled by the Alice-Bob case: for instance, if all but only one agent is considered to be under the control of the attacker then then that single agent is represented by Alice and all the rest by Bob. On the other hand, if the secret is shared by all but a single compromised agent then that single agent is Bob and the rest taken together are Alice.[1]

However, in general there may exist agents which are not under the control of the attacker, and whose actions are not visible to her, but whose behaviour is not directly influenced by the value of the key. This is modelled by introducing a further random variable X_C, representing the strategies chosen by the 'neutral' agents. The system is represented by a fixed matrix of conditional probabilities $p_{Y|X_A,X_B,X_C}(y|x_A,x_B,x_C)$, and the total information flow is the maximum value of the conditional mutual information $I(Y\,;X_A|X_B)$, taken over marginal probability distributions p_{X_A}, p_{X_B} and p_{X_C}, that is

$$I_M = \max_{p_{X_A},p_{X_B},p_{X_C}} I(Y\,;X_A|X_B).$$

We have an equivalent to Proposition 4.7, showing that we may also assume that X_C is a pure strategy.

Proposition 8.2. *Let X_A, X_B, X_C and Y be random variables, and let the conditional distribution function $p_{Y|X_A,X_B,X_C}$ be fixed. Then*

$$\max_{p_{X_A},p_{X_B},p_{X_C}} I(Y\,;X_A|X_B) = \max_{x_B\in\mathcal{X}_B,x_C\in\mathcal{X}_C} \max_{p_{X_A}} I(Y\,;X_A|X_B=x_B,X_C=x_C).$$

Proof. We have $I(Y;X_A|X_B) = H(X_A|X_B) - H(X_A|X_B,Y) = H(X_A) - H(X_A|X_B,Y)$, so it suffices to prove that $H(X_A|X_B,Y)$ is minimised for point values of x_B, x_C.

We have $H(X_A|X_B,Y) \geq H(X_A|X_B,X_C,Y)$ by Theorem 2.6.5 of [19] ('Information can't hurt'). But we have

$$\begin{aligned} H(X_A|X_B,X_C,Y) &= \sum_{x_C\in\mathcal{X}_C} p_{X_C}(x_C) H(X_A|X_B,Y,X_C=x_C) \\ &\geq \min_{x_C\in\mathcal{X}_C} H(X_A|X_B,Y,X_C=x_C). \end{aligned}$$

[1] More generally, we can handle any situation in which each agent is either in possession of the secret or is under the control of the attacker

Hence

$$\max_{p_{X_A}, p_{X_B}, p_{X_C}} I(Y\,; X_A | X_B) = \max_{x_C \in \mathcal{X}_C} \max_{p_{X_A}, p_{X_B}} I(Y\,; X_A | X_B, X_C = x_C),$$

and applying Proposition 4.7 yields the result. □

It is clear how to extend the theory of Chapter 5 to model systems with 'neutral' agents: just extend the input and output languages of the transducer to $\Sigma_A \times \Sigma_B \times \Sigma_C$ and $\Gamma_A \times \Gamma_B \times \Gamma_C$ respectively, where Σ_C and Γ_C are the input and output events under the control of the neutral agents. Note that if there are several independent neutral agents then we will have to put further constraints on the set of admissible strategies for C, to say that the inputs from a particular agent can only depend on its outputs.

A question for future research is how to compute I_M for such a system.

Note that this also covers the case of nondeterministic systems, where we assume that the value of the secret cannot directly affect how the nondeterminism is resolved: we simply treat the nondeterminism as being under the control of a neutral agent.

Bibliography

[1] `www.cs.ox.ac.uk/people/david.mestel/model-shifting.csp`.

[2] M. S. Alvim, K. Chatzikokolakis, A. McIver, C. Morgan, C. Palamidessi, and G. Smith. Additive and multiplicative notions of leakage, and their capacities. In *2014 IEEE 27th Computer Security Foundations Symposium*, pages 308–322, July 2014.

[3] M. S. Alvim, K. Chatzikokolakis, C. Palamidessi, and G. Smith. Measuring information leakage using generalized gain functions. In *2012 IEEE 25th Computer Security Foundations Symposium*, pages 265–279, June 2012.

[4] Mário S. Alvim, Miguel E. Andrés, and Catuscia Palamidessi. Quantitative information flow in interactive systems. *J. Comput. Secur.*, 20(1):3–50, January 2012.

[5] Miguel E. Andrés, Catuscia Palamidessi, Peter van Rossum, and Geoffrey Smith. Computing the leakage of information-hiding systems. In Javier Esparza and Rupak Majumdar, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 373–389, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[6] Philip Armstrong, Gavin Lowe, Joël Ouaknine, and A.W. Roscoe. Model checking timed CSP. In Andrei Voronkov and Margarita Korovina, editors, *HOWARD-60. A Festschrift on the Occasion of Howard Barringer's 60th Birthday*, pages 13–33. EasyChair, 2014.

[7] Christie Bolton and Jim Davies. A singleton failures semantics for communicating sequential processes. *Formal Aspects of Computing*, 18(2):181–210, 2006.

[8] M. Boreale and Francesca Pampaloni. Quantitative information flow under generic leakage functions and adaptive adversaries. *Logical Methods in Computer Science*, Volume 11, Issue 4, November 2015.

[9] Michele Boreale, Francesca Pampaloni, and Michela Paolini. Asymptotic information leakage under one-try attacks. In *Proceedings of the 14th International Conference on Foundations of Software Science and Computational Structures: Part of the Joint European Conferences on Theory and Practice of Software*, FOSSACS'11/ETAPS'11, pages 396–410, Berlin, Heidelberg, 2011. Springer-Verlag.

[10] Michele Boreale and Michela Paolini. On formally bounding information leakage by statistical estimation. In Sherman S. M. Chow, Jan Camenisch, Lucas C. K. Hui, and Siu Ming Yiu, editors, *Information Security*, pages 216–236, Cham, 2014. Springer International Publishing.

[11] Christelle Braun, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Quantitative notions of leakage for one-try attacks. *Electronic Notes in Theoretical Computer Science*, 249:75 – 91, 2009. Proceedings of the 25th Conference on Mathematical Foundations of Programming Semantics (MFPS 2009).

[12] Graham Brightwell. Random k-dimensional orders: Width and number of linear extensions. *Order*, 9(4):333–342, 1992.

[13] E. Rodney Canfield. On a problem of Rota. *Advances in Mathematics*, 29(1):1–10, 1978.

[14] Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Prakash Panangaden. Anonymity protocols as noisy channels. *Information and Computation*, 206(2):378 – 401, 2008. Joint Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis (FCS-ARSPA 06).

[15] David Clark and Sebastian Hunt. Formal aspects in security and trust. chapter Non-Interference for Deterministic Interactive Programs, pages 50–66. Springer-Verlag, Berlin, Heidelberg, 2009.

[16] David Clark, Sebastian Hunt, and Pasquale Malacaria. Quantified interference: information theory and information flow. In *Presented at Workshop on Issues in the Theory of Security (WITS04*, page 04, 2004.

[17] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: `http://www.grappa.univ-lille3.fr/tata`, 2007. release October, 12th 2007.

[18] Corinna Cortes, Mehryar Mohri, Ashish Rastogi, and Michael Riley. On the computation of the relative entropy of probabilistic automata. *Int. J. Found. Comput. Sci.*, 19:219–242, 02 2008.

[19] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., 2nd edition, 2005.

[20] Dorothy E. Denning. *Cryptography and Data Security*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1982.

[21] Dorothy E. Denning and Peter J. Denning. Certification of programs for secure information flow. *Commun. ACM*, 20(7):504–513, July 1977.

[22] Robert P. Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics*, pages 161–166, 1950.

[23] Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming*, pages 1–12, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[24] Riccardo Focardi. Comparing two information flow security properties. In *Computer Security Foundations Workshop, 1996. Proceedings., 9th IEEE*, pages 116–122. IEEE, 1996.

[25] Riccardo Focardi and Roberto Gorrieri. A classification of security properties for process algebras. *Journal of Computer security*, 3(1):5–33, 1995.

[26] Paweł Gawrychowski, Dalia Krieger, Narad Rampersad, and Jeffrey Shallit. Finding the growth rate of a regular of [sic] context-free language in polynomial time. In *Developments in Language Theory*, pages 339–358. Springer, 2008.

[27] Thomas Gibson-Robinson, Philip Armstrong, Alexandre Boulgakov, and A.W. Roscoe. FDR3—a modern refinement checker for CSP. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 187–201. Springer, 2014.

[28] Thomas Gibson-Robinson, Henri Hansen, A.W. Roscoe, and Xu Wang. Practical partial order reduction for CSP. In *NASA Formal Methods*, pages 188–203. Springer, 2015.

[29] Seymour Ginsburg. *The Mathematical Theory of Context Free Languages*. McGraw-Hill Book Company, 1966.

[30] Seymour Ginsburg and Edwin H. Spanier. Bounded algol-like languages. *Transactions of the American Mathematical Society*, 113(2):333–368, 1964.

[31] Joseph A. Goguen and José Meseguer. Security policies and security models. In *1982 IEEE Symposium on Security and Privacy*, pages 11–20. IEEE, 1982.

[32] James W. Gray, III. Toward a mathematical foundation for information flow security. *J. Comput. Secur.*, 1(3-4):255–294, May 1992.

[33] G. H. Hardy, J. E. Littlewood, and George Pólya. *Inequalities*. Cambridge university press, 1952.

[34] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1985.

[35] David Mestel (https://mathoverflow.net/users/69143/david). Estimating the growth rate of nondeterministic finite automata. MathOverflow. URL:https://mathoverflow.net/q/199797 (version: 2015-03-12).

[36] D. Kleitman, M. Edelberg, and D. Lubell. Maximal sized antichains in partial orders. *Discrete Mathematics*, 1(1):47 – 53, 1971.

[37] Paul Kocher, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks: Exploiting speculative execution. *CoRR*, abs/1801.01203, 2018.

[38] Boris Köpf and David Basin. An information-theoretic model for adaptive side-channel attacks. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, pages 286–296, New York, NY, USA, 2007. ACM.

[39] Dalia Krieger, Narad Rampersad, and Jeffrey Shallit. Finding the growth rate of a regular language in polynomial time. *CoRR*, abs/0711.4990, 2007.

[40] Leslie Lamport. What good is temporal logic? In R. E. A. Mason, editor, *Information Processing 83: Proceedings of the IFIP 9th World Congress*, September 1983.

[41] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown. *CoRR*, abs/1801.01207, 2018.

[42] Gavin Lowe. Quantifying information flow. In *Proceedings of the 15th IEEE Workshop on Computer Security Foundations*, CSFW '02, pages 18–31, Washington, DC, USA, 2002. IEEE Computer Society.

[43] Gavin Lowe. On information flow and refinement-closure. In *Proceedings of the Workshop on Issues in the Theory of Security (WITS07)*, 2007.

[44] George H. Mealy. A method for synthesizing sequential circuits. *Bell Labs Technical Journal*, 34(5):1045–1079, 1955.

[45] David Mestel. On the widths of regular and context free languages, with an application to information flow. *CoRR*, abs/1709.08696, 2017.

[46] David Mestel and A.W. Roscoe. Reducing complex csp models to traces via priority. *Electronic Notes in Theoretical Computer Science*, 325:237 – 252, 2016. The Thirty-second Conference on the Mathematical Foundations of Programming Semantics (MFPS XXXII).

[47] Robin Milner. *A Calculus of Communicating Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982.

[48] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, I. *Information and Computation*, 100(1):1–40, 1992.

[49] Ira S. Moskowitz, Richard E. Newman, Daniel P. Crepeau, and Allen R. Miller. Covert channels and anonymizing networks. In *Proceedings of the 2003 ACM Workshop on Privacy in the Electronic Society*, WPES '03, pages 79–88, New York, NY, USA, 2003. ACM.

[50] G. W. Peck. Maximum antichains of rectangular arrays. *Journal of Combinatorial Theory, Series A*, 27(3):397–400, 1979.

[51] Iain Phillips. Refusal testing. *Theoretical Computer Science*, 50(3):241–284, 1987.

[52] A. W. Roscoe. *The Theory and Practice of Concurrency*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1997.

[53] A.W. Roscoe. Revivals, stuckness and the hierarchy of CSP models. *The Journal of Logic and Algebraic Programming*, 78(3):163–190, 2009.

[54] A.W. Roscoe. *Understanding Concurrent Systems*. Texts in Computer Science. Springer, 2010.

[55] A.W. Roscoe. The expressiveness of CSP with priority. In *Proceedings of MFPS 2015*, 2015.

[56] A.W. Roscoe and Naiem Dathi. The pursuit of deadlock freedom. *Information and Computation*, 75(3):289 – 327, 1987.

[57] A.W. Roscoe and Philippa J. Hopcroft. Slow abstraction via priority. In Zhiming Liu, Jim Woodcock, and Huibiao Zhu, editors, *Theories of Programming and Formal Methods*, pages 326–345. Springer-Verlag, Berlin, Heidelberg, 2013.

[58] A.W. Roscoe, J.C.P. Woodcock, and L. Wulf. Non-interference through determinism. In *Computer Security—ESORICS 94*, pages 31–53. Springer, 1994.

[59] P. Y. A. Ryan, J. McLean, J. Millen, and V. Gligor. Non-interference, who needs it? In *Proceedings. 14th IEEE Computer Security Foundations Workshop, 2001.*, pages 237–238, 2001.

[60] P. Y. A. Ryan and S. A. Schneider. Process algebra and non-interference. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop*, pages 214–227, 1999.

[61] J. Shallit. *A Second Course in Formal Languages and Automata Theory*. Cambridge University Press, 2009.

[62] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, July 1948.

[63] Geoffrey Smith. On the foundations of quantitative information flow. In *Proceedings of the 12th International Conference on Foundations of Software Science and Computational Structures: Held As Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009*, FOSSACS '09, pages 288–302, Berlin, Heidelberg, 2009. Springer-Verlag.

[64] Emanuel Sperner. Ein satz über untermengen einer endlichen menge. *Mathematische Zeitschrift*, 27(1):544–548, 1928.

[65] Ron van der Meyden and Chenyi Zhang. Algorithmic verification of noninterference properties. *Electronic Notes in Theoretical Computer Science*, 168:61 – 75, 2007. Proceedings of the Second International Workshop on Views on Designing Complex Architectures (VODCA 2006).

[66] Ron van der Meyden and Chenyi Zhang. A comparison of semantic models for noninterference. *Theoretical Computer Science*, 411(47):4123 – 4147, 2010.

[67] Andreas Weber and Helmut Seidl. On the degree of ambiguity of finite automata. *Theor. Comput. Sci.*, 88(2):325–349, October 1991.

[68] Douglas B. West. Extremal problems in partially ordered sets. In Ivan Rival, editor, *Ordered Sets: Proceedings of the NATO Advanced Study Institute held at Banff, Canada, August 28 to September 12, 1981*, pages 473–521. Springer Netherlands, Dordrecht, 1982.