RATIONAL INATTENTION AND A CAUSAL ACCOUNT OF PROGRAM SECURITY

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Matvey Soloviev

August 2021

RATIONAL INATTENTION AND A CAUSAL ACCOUNT OF PROGRAM SECURITY

Matvey Soloviev, Ph.D.

Cornell University 2021

This thesis consists of two parts, representing two separate strands of research. The first part is concerned with the formal analysis of program security. We argue that security properties of computer systems can be thought of causally, and use a formal model of causality, the Halpern-Pearl (HP) model, to represent programs and capture a variety of security properties such as noninterference, robust declassification and endorsement. This provides new insights into both causality, where security-inspired scenarios put the existing theory to the test and motivate us to consider various extensions, and security, where causality lets us express security properties in intuitive terms and see what they denote in natural settings.

In the second part, we introduce a theoretical model of information acquisition under resource limitations in a noisy environment. An agent must guess the truth value of a given Boolean formula φ after performing a bounded number of noisy tests of the truth values of variables in the formula. We observe that, in general, the problem of finding an optimal testing strategy for φ is hard, but we suggest a useful heuristic. The techniques we use also give insight into two apparently unrelated, but well-studied problems: (1) *rational inattention*, that is, when it is rational to ignore pertinent information (the optimal strategy may involve hardly ever testing variables that are clearly relevant to φ), and (2) what makes a formula hard to learn/remember.

BIOGRAPHICAL SKETCH

Matvey Soloviev was born in St. Petersburg, Russia, and moved to Germany at an early age. He attended the Martin-Andersen-Nexö-Gymnasium in Dresden, a specialised secondary school for mathematics and the natural sciences, and graduated with the *Abitur* in 2008. Subsequently, he attended Homerton College at the University of Cambridge in the United Kingdom, completing the the three-year Computer Science Tripos and the one-year Part III of the Mathematical Tripos and graduating with a combined BA and MMath in 2013. In the fall of the same year, he entered the PhD program in Computer Science at Cornell University in Ithaca, NY, in the United States of America.

ACKNOWLEDGEMENTS

I am immensely grateful to my advisor, Joe Halpern, for his guidance, insight, support and limitless patience throughout the years. If it were not for his mentorship, I would surely not have made it to this point.

During my time at Cornell, Bobby Kleinberg and Dexter Kozen offered help and advice and helped me overcome many a personal and academic conundrum. I am indebted for their generosity in time and ideas. I would also like to thank Éva Tardos and Karola Mészáros for their advice and being the most understanding Special Committee I could have asked for.

I am further indebted to the colleagues and mentors who have helped me find and retain my bearings as I strayed into the field of program security, including in particular Andrew Myers, M. Milano, Owen Arden, Fred Schneider, Limin Jia, David Naumann, Deepak Garg and Aslan Askarov, and the many others who responded to my questions and provided feedback in the various areas I have meandered through. Thanks is also due to the friends and coauthors with whom I have collaborated along the way, including Seunghee Han, Yuka Kihara, Sarah Tan and Yuwen Wang, and the many academic mentors who have supported me throughout school and university and guided me onto the academic track. I would also like to thank the many friends I have made at Cornell, and those who have stayed with me since before I came here, for the sanity and companionship they provided. Lastly, I thank my family for my upbringing and their unwavering support.

TABLE OF CONTENTS

CHAPTER 1

OVERVIEW

In this thesis, I present two separate strands of research that I have worked on over the course of the recent years. The two parts are loosely connected by a common theme of looking at the intersection of concepts from analytic philosophy, which seek to make formal sense of the way humans and other agents understand and interact with the real world, and computer science.

The first part, *A Causal Account of Program Security*, is concerned with the formal analysis of when a computer system is secure. In order to be considered secure, a system has to maintain specific desirable properties in the face of a malicious attacker, who seeks to use the capabilities at their disposal to subvert them. We argue that stipulations of this type can be thought of causally, in terms of natural propositions that some event or circumstance is a cause of another. Using a formal model of causality, we give alternative definitions of several widely discussed security properties, as well as intuitive accounts of a variety of natura security-relevant scenarios, and thereby seek to demonstrate the utility of this approach. This part incorporates material from two working papers, *Security Properties as Nested Causal Statements* as well as one which shares the title of this part, which are joint work with my advisor, Joe Halpern.

In the second part, *Information Acquisition Under Resource Limitations in a Noisy Environment*, we introduce a game-theoretic model of having to allocate a limited budget of tests or queries among different binary information sources in order to learn whether a Boolean proposition, which may depend on the sources in an arbitrary way, is true or false. This can be seen as a model of a variety of scenarios of interest, such as scientific research, quality assurance and even a

living agent's perception. In this model, it turns out to sometimes be optimal to ignore certain information sources entirely and focus on others, even when a basic symmetry argument shows that the ignored sources are no less important than the ones being attended to. We argue that this is an instance of the notion of *rational inattention*, as introduced by the economist Christopher Sims [37]. After formally defining rational inattention as it emerges in our model, we relate the optimal testing strategies for a given Boolean formula to the points of a particular convex polytope and use linear programming techniques to computationally show that this phenomenon is quite widespread. Finally, we also show that this model can be used as a measure of formula complexity, as some Boolean propositions are easier to learn the truth of by testing than others. This part is based on a single eponymous paper, an earlier version of which appeared at the AAAI 2018 conference.

Part I

A Causal Account of Program Security

CHAPTER 2

INTRODUCTION

Causality is a common feature of our discourse; indeed, it could be argued that the notion that some circumstance is the cause of another is fundamental to the way we make sense of the world around us, providing both explanations of why things are the way they are and guidance on how we should act in order to influence their course. The standard approach to causality involves *counterfactuals*: had the cause not occurred or occurred in a different way than it actually did, the effect would not have come to pass. We are typically interested in the effects of *interventions*, which can be viewed as ways of making the cause occur in a different way.

When designing systems, we are often concerned with whether they are secure. Intuitively, this means that an adversary interacting with the system cannot cause the system to behave in a way the designer didn't intend, nor obtain any information about the system that the designer did not want it to obtain. Programming language theorists have long sought to make precise various intuitions about what behaviours of the system, and what information releases, should be considered acceptable. This has led them to consider security properties, such as *confidentiality* (the notion that the data a system operates on can be partitioned into "more secret" and "less secret" pieces, and the more secret ones should never influence the less secret ones) or *robust declassification* (the notion that decisions to declassify some "more secret" data themselves should not be subject to the influence of an untrusted party). These properties are expressed using a variety of different formalisms, ranging from purely syntactic properties to the runs-and-systems framework underlying Clarkson and Schneider's

[7] hyperproperties and the event-driven approach of Haigh and Young [15].

We argue that many of these properties are best thought of in terms of *causality*: that is, as statements about whether certain facts about the execution of the program are causes of other facts. Causality is arguably fundamental to the way we make sense of the world around us, providing at once explanations of why things are the way they are, judgements on how things should be, and guidance on how we should act in order to influence their course. A causal account of security properties may serve as a bridge between often convoluted statements of security properties and human intuition. In contrast to the aforementioned models from the programming language research community, which were typically designed for the exclusive purpose of analysing computer programs, general-purpose formalisms for causality can capture and represent a wider range of scenarios, including everyday events and their relations. This means that a causal characterization of a security property can be evaluated with respect to a real-world scenario, rather than only the operation of some computer system. Since real-world scenarios are typically closer to human intuition than computer programs, we expect this to be helpful in understanding what a particular security property "really means".

The following examples illustrate how causality arises in security settings:

- A secure facility's janitor is free to open and close the front door, and similarly free to receive money from strangers. However, if he opened the door *because* a stranger gave him money, then he is likely corrupt. This is a violation of *noninterference*, which can be understood as requiring that certain protected effects should never be due to causes that are inappropriate.
- The government is free to research UFOs, and the press is free to write ar-

ticles that claim that the government is researching UFOs. If a newspaper writes an article *because* the government researches UFOs – that is, the article would not have been written had the government decided not to do the research – then this is indicative of an information leak. This is also a violation of noninterference, though the interpretation of what "protected" means is quite different.

- Occasionally, the government chooses to declassify information about UFO research. However, if the disclosure, say, a newspaper report on government UFO research, happened *because* an enterprising journalist bribed a government official to release the information, something is probably amiss. *Robust declassification* requires that no untrusted event should be a cause of a secret event being a cause of a public event; it was likely violated in this example.

A variety of formal models have been proposed for reasoning about causal statements and formally defining what it means to be a cause. We will work in the Halpern-Pearl (HP) framework, which represents situations using *causal models* [17]. Causal models describe the world in terms of variables, which represent significant features, and *structural equations* that describe how these variables depend on each other. In order to be able to reason about programs in the HP framework, we first describe how to represent the behaviour of programs as causal models. We then review a number of security properties from the literature, namely noninterference [14], robust declassification [44], *qualified robustness* [30], and (we believe) *IP-security* [15]. We show that these can be captured by causal formulae asserting that whenever certain causal relationships hold between variables, the security labels of the variables must be appropriately ordered in the policy. To do this, we need to use a language richer than

that that has been traditionally used in work on causality. Specifically, many security properties are most naturally expressed used nested causal statements, which assert that causal relationships themselves have causes. Moreover, due to a natural association between the counterfactual interventions that are used to define causality in the HP framework and actions that agents could potentially have taken, we encode the set of allowed interventions for a given causal relationship as part of the formula, thereby representing the capabilities of the relevant adversary. Finally, we discuss the feasibility of using this approach to evaluate whether a program satisfies a security property. To do so, we use *causal graphs*, a well-known graphical representation of causal models that does not contain all the information of the causal model, but can still provide guarantees that certain causal relationships do not hold in the original model, properties that can be computed using fast graph algorithms.

Organisation. The rest of this part is organised as follows. In Chapter 3, we briefly discuss related work and review the Halpern-Pearl definition of causality. Then, in Chapter 4, we discuss some extensions to the HP framework that are motivated by security, in particular nested causal expressions (such as "A caused B causing C") and the circumstance that we may sometimes need to omit information about the real world from the cause and effect in a causal statement so as to be able to interpret it in an a priori unknown counterfactual setting. In Chapter 5, we then can proceed to show how security properties can be represented in this language. We start with some intuitive, natural-language scenarios (Section 5.1); then, after showing how formally specified programs can be represented as causal models (Section 5.2), we proceed to show how several security properties known from the literature, and other ones that we can imagine to matter, can be formally represented with respect to causal models of

programs. Finally, in Chapter 6, we discuss the complexity of model-checking causal models, and suggest *causal graphs*, a common abstraction of causal models, as a vehicle to improve upon this complexity significantly at the expense of completeness (so it may become impossible to prove some programs that are actually secure to be such).

CHAPTER 3

BACKGROUND

3.1 Related work

In a 2001 survey of security properties, Focardi and Gorrieri [12] lamented the circumstance that countless security properties have been proposed, which are defined on almost as many different system models and accordingly difficult to compare. Many attempts have since been undertaken to systematise the body of security properties [34, 35, 7, 29] and bridge gaps between different models (e.g. [13]). We take particular inspiration from the ambition behind Clarkson and Schneider's hyperproperties framework [7] to identify common structure among security properties and stratify them by complexity (in terms of the number of traces they refer to); our consideration of nesting in causal formulae, and in particular its depth, can be interpreted similarly.

There are multiple ways to cleave the corpus. A somewhat clear separation can be made between data-protection approaches, where security policies are taken to concern data and loci of storage [9, 42, 44, 30, 4], and event-based ones, where security policies are taken to concern the visibility of events or transitions [39, 12, 28, 33, 15]. Less clearly, there is a sliding scale from "passive" approaches, which are mostly concerned with interpreting security policies and formalising what it means for a system to satisfy them [7, 39], and "active" ones, which aim to aid the system designer in making systems secure or even prevent the specification of insecure systems, which in particular includes the field of language-based security [34]. In this taxonomy, our work falls into the "passive" and data-protection categories, though we discuss connections to event-

based work and in particular intransitive policies [15, 39] that are more popular in that sphere, and are planning to explore language-based approaches derived from our framework in future work.

Noninterference [14, 8] may be considered the prototypical security policy [34, 35], and is generally too strong a condition to admit interesting programs. As Sabelfeld and Sands argue (the *conservativity* principle of [35]) for declassification, many security properties take the form of weakenings of, or exceptions to, noninterference. Examples of this that we consider include robust declassification, endorsement, and qualified robustness [44, 30, 31, 4]; the "whitelisting" intransitive policies that we interpret as an instance of IP-security [15, 33, 39] can also be considered an instance.

While we are not aware of existing work to explicitly apply formal models that were developed for causality to security properties, several lines of research involve more generally investigating causality in the context of computer systems. For example, Beer et al. [2] have shown how notions of causality can be used to help determine to what extent each line of code is resposible for a program to fail to satisfy its specification; Ibrahim and his colleagues have used it for accountability, to explain mishaps and possibly to hold misbehaving parties responsible for violations (see, e.g., [21, 22]); Koppel and Jackson [24] used causality to demystify the notion of dependence between modules in software and software modularity. Some of the ideas in these papers may also be relevant in our setting. For example, like Koppel and Jackson, we could consider the security implications of malicious interference with parts of the system at the design rather than the execution level; the notion of degree of responsibility used by Beer et al. (which actually goes back to Chockler and Halpern [6]) is

also relevant to our causally-specified security properties.

3.2 Review of the HP framework

We first review the Halpern-Pearl notion of causality. The first step is to define causal models.

Definition 3.2.1. A *causal model* is a pair $(\mathcal{S}, \mathcal{F})$, consisting of a *signature* $\mathcal{S}$ and a collection of *structural equations* $\mathcal{F}$ for this signature. The signature $\mathcal{S}$ is a triple $(\mathcal{U}, \mathcal{V}, \mathcal{R})$; $\mathcal{U}$ is a nonempty finite set of *exogenous variables*, to be thought of as external inputs to the model, or features of the world whose values are determined outside the model; $\mathcal{V}$ is a nonempty finite set of *endogenous variables*, whose causal dependencies on each other and on the inputs we wish to analyse; each variable $W \in \mathcal{U} \cup \mathcal{V}$ can take values from a finite range $\mathcal{R}(W)$; $\mathcal{F}$ associates with each endogenous variable $V \in \mathcal{V}$ a function denoted F_V that determines the value of V in terms of the values of all the other variables in $\mathcal{U} \cup \mathcal{V}$; thus, $F_V : \prod_{W \in (\mathcal{U} \cup \mathcal{V} - V)} R(W) \to \mathcal{R}(V)$. We typically write, say, $V = U + X$ rather than $F_V(u, x) = u + x$ for all $u \in \mathcal{R}(U)$ and $x \in \mathcal{R}(X)$. □

A variable V *depends on* W if the structural equation for V nontrivially depends on the value taken by W: that is, there are some settings $\vec{z}$ and $\vec{z}'$ of the variables in $\mathcal{U}$ and $\mathcal{V}$ other than V that only differ in the entry corresponding to W such that $F_V(\vec{u}) \neq F_V(\vec{u}')$. Note that this notion represents only *immediate* dependency, and is not transitive: V depending on W and W depending on U does not imply that V depends on U. In this thesis, as is typical in the literature, we consider only models where the dependence relation is acyclic. It follows

that given a *context*, that is, a setting of the exogenous variables, the variables of all the endogenous variables are determined by the equations.

Example 3.2.2. The model $M^{\wedge\text{lamp}}$ has an exogenous variable U whose range is a singleton $\mathcal{R}(U) = \{u\}$ and three endogenous variables $\mathcal{V} = \{\text{SWITCH1}, \text{SWITCH2}, \text{LAMP}\}$, all of whose ranges are $\{\text{on}, \text{off}\}$. In context u, SWITCH1 = SWITCH2 = on, and the structural equation for LAMP is

$$\text{LAMP} = \begin{cases} \text{on} & \text{if SWITCH1=on and SWITCH2=on} \\ \text{off} & \text{otherwise.} \end{cases}$$

□

In order to reason about causal models, HP define the following language, which we will refer to as LI. We start with atomic propositions of the form $X = x$, where X is an endogenous variable and $x \in \mathcal{R}(X)$, and close off under conjunction and negation, and all formulas of the form $[\vec{X} \leftarrow \vec{x}]\varphi$, where X is a vector of endogenous variables, which says that after intervening to set the variables in $\vec{X}$ to $\vec{x}$, φ holds.

To interpret the truth of a statement in a world described by a causal model, we need to determine what values the variables can actually take, given the structural equations and the values taken by exogenous variables. An *assignment* to the endogenous variables $\mathcal{V}$ is a vector $\vec{v}$ implicitly indexed by $\mathcal{V}$, where the entry corresponding to each variable $V \in \mathcal{V}$ is in $\mathcal{R}(V)$. The context $\vec{u}$ is an assignment to the exogenous variables, defined similarly. Given a context $\vec{u}$ and an assignment $\vec{v}$ of the endogenous variables, we say that the pair $(\vec{u}, \vec{v})$ is *compatible* with M if the entry in $\vec{v}$ for each variable $V \in \mathcal{V}$ is *compatible* with M, that is, $\vec{v}(V) = F_V(\vec{u}, \vec{v} - \{\vec{v}(V)\})$ for all $V \in s(V)$. For models where the

depends-on relation is acyclic, as we assume here, there is a unique $\vec{v}$ for each $\vec{u}$ such that $(\vec{u}, \vec{v})$ is compatible with M.

We can now give a semantics to propositional formulae involving atomic propositions of the form $V = x$, where $V \in \mathcal{V} \cup \mathcal{U}$, taken to mean that the variable V takes value x. Recursively, we set $(M, \vec{u}) \models X = x$ if the X-indexed entry of $(\vec{u}, \vec{v})$ has value x when $\vec{v}$ is the unique assignment such that $(\vec{u}, \vec{v})$ is compatible with M. We say $M \models X = x$ when $(M, \vec{u}) \models X = x$ for *all* $\vec{u}$. We can extend $\models$ to Boolean combinations of atomic formulas, as well as quantifications over values ("$\forall v \in \mathcal{R}(V)$") in the obvious way.

To give semantics to formulas of the form $[X \leftarrow x]\varphi$ in M, we first need to define the model $M_{\vec{X} \leftarrow \vec{x}}$. The model $M_{\vec{X} \leftarrow \vec{x}}$ is just like M, except that the structural equations for the variables in $\vec{X}$ are replaced by the corresponding entries of $\vec{x}$; that is, the equation for $X \in \vec{X}$ becomes $X = x$. We then set $(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}]\varphi$ iff $(M_{\vec{X} \leftarrow \vec{x}}, \vec{u}) \models \varphi$. We read this formula as "if the variables $\vec{X}$ were set to the values $\vec{x}$, then φ would be true".

For example, the model $M^{\wedge\text{lamp}}_{\text{SWITCH1} \leftarrow \text{off}, \text{LAMP} \leftarrow \text{on}}$ is the same as M, except that in it, $F_{\text{SWITCH1}} = \text{off}$ and $F_{\text{LAMP}} = \text{on}$. In particular, in this model, whether the lamp is on does not depend on the state of either of the switches. We also have

$$M^{\wedge\text{lamp}} \models \text{LAMP} = \text{on} \wedge [\text{SWITCH1} \leftarrow \text{off}]\text{LAMP} = \text{off} :$$

the lamp is on, and if the first switch were set to be switched off, the lamp would be off.

As is standard, we define $M \models \varphi$ to mean that $(M, \vec{u}) \models \varphi$ for all contexts $\vec{u}$. In particular, if there is only a single possible context u (as is the case in many of our examples), then it is equivalent to $(M, u) \models \varphi$. Moreover, since the ranges

of all variables are finite, we can take $\exists v \in \mathcal{R}(X).\, \varphi$ to be syntactic sugar for the disjunction $\bigvee_{x \in \mathcal{R}(X)} \varphi[x/v]$, that is, the formula that is true iff it is true with some value from the range of X substituted for v. When the range is clear from the context, we may simply write $\exists v.\, \varphi$.

When capturing some concrete security properties, it actually turns out that the general HP definition of causality is not necessarily the appropriate one to use. Instead, we will want to use the well-known more basic notion of *but-for causality*, which, roughly speaking, simply says that A is a cause of B if, had A not occurred, then B would have occurred. In a slight twist on this standard notion, we introduce an additional parameter that allows us to restrict the set $\mathcal{I}$ of possible interventions. For some interventions $\vec{X} \leftarrow \vec{x}$, the model $M_{\vec{X} \leftarrow \vec{x}}$ is nonsensical, uninteresting, or exceedingly unlikely. For example, when discussing the security of computer systems, we would rarely invoke counterfactual scenarios corresponding to random "cosmic ray" bit flips as justifications for a causal relationship. Whether a counterfactual is appropriate may even differ within a single formula containing multiple causal statements, if they concern different principals: It may, for example, seem fair to say that a system failure was caused by an administrator who could have updated a key component but didn't, but less so that it was caused by a user who would never have the ability to do so.

Definition 3.2.3. $\vec{X} = \vec{x}$ is a *but-for cause* of φ in $(M, \vec{u})$ with respect to the interventions $\mathcal{I}$, denoted $(M, \vec{u}) \models (\vec{X} = \vec{x}) \rightsquigarrow_{\mathcal{I}} \varphi$, if

- $(M, \vec{u}) \models \vec{X} = \vec{x}$ (the purported cause is true),
- $(M, \vec{u}) \models \varphi$ (the purported effect is true) and
- there exists some $\vec{X} \leftarrow \vec{x}' \in \mathcal{I}$ such that $(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}']\neg\varphi$ (if $\vec{X}$ hadn't been $\vec{x}$, then the effect would not have occurred).

If the set $\mathcal{I}$ contains all possible interventions $\{\vec{X} \leftarrow \vec{x}' \mid \vec{x}' \in \mathcal{R}(\vec{X})\}$, we can omit it to simply write $\vec{X} = \vec{x} \rightsquigarrow \varphi$.

By the first condition in the definition of but-for causality, in every context, there is only one possible value of $\vec{x}$ for which $(\vec{X} = \vec{x}) \rightsquigarrow_{\mathcal{I}} \varphi$ is possibly true; sometimes it is useful to be able to make a causal statement independent of this value. We take $\vec{X} \rightsquigarrow_{\mathcal{I}} \varphi$ to be an abbreviation of $\exists \vec{x}.(\vec{X} = \vec{x}) \rightsquigarrow_{\mathcal{I}} \varphi$. Similarly, by the second condition in the definition, there is only one possible value of $\vec{y}$ such that $(\vec{X} = \vec{x}) \rightsquigarrow_{\mathcal{I}} \vec{Y} = \vec{y}$ is true. We define the abbreviation $\vec{X} \rightsquigarrow_{\mathcal{I}} \vec{Y}$ analogously. We then say that *X is a (but-for) cause of* φ, *X is a (but-for) cause of Y*, and so on. □

Note that we allow φ to itself be a modal formula. This was not allowed in the original HP definition; φ was restricted to being a Boolean combination of primitive events of the form $\vec{X} = \vec{x}$, although the HP definition makes sense (without change) if we do allow it. Allowing φ to be a modal formula enables us to discuss propositions involving nested causality such as $A \rightsquigarrow (B \rightsquigarrow C)$.

The set of allowed interventions allows us to exercise control over the counterfactual scenarios we are willing to invoke in a causal statement, thus encoding the relevant set of capabilities of an attacker that we want to analyse security properties relative to. In some scenarios, we want to further restrict this set to take into the account additional knowledge that we obtained about the values a variable could take, so we may ask questions such as "given that A could have taken only the values 0 or 2, could an attacker intervention upon A have caused this effect?". By analogy with probability, we call this *conditioning*, as the probabilistic notion likewise may capture how our understanding of a scenario changes if we can rule out certain events. The notation we chose is meant to

suggest this connection.

Definition 3.2.4. If $\mathcal{I}$ and S are sets of allowed interventions of the form $\vec{X} \leftarrow \vec{x}$, then the set $\mathcal{I} \mid S$, read $\mathcal{I}$ *conditioned on* S, denotes the intersection of $\mathcal{I}$ and S. We often abbreviate S as a set of values, such as $V \in \{0, 2\}$, rather than an explicit set of interventions. This should be understood to denote the set of all possible $\vec{X} \leftarrow \vec{x}$ that satisfy the predicate. Thus, for example, $\mathcal{I} \mid (V \in \{0, 2\})$ is an abbreviation of $\mathcal{I} \mid \{V \leftarrow 0, V \leftarrow 1\}$. □

In our setting, it suffices to consider restrictions on the possible values of each variable separately, so we do not take into account "entangled" restrictions such as "$A + B = 1$", but such a generalisation could be defined in a straightforward manner.

3.2.1 Actual Causality

In Chapter 4, we discuss several extensions to the HP framework. Even though these extensions are motivated by security, we believe that they should be of more general interest, and it is therefore in our interest to make them as widely applicable as possible. To that end, we will define and analyse our extensions for a state-of-the-art definition of causality that has been argued to capture many real-world scenarios, rather than only the basic but-for causality that we require for the security properties that we consider later. Halpern and Pearl actually discuss three definitions (see [18, 19, 16, 17]). We consider the most recent one [16, 17], called the *modified HP definition*, since it is simplest and seems most robust. The key results in that chapter apply equally to but-for causality with minimal or no change.

Definition 3.2.5. $\vec{X} = \vec{x}$ is an *actual cause* of φ in $(M, \vec{u})$ if

AC1. $(M, \vec{u}) \models \varphi$ and $(M, \vec{u}) \models \vec{X} = \vec{x}$;

AC2. There is a set $\vec{W}$ of variables in $\mathcal{V}$ and a set of alternative values $\vec{x}'$ for the variables in $\vec{X}$ such that if $(M, \vec{u}) \models \vec{W} = \vec{w}^*$, then $(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}', \vec{W} \leftarrow \vec{w}^*]\neg\varphi$.

AC3. $\vec{X}$ is minimal; there is no strict subset $\vec{X}'$ of $\vec{X}$ for which AC2 holds.

For the rest of this section, we will take $\vec{X} = \vec{x} \rightsquigarrow \varphi$ to represent that $\vec{X} = \vec{x}$ is an actual cause of φ, so that $(M, \vec{u}) \models \vec{X} = \vec{x} \rightsquigarrow \varphi$ if AC1–3 hold. Note that conditions AC1–3 are all expressible in the language LI. □

In Definition 3.2.5, HP assume that φ is a Boolean combination of atomic propositions. In particular, φ may not itself be a formula of the form $\vec{X} = \vec{x} \rightsquigarrow \varphi$. Since we want to reason about the expressive power of this notion of causality, it is useful to explicitly define the language obtained by augmenting propositional logic with it.

Definition 3.2.6. A formula φ is *simple* if it is a Boolean combination of atomic propositions of the form $X = x$.

A formula ψ is a *simple causal formula* if it is a Boolean combination of atomic propositions of the form $X = x$ and causal statements of the form $\vec{X} = \vec{x} \rightsquigarrow \varphi$, where φ is simple. The language of all simple causal formulae is called LC_1. □

We give semantics to formulae in LC_1 by converting them to formulas in LI and using Definition 3.2.5.

CHAPTER 4

EXTENDING THE HP FRAMEWORK FOR NESTED CAUSALITY

In the HP framework, potential causes are taken to be conjunctions of atomic propositions about the values taken by variables, while effects are taken to be Boolean combinations of atomic propositions. However, in real-world discourse, we often encounter seemingly more complex statements, including, in particular, ones where the purported effect is itself a causal statement ("because I have paid my electricity bill on time, flipping the light switch on causes the lamp to turn on"). Nested causal statements of this form are particularly common when discussing notions of authorisation, delegation, and endorsement.

For example, consider a government employee who is authorised to publicly confirm a classified piece of information, say, that the government has made contact with an alien civilization. This employee is corrupt, and may release the information if he is bribed. The release of the information, which itself can be interpreted causally as saying that the fact that contact was made is a cause of the newspaper article saying that it is, is allowed; the employee is authorised to release information. The bribe on its own is also not necessarily bad; nothing prohibits acts of kindness towards strangers. The problem here is that the gift of money caused the alien contact to be a cause of the newspaper article. This corresponds to the notion of *robust declassification* from the security literature [44, 30, 4], which can be interpreted as more generally saying that whenever secret causes have public effects, this must not itself be due to untrustworthy causes. The underlying notion of robustness can be taken more generally to denote that some security-relevant circumstance, which could be primitive or itself involve causality, is not caused by an untrusted party.

Conversely, nested causes may also render normally unacceptable causal relationships acceptable. For example, if A performs an action that infringes on B's possessions, such as redecorating their office, shredding papers, or changing the settings of B's computer, and B finds this objectionable, a common defense that A might invoke is to say that they would have stopped if B had told them to (and B had the opportunity to do so). In other words, whatever causal relationship there was between A's intentions and B's property held only because B didn't voice an objection, and thus implicitly endorsed the act. We can view this line of reasoning, where an unauthorised cause has a certain effect because of an authority's implicit or explicit go-ahead, as a form of authorisation. The construction can easily be nested, obtaining examples where one authority holds a veto over another authority's ability to hold a veto over a causal relationship, and so on. There is extensive literature on reasoning about authorisation chains of this kind using *authorisation logics* [1], and nested causality can be used as a way to interpret them.

The HP definition of causality does not deal with the nested causality statements of the type that occur in the examples above. Fortunately, the HP definition can be extended without change to apply to nested statements, and seems to give sensible results. Using nested causality allows us to distinguish causal scenarios that appear different in a security-relevant way that cannot be distinguished without nesting unless we to unnatural edge cases of the definitions (Theorem 4.1.3). Moreover, having effects that themselves involve counterfactuals introduces new considerations that are, in a precise sense, irrelevant to the "simple" causal statements the HP model was designed for. These considerations suggest a further modification to the HP definition. Specifically, the HP definition assumes that both cause and effect are formed from Boolean combi-

nations of atomic propositions of the form $X = x$: variable X has value x. Thus, it does not say "the switch's position is the cause of whether the lamp was on", but rather "the switch being on is the cause of the lamp being on". The latter statement is more idiomatic in natural language, but provides no more information than the former: the HP definition of actual causality implies that the statement "$X = x$ is a cause of $Y = y$" can be true for only one particular value of x and y, namely the values of X and Y in the actual context.

The picture becomes more interesting with nested causality. For example, imagine an American vegetable grower who relocated to Texas. As it happens, the year was marred by climatic irregularities; while the southern states, including Texas, experienced a drought, all remaining states, including the grower's state of origin, were subject to catastrophic floods instead. The grower now makes the following statement: "Because we moved to Texas, the weather caused our crop to fail." This is arguably false: had the grower not moved to Texas, the flooding would have led to crop failure all the same. How would we make sense of this in the HP framework? What value of the variable weather are we referring to here? Naively extending upon the previous observation, the first thought would be to use the value of weather in the actual context, so it becomes "Because we moved to Texas, the weather being dry caused our crop to fail." But this statement is true: had the grower stayed in New York, the weather would not have been dry, and so dry weather could not have caused crop failure. Plugging in another constant value does not work either; while "Because we moved to Texas, the weather being very wet caused our crop to fail" is false, due to the statement "the weather being very wet caused our crop to fail" itself being false. This statement can't be the intended meaning of "the weather caused our crop to fail", because we would normally take the latter

statement to be true!

We claim that the most reasonable interpretation of this type of statement is instead as a form of causality that is independent of the concrete value that the weather takes. This can be interpreted in terms of the HP notion in terms of existential quantification: "there exists a state v of the weather such that the weather being v caused our crop to fail". We return to this point in Section 4.2.

4.1 Nested causal statements

Our goal is to investigate the role of nested causal statements such as "$\vec{A} = \vec{a}$ is a cause of $\vec{B} = \vec{b}$ being a cause of φ". These statements have no formal counterpart in LC_1, but we can define a language which includes them.

Definition 4.1.1. The language LC_∞ of *nested causal formulae* is defined recursively as follows:

- Simple formulae φ are in LC_∞.
- If φ is in LC_∞, then so is $\vec{X} = \vec{x} \rightsquigarrow \varphi$.
- Boolean combinations of formulae in LC_∞ are in LC_∞.

□

Since Definition 3.2.5 does not depend on the structure of φ in $\vec{X} = \vec{x} \rightsquigarrow \varphi$, we can once again use it to give a semantics to LC_∞ by evaluating the corresponding formula in LI. As we now show, the inclusion of nested causal statements results in LC_∞ being more expressive than LC_1 once we exclude a particular set of undesirable formulae.

Let $M^{\neg\otimes\text{lamp}}$ be the same as the model $M^{\wedge\text{lamp}}$ from Example 3.2.2, except that

$$F_{\text{LAMP}} = \begin{cases} \text{on} & \text{if SWITCH1=SWITCH2} \\ \text{off} & \text{otherwise,} \end{cases}$$

so the lamp is on if both or neither switch is. Intuitively, the two models $M^{\wedge\text{lamp}}$ and $M^{\neg\oplus\text{lamp}}$ are quite distinct. If we think of each model as representing different setups in which two people each control a light switch, then in $M^{\wedge\text{lamp}}$, each participant has a veto on whether the light is on: if they choose to keep their switch "off", then nothing the other person can do has any bearing on whether the light is on or not. On the other hand, in $M^{\neg\oplus\text{lamp}}$, by flipping their own switch, each person can only temporarily toggle the light, perhaps to mess with the other participant, but has no way of ensuring that the light will permanently remain in any particular state. In security parlance, we could think of this as saying that in $M^{\wedge\text{lamp}}$, each person has to independently authorise the other to be able to influence the light.

We want to show that natural simple causal formulae are not sufficiently expressive to capture the difference between $M^{\wedge\text{lamp}}$ and $M^{\neg\oplus\text{lamp}}$. The qualification *natural*, however, does some work here: in order to make this statement precise, we need to restrict the set of causal formulas that we consider. Specifically, the intuition above does not necessarily hold for some causal statements where cause and effect refer to the same thing (such as "it is raining because it is warm and raining"). We typically do not make such statements; it seems strange to say "$X = x$ is a cause of $X = x$" or "$X = x$ is a cause of $X = x$ and $Y = y$".

Definition 4.1.2. For a given formula φ let $\text{Vars}(\varphi)$ denote the set of variables (each X in the atom $X = x$) in φ. The formula $\varphi \rightsquigarrow \psi$ is *circular* if $\text{Vars}(\varphi) \cap \text{Vars}(\psi) \neq \varnothing$. Let LC_1^{nc} and $\text{LC}_\infty^{\text{nc}}$ denote the *n*on-*c*ircular fragments of LC_1 and

LC_∞ respectively. □

As we now show, we cannot distinguish $M^{\wedge\text{lamp}}$ and $M^{\neg\oplus\text{lamp}}$ using non-nested non-circular formulae, but with nested non-circular formulae, we can.

Theorem 4.1.3. *For all φ in $\mathrm{LC}_1^{\mathrm{nc}}$,*

$$M^{\wedge\text{lamp}} \models \varphi \textit{ iff } M^{\neg\oplus\text{lamp}} \models \varphi,$$

but there exists a nested non-circular causal statement $\psi \in \mathrm{LC}_\infty^{\mathrm{nc}}$ such that $M^{\wedge\text{lamp}} \models \psi$ and $M^{\neg\oplus\text{lamp}} \models \neg\psi$.

Proof. We start by showing that non-circular formulae cannot distinguish the models. By exhaustive checking, we can confirm that $M^{\wedge\text{lamp}} \models X = x$ iff $M^{\neg\oplus\text{lamp}} \models X = x$ for all $X \in \mathcal{V}$ and $x \in \mathcal{R}(X)$. It easily follows that $M^{\wedge\text{lamp}} \models \varphi$ iff $M^{\neg\oplus\text{lamp}} \models \varphi$ for simple formulae.

Formulae in $\mathrm{LC}_1^{\mathrm{nc}}$ are Boolean combinations of atomic propositions of the form $X = x$ and non-circular causal formulae of the form $\vec{X} = \vec{x} \rightsquigarrow \varphi$, where φ is a simple formula. Hence, if we can also establish that causal formulas are valid in $M^{\wedge\text{lamp}}$ iff they are valid in $M^{\neg\oplus\text{lamp}}$, the result easily follows. We do this by considering a number of cases.

If the variable LAMP does not occur in φ, then the causal formula is false in both $M^{\wedge\text{lamp}}$ and $M^{\neg\oplus\text{lamp}}$. To see this, note that by non-circularity, φ can mention only SWITCH1 and SWITCH2, but neither of these variables depends on any other variable in either model. By non-circularity, $\vec{X}$ does not mention the variables in φ, so no change to the value of $\vec{X}$ change the value of SWITCH1 or SWITCH2 (even if some variables $\vec{W}$ are fixed to their actual values). Thus, AC2 cannot hold in either model, so the causal formula is false in both models.

Now suppose that LAMP occurs in φ, and hence LAMP $\notin \vec{X}$. Consider the possible cases for $\vec{X}$. If $\vec{X} = \vec{x}$ contains either SWITCH1 = off or SWITCH2 = off, then $\vec{X} = \vec{x}$ is false in both models, so AC1 fails, and the causal formula is false in both models. If $\vec{X} = \emptyset$, then AC2 must fail in both models (since no change in the value of a variable in $\vec{X}$ can cause a change in the truth value of φ. If $\vec{X} = \vec{x}$ is SWITCH1 = on, then changing SWITCH1 to off (while possibly keeping some variables fixed at their actual values) has the same effect on the truth value of all the variables in both models, so AC2 will either hold in both models or in neither, and AC3 trivially holds in both. A similar argument works if $\vec{X} = \vec{x}$ is SWITCH1 = on. Finally, if $\vec{X} = \vec{x}$ is SWITCH1 = on $\wedge$ SWITCH1 = on, then the causal formula must be false in both models. Either at least one of AC1 and AC2 is violated, or we must have $\varphi \Leftrightarrow$ LAMP = on in both models as AC1 necessitates $M^{\wedge\text{lamp}}, M^{\neg\oplus\text{lamp}} \models \varphi$, AC2 necessitates $M^{\wedge\text{lamp}}, M^{\neg\oplus\text{lamp}} \models [\text{SWITCH1} \leftarrow v', \text{SWITCH2} \leftarrow w']\neg\varphi$, and by non-circularity, φ can mention only LAMP. But in both models, [SWITCHi $\leftarrow$ off]$\neg$(LAMP = on) for $i \in \{1, 2\}$, so SWITCHi = on is already a cause of LAMP = on and AC3 (minimality) is violated.

We now show that the models can be distinguished by nested non-circular formulae. We claim that

$$M^{\wedge\text{lamp}} \models \text{SWITCH1} = \text{on} \rightsquigarrow (\exists v.\, \text{SWITCH2} = \text{on} \rightsquigarrow \text{LAMP} = v),$$

but

$$M^{\neg\oplus\text{lamp}} \not\models \text{SWITCH1} = \text{on} \rightsquigarrow (\exists v.\, \text{SWITCH2} = \text{on} \rightsquigarrow \text{LAMP} = v).$$

In both models, both SWITCH1 = on and SWITCH2 = on $\rightsquigarrow$ LAMP = on are valid, as intervening to set SWITCH2 $\leftarrow$ off results in LAMP = off. However, if we

intervene to set SWITCH1 $\leftarrow$ off, we have $M^{\wedge\text{lamp}} \not\models \text{SWITCH2} = \text{on} \rightsquigarrow \text{LAMP} = v$ for any v, as LAMP = off regardless of the setting of SWITCH2. This is not the case in $M^{\neg\oplus\text{lamp}}$, as there we have

$$M^{\neg\oplus\text{lamp}} \models [\text{SWITCH1} \leftarrow \text{off}](\text{SWITCH2} = \text{on} \rightsquigarrow \text{LAMP} = \text{off}).$$

Intervening further to set SWITCH2 $\leftarrow$ off results in the lamp turning back on, as both switches are in the same position again. □

Remark 4.1.4. It is worth noting that non-circularity is really a necessary condition here. Let φ be the formula SWITCH1 = on $\wedge$ SWITCH2 = on $\wedge$ LAMP = on; let ψ be the formula SWITCH1 = on $\vee$ SWITCH2 = on $\vee$ LAMP = on. Note that the formula $\varphi \rightsquigarrow \psi$ is circular. Moreover, $M^{\neg\oplus\text{lamp}} \models \varphi \rightsquigarrow \psi$, as we can intervene to set all the variables to *off*, but $M^{\wedge\text{lamp}} \not\models \varphi \rightsquigarrow \psi$, because AC3 is violated: we have $M^{\wedge\text{lamp}} \models \text{SWITCH1} = \text{on} \wedge \text{SWITCH2} = \text{on} \rightsquigarrow \psi$. The corresponding causal statement does not hold in $M^{\neg\oplus\text{lamp}}$ because LAMP = on after setting both switches *off*.

In fact, as we show in Section 4.3, similar circular formulae in LC_1 can distinguish any two distinct causal models. □

While Theorem 4.1.3 formalizes our claim that $M^{\wedge\text{lamp}}$ and $M^{\neg\oplus\text{lamp}}$ can be distinguished by nested causal formulae, the existential quantification in the distinguishing statement makes it somewhat difficult to understand exactly what it is about the models that is different. The formula seems to be saying that the first switch being on is a cause of the second switch being on causing *something* about the lamp.

This *something* can not be expressed as LAMP taking a particular value: if we took it to be LAMP = on, then the resulting nested causal statement would be

valid in both $M^{\wedge\text{lamp}}$ and $M^{\neg\oplus\text{lamp}}$ by AC2, as after intervening to set SWITCH1 $\leftarrow$ off, LAMP is not = on anymore, and so the effect SWITCH2 = on $\rightsquigarrow$ LAMP = on is becomes false by AC1. On the other hand, if we took it to be LAMP = off, then the nested causal statement would be invalid in both, as AC1 requires that the effect SWITCH2 = on $\rightsquigarrow$ LAMP = off is valid in the real world, and another application of AC1 necessitates LAMP = off, but the lamp is really on.

In the next section, we argue that the deliberately fuzzy wording ("…causing something about the lamp") actually captures the existential quantification, which sidesteps the issue of not being able to choose a fixed value, and that in the presence of nested causality, it turns out to sometimes make sense to avoid committing to particular values.

4.2 Variables, rather than facts, as causes

We previously observed that out of all the possible causal statements of the form $A = a \rightsquigarrow (C = c)$, where $a \in \mathcal{R}(A)$, $c \in \mathcal{R}(C)$, only one is potentially true, namely the one where a and c are the actual values that A and C, respectively, take in the context. There is therefore a sense in which specifying a and c is redundant: we could unambiguously interpret a formula like $A \rightsquigarrow C$ as meaning $A = a \rightsquigarrow (C = c)$ in each context u with a, c such that $(M, u) \models A = a \wedge C = c$. Things are not so simple in the case of nested causality. Suppose that, in the causal formula, $\vec{C} = \vec{c} \rightsquigarrow \varphi$, φ itself is a causal formula of the form $\vec{A} = \vec{a} \rightsquigarrow (\vec{B} = \vec{b})$. Now the values of a and b for which the formula is true depend on the value c' to which C is set when evaluating the counterfactual. As the following example shows, this can play a critical role.

Example 4.2.1. Let M be the following model of the farmer story from the introduction. A farmer may relocate to Texas ($R = 1$) or stay in New York ($R = 0$), and this will impact the level of drought his crops are exposed to ($W = 0$ standing for drought, $W = 1$ standing for normality, and $W = 2$ standing for flooding). If he relocates, his crops will suffer dry weather; otherwise, they will be flooded:

$$W = \begin{cases} 2 & \text{if } R = 0 \\ 0 & \text{otherwise.} \end{cases}$$

The crops, however, can survive ($C = 1$) only if the weather is fair ($W = 1$):

$$C = \begin{cases} 1 & \text{if } W = 1 \\ 0 & \text{otherwise.} \end{cases}$$

Intuitively, assuming the farmer relocated, we hold that the statement "the weather caused the crops to fail" is true; but the statement "because the farmer relocated to Texas, the weather caused the crops to fail" is false. Finally, we add a single exogenous variable U with singleton range. In this context, $R = 1$. □

What should the formal interpretation of the nested statement be? If we take the interpretation of "the weather caused the crops to fail" to be $W = 0 \rightsquigarrow (C = 0)$, then this formula is indeed true (as $M \models [W \leftarrow 1]C = 1$). If we then interpret the nested statement as

$$R = 1 \rightsquigarrow (W = 0 \rightsquigarrow (C = 0)),$$

then this is in fact true as well: $M \models [R \leftarrow 0](W = 2)$, and hence $M \models [R \leftarrow 0]\neg(W = 0 \rightsquigarrow (C = 0)$! It does not help to interpret the causal formula as

$$R = 1 \rightsquigarrow (W = 2 \rightsquigarrow (C = 0)).$$

This statement is false for vacuous reasons: $R = 1$ is not a cause because $W = 2 \rightsquigarrow C = 0$ is false, as W is not 2. What seems to capture this example best is to use of existential quantification:

$$R = 1 \rightsquigarrow (\exists w.\, W = w \rightsquigarrow (C = 0)).$$

The common feature of this example and the distinguishing formula in the proof of Theorem 4.1.3 is that we do not know in advance what value is appropriate to impute to variables in the inner causal statement: when considering the outer statement, we need to consider all possible counterfactual scenarios concerning its cause, but the values of the variables mentioned in the inner statement may be different in each of those scenarios. Therefore, we make the following definition, which allows us to capture the notion that A taking the situationally appropriate value, whatever it is in the scenario that we might be considering for a particular causal (sub)formula, is the cause of the B taking whatever value it happens to take. In the next section, we will see examples of several security properties that are expressed more succinctly with this notation, and in some cases cannot be expressed adequately without the existential quantification at all.

Definition 4.2.2. Suppose $\vec{A}$ and $\vec{B}$ are lists of variables of an appropriate causal model. Let $\vec{A} \rightsquigarrow \vec{B}$ denote the formula

$$\exists \vec{a} \in \mathcal{R}(\vec{A}), \vec{b} \in \mathcal{R}(\vec{B}).\, \vec{A} = \vec{a} \rightsquigarrow (\vec{B} = \vec{b}).$$

We can define $\vec{A} = \vec{a} \rightsquigarrow \vec{B}$ and $\vec{A} \rightsquigarrow (\vec{B} = \vec{b})$ analogously. □

Using this definition, we can now state the causal statement of Example 4.2.1 in a way that mirrors the natural-language version, by saying $R = 1 \rightsquigarrow (W \rightsquigarrow$

$(C = 0))$, or even more succinctly as $R \rightsquigarrow (W \rightsquigarrow C)$. Likewise, the distinguishing formula from the proof of Theorem 4.1.3 can now be stated as SWITCH1 = on $\rightsquigarrow$ (SWITCH2 = on $\rightsquigarrow$ LAMP), or more compactly as SWITCH1 $\rightsquigarrow$ (SWITCH2 $\rightsquigarrow$ LAMP).

4.3 The power of circular causal statements

Proposition 4.3.1. *Let $(M^1, \vec{u}_1)$ and $(M^2, \vec{u}_2)$ be two contexts of recursive models with the same signature $(\mathcal{U}, \mathcal{V}, \mathcal{R})$, same immediate dependency ordering, same unique consistent assignment $\vec{v}$ and different structural equations. Then there is a simple causal formula $\vec{X} = \vec{x} \rightsquigarrow \varphi \in \mathrm{LC}_1$ that is valid in $(M^1, \vec{u}_1)$ but not in $(M^2, \vec{u}_2)$.*

Proof. Let D be a minimal variable in the dependency ordering (which is th same for both models) whose structural equation takes different values in $(M^1, \vec{u}_1)$ and $(M^2, \vec{u}_2)$ on some input (assignment to endogenous variables that the structural equation depends on) $\vec{v}'$, say $F^1_D(\vec{u}_1, \vec{v}') \neq F^2_D(\vec{u}_2, \vec{v}')$, and take $\vec{X} \subseteq \mathcal{V}$ to be the strict descendants of D in the dependency ordering. Let $\vec{x}$ be the corresponding entries of $\vec{v}$, and let

$$\varphi = \neg(D = F^1_D(\vec{u}_1, \vec{v}')) \vee \bigvee_{Y \in \vec{X}} \neg(Y = \vec{v}'(Y)).$$

That is, φ says that D or one of its descendants has a different value than it would have on input $\vec{v}'$.

Then $(M^1, \vec{u}_1) \models \vec{X} = \vec{x}$ and $(M^2, \vec{u}_2) \models \vec{X} = \vec{x}$ by construction, and $(M^1, \vec{u}_1) \models \varphi$, $(M^2, \vec{u}_2) \models \varphi$ because $\vec{v} \neq \vec{v}'$ (recall we are assuming that the two models have the same unique consistent assignment, so $F^1_D(\vec{v}) = F^2_D(\vec{v}) = \vec{v}(D)$) and so at least one of the disjuncts must be true in both models (which are

only consistent with the assignment $\vec{v}$). Moreover, $(M^1, \vec{u}_1) \models [\vec{X} \leftarrow \vec{x}']\neg\varphi$, where $\vec{x}'$ are the entries of $\vec{v}'$ corresponding to $\vec{X}$: the clauses of the disjunction are satisfied by the intervention directly setting $Y = \vec{v}'(Y)$ for all $Y \in \vec{X}$, and $D = F^1_D(\vec{v}')$ as a consequence of the structural equations of M^1. So either $(M^1, \vec{u}_1) \models \vec{X} = \vec{x} \rightsquigarrow \varphi$, or this only fails AC3 and so there is some proper subvector $\vec{X}'$ such that $(M^1, \vec{u}_1) \models \vec{X}' = \vec{x} \rightsquigarrow \varphi$ for which this is true. However, for all $\vec{x}''$, $(M^2, \vec{u}_2) \models [\vec{X} \leftarrow \vec{x}'']\varphi$: either $\vec{x}'' \neq \vec{x}'$, in which case one of the disjuncts about $Y \in \vec{X}$ is true in $(M^2_{\vec{X}\leftarrow\vec{x}''}, \vec{u}_2)$, or $\vec{x}'' = \vec{x}'$, in which case we have $(M^2_{\vec{X}\leftarrow\vec{x}''}, \vec{u}_2) \models D = F^2_D(\vec{v}')$, and hence $\neg(D = F^1_D(\vec{v}'))$. Since this is true for all values we could assign to $\vec{X}$, restricting to a subvector $\vec{X}'$ of $\vec{X}$ will not help, and $(M^2\vec{u}_2) \not\models \vec{X}' = \vec{x} \rightsquigarrow \varphi$ for all $\vec{X}' \subseteq \vec{X}$. □

Note that if the two models do not have the same unique consistent assignment $\vec{v}$, we don't even need causal statements to distinguish them: just use a single atomic proposition about a variable on which their consistent assignments differ.

CHAPTER 5

A CAUSAL LOOK AT SOME SECURITY PROPERTIES

5.1 Informal examples

Now that we have formally defined causality and shown that causal statements in our extended language are meaningful, we are ready to revisit several examples of propositions about security that we want to argue are naturally viewed as nested causal statements.

Example 5.1.1. A government employee has the authority to declassify government secrets and release them to the press. The employee turns out to be corrupt: if someone pays him a sufficient amount of money, he will declassify a secret and have it published in the press. As it happens, a UFO enthusiast community scrapes together a bribe and pays the employee, who subsequently publishes the announcement that the government has been in contact with aliens. □

In what sense can we say that something inappropriate occurred? By assertion, we considered it permissible for the employee to declassify and release the secret (and thus for the truth about aliens to be a cause of the press release). In a free market economy, tax regulations not withstanding, people are free to give money to whomever they please. Lastly, had the UFO enthusiasts instead paid a struggling newspaper directly to announce that the government found UFOs, this would also not be problematic. The problem here is that that whether the information was released depended on whether the bribe was paid. In other words, the problem is that the bribe was a cause of the government being in

contact with aliens being a cause of the press release.

Formally, we can represent the example as a causal model M^{aliens} with a single exogenous variable U whose range is a singleton, and three endogenous binary variables S, P, and B, representing whether the government is secretly in contact with aliens, whether there is a press article to the effect, and whether the UFO enthusiasts paid a bribe respectively. Due to the employee's corruption, we have

$$F_P = \begin{cases} S & \text{if } B{=}1 \\ 0 & \text{otherwise.} \end{cases}$$

The undesirable causal relationship then is represented by the formula $B \rightsquigarrow (S \rightsquigarrow P)$.

The security property being violated here is an instance of Zdancewic and Myers's notion of *robust declassification* [44, 30, 4]. Roughly speaking, a declassification (release of a secret) is considered *robust* if whether the declassification occurred was not up to an untrusted actor. What is considered a secret and what actors are trusted (to declassify the secret) has to be specified as part of the security policy. In causal terms, we can say that a system satisfies robust declassification if there is no instance of an *untrusted* variable (such as the UFO enthusiasts' decision to pay) being a cause of a *secret* variable being a cause of a *public* variable. Which variables belong to each of the three classes has to be specified as part of the security policy. Intuitively, secret variables are those for which we would consider it *a priori* unacceptable for parties unaffiliated with the principal that the security policy seeks to protect to learn their value, unless this was explicitly desired by the system designer. We can assume that their value is not directly visible to outsiders, for otherwise the system would be triv-

ially insecure. Public variables are all those that are assumed to be visible to outside observers. Trusted variables are those whose value is taken to be under the control of the principal; untrusted variables may have had their value influenced by outsiders whose interests may not align with those of the principal.

The presence of an untrusted variable as a cause can turn an otherwise acceptable causal relationship unacceptable. Conversely, the presence of a trusted cause can turn an otherwise unacceptable causal relationship acceptable.

Example 5.1.2. Alice's computer-illiterate boss, Bob, has asked Alice to fix his computer. While she is at it, she realises that his desktop background is the default colour (say, white). She decides to set the desktop background to her favourite colour. (For simplicity, in the remainder of the discussion, we asume that there are only two possible colours.) Consider two scenarios:

1. Alice is sensitive to the circumstance that she is working on somebody else's machine. Her understanding with Bob is that has she is entitled to change some setting (like background colour) unless Bob explitly tells her not to.
2. Alice is quite fed up with Bob's lack of taste and clueless management. If Bob were to tell her to leave the desktop background alone, she would just get spiteful and instead set it to the opposite of her favourite colour. □

Our intuition says that in the first case, the colour change was (implicitly) authorised by Bob. Were he to complain about it, Alice could rightly respond that she wouldn't have done it if he had told her not to, and he had ample opportunity to. On the other hand, Bob would not be wrong to complain about her meddling and insubordination in the second case. This is not just a matter

of control; knowing Alice's behaviour, Bob can make her set the background to any colour he prefers by tactically choosing whether to tell her to back off.

Formally, we could represent the cases as causal models M^{obedient} and M^{defiant} with a single exogenous variable whose range is a singleton, and three binary endogenous variables, A representing Alice's favourite colour, B representing whether Bob tells Alice that it is okay to change the colour, and C representing the resulting background colour. In the first, "obedient" case, we have

$$F_C^{\text{obedient}} = \begin{cases} A & \text{if } B{=}1 \\ 1 & \text{otherwise.} \end{cases}$$

On the other hand, in the "defiant" case,

$$F_C^{\text{defiant}} = \begin{cases} A & \text{if } B{=}1 \\ 1 - A & \text{otherwise.} \end{cases}$$

(F_A and F_B just set A and B to Alice and Bob's actual actions in each case.)

It is easy to check that these two models are just relabellings of the models $M^{\wedge\text{lamp}}$ and $M^{\neg\oplus\text{lamp}}$ from earlier, respectively; thus, we have $M^{\text{obedient}} \models B \rightsquigarrow (A \rightsquigarrow C)$, but $M^{\text{defiant}} \not\models B \rightsquigarrow (A \rightsquigarrow C)$. More generally, we could consider this an instance of a security policy that we could call *authorisation*: the untrusted variable A is only a cause of the privileged outcome C if this causal relationship itself had a trusted cause B, interpreted as the causation happening at B's pleasure, with B having the option to prevent it and choosing to not making use of it.

The "authorisation" construction that we have just described can be easily iterated to generate more complex meaningful examples.

Example 5.1.3. Suppose Bob is not present during Alice's fixing of his computer, and instead has told Alice to let Bob's secretary Dylan supervise her. Would

Alice listen to Dylan if he were to tell her to leave Bob's desktop background unchanged (which, in fact, he doesn't)? Once again, consider two cases:

1. Alice respects Bob's delegation of authority, and sets the desktop to her preferred background colour only if Dylan doesn't tell her to leave it alone. If Bob had instead told her not to listen to Dylan, she would have strictly acted according to her own best judgement, and set the desktop background to her preferred colour no matter what he said.
2. Alice thinks much more highly of Dylan than the boss they work for, and will listen to him even if Bob tells her not to. As it happens, Bob trusts Alice's artistic judgement much more than Dylan's, and will be quite displeased to hear that his overbearing underling stopped Alice from setting him up with an artfully chosen background.

□

In both scenarios, the final setting of the desktop background is caused by Alice's preferred colour, and this causal relationship in turn is caused by Dylan's acquiescence. What intuitively distinguishes the two scenarios (and would continue distinguishing them if Alice's defiance were to come to the fore under some combinations of Bob's and Dylan's instructions) is whether Dylan's control over this itself was "at Bob's pleasure", that is, could have been vetoed by Bob, or Bob's authority was usurped. Formally, we can capture them as two causal models, both with a single exogenous variable whose range is a single-

ton, four endogenous variables, and

$$F_C^1 = \begin{cases} A & \text{if } B = 1 \text{ and } D = 1 \\ 1 & \text{else} \end{cases}$$

$$F_C^2 = \begin{cases} A & \text{if } D = 1 \\ 1 & \text{else} \end{cases}.$$

We then find while $M^1 \vDash B \rightsquigarrow (D \rightsquigarrow (A \rightsquigarrow C))$, we have $M^2 \nvDash B \rightsquigarrow (D \rightsquigarrow (A \rightsquigarrow C))$.

This construction can be iterated further in a straightforward manner, allowing us to express any number of steps of delegation. Such chains of delegation are often considered in *authorisation logics* (see e.g. Abadi's survey [1]), but rarely given a formal semantics [20], let alone one that can be applied in a setting as general as causal models.

We have shown that a variety of security properties can be expressed as nested causal statements. To give a formal account of such statements, we extended the Halpern-Pearl framework of causality to allow formulae that may themselves refer to causal relationships as effects of causal statements. As we have shown, the language of nested causal statements thus obtained is more expressive than the language of simple causal statements that the HP framework normally deals in. This extension also led us to revisit a particular design assumption of the HP framework, namely that it is always appropriate to have causal statements refer to variables taking particular values as causes and effects. We have argued that in nested causal statements, it is often more natural to implicitly existentially quantify over values, using just the variables as causes and effects (interpreted as meaning that whatever value the variable actually

takes is the cause or effect).

Having laid the groundwork for the characterization of security properties using (nested) causality and considered some informal examples, we are now ready to take a formal look at the relation between our approach and existing formalisms from the programming language community. The examples we have considered so far all were evaluated with respect to ad-hoc causal models that we argued to represent security-relevant, but ultimately organic rather than formal, scenarios. In order to move on to analysing real programs, we must therefore first develop a means of capturing the behaviour of formally specified programs as causal models.

5.2 Representing programs as causal models

We model the systems whose security we analyse as being specified by *programs* from a set of possible programs $\mathcal{P}$. These programs operate on the contents of *memory locations*. (These memory locations are sometimes called variables, but we reserve that word here for the variables of a causal model.) We assume that the set of memory locations for a given program is finite. A *state* describes the value in each memory location at a given time.

We assume that the set of memory locations for a given program is finite. A *state* describes the value in each memory location at a given time. We assume that program execution is given by a small-step relation $\rightarrow$ which relates pairs $\langle p, \sigma \rangle$ of programs $p \in \mathcal{P}$ and states σ. In this thesis, we will also take this relation to be deterministic, so that there is in fact always a unique next state $\langle p, \sigma \rangle \rightarrow \langle p', \sigma' \rangle$, unless $p = ()$ (and so the program has terminated). We will

denote this by writing induced functions $\mathsf{next}_{\rightarrow,L}\langle p, \sigma\rangle = \sigma'(L)$ for $L \in \mathcal{M}$ and $\mathsf{next}_{\rightarrow,P}\langle p, \sigma\rangle = p'$ when it is convenient. Without the subscript specifying the variable, we can write $\mathsf{next}_{\rightarrow}\langle p, \sigma\rangle = \langle p', \sigma'\rangle$. When the initial program is fixed in the context, we sometimes omit it altogether and just write $\mathsf{next}^k_{\rightarrow}(\sigma)$ for the state that results from taking k steps starting with state σ and the initial program.

To capture this in a causal model, for each memory local X, we have variables X_0, X_1, X_2, ...in the causal model, where the value of X_t represents the contents of memory location X at time t. Thus, if $\mathcal{M}$ is the set of memory locations referenced in the program, the set of variables can be identified with $\mathcal{M}_p = \{X_t : X \in \mathcal{M}, t \leq \#(p)\}$, where $\#(p)$ is the least upper bound on the number of steps that p runs on all inputs. (We take $\#(p) = \infty$ if the running time of p is unbounded.) For simplicity, we assume that the contents of each memory location is an integer, so $\sigma : \mathcal{M} \to \mathbb{Z}$. We can then define the causal model associated with a program p operating on states $\sigma \in (\mathcal{M} \to \mathbb{Z})$ as follows.

Definition 5.2.1. Let $p \in \mathcal{P}$ be a program with semantics given by $\rightarrow$, which operates on a set of memory locations $\mathcal{M}$. Let M^p be the causal model with

- exogenous variables $\mathcal{U} = \{U_X : X \in \mathcal{M}\}$, representing the initial state of memory;
- endogenous variables $\mathcal{V} = \mathcal{M}_p \cup \{P_t : t \leq \#(p)\}$. We can think of P_t as representing the remaining program after t steps.
- $\mathcal{R}(X) = \mathcal{R}(X_0) = \mathcal{R}(X_1) = \ldots = \mathbb{Z}$ for every variable $X \in \mathcal{M}$; and for all i, $\mathcal{R}(P_i)$ is the set of all possible programs.
- $\mathcal{F}(X_0) = U_X$ for every variable $X \in \mathcal{M}$, so the initial value of X (given by X_0) is is determined by the exogenous variable corresponding to X, and $\mathcal{F}(P_0) = p$;

- The variables at time $i+1$ are computed as the result of one step taken from the values of the variables at time i: The structural equations for the variables at time $i+1$ are given by the semantics in terms of the values at time i: $\mathcal{F}(X_{i+1}) = \sigma'(X)$ and $\mathcal{F}(P_{i+1}) = p'$, where $\langle \sigma', p' \rangle = \mathsf{next}_{\rightarrow} \langle (X_i \mid X \in \mathcal{M}), P_i \rangle$. □

With a slight modification, our earlier example is such a causal model.

Example 5.2.2. Suppose that p is the program $A := A \wedge B$ (with $\wedge$ defined to be the bitwise AND on two integers), and $\langle \sigma, A := A \wedge B \rangle \rightarrow \langle \sigma[\sigma(A) \wedge \sigma(B)/A], () \rangle$. Then in the causal model M^p, we have

- $\mathcal{U} = \{U_A, U_B\}$,
  $\mathcal{V} = \{A_0, A_1, B_0, B_1, P_0, P_1\}$;
- $\mathcal{F}(A_0) = U_A$, $\mathcal{F}(B_0) = U_B$, $\mathcal{F}(A_1) = A_0 \wedge B_0$, $\mathcal{F}(B_1) = B_0$, and $\mathcal{F}(P_0) = A := A \wedge B$, $\mathcal{F}(P_1) = ()$.

In $(M^p, \{A \mapsto 1, B \mapsto 1\})$, we then have, among other things,

- $A_0 = 1$, $B_0 = 1$ and $A_1 = 1$;
- $[A_0 \leftarrow 0](A_1 = 0)$, so $A_0 \rightsquigarrow A_1 = 1$;
- $[B_0 \leftarrow 0](A_1 = 0)$, so $B_0 \rightsquigarrow A_1 = 1$. □

As we noted in the introduction, we view the objective of security as that of ensuring that an attacker can not make the system behave in a way that was not intended by the system designer. An *attacker model* describes what an attacker can do to disrupt a system. Security always has to be considered relative to an

attacker model: in the extreme case, no meaningful security is possible at all when the attacker can intervene to change any aspect of the system at any time. We identify the attacker model with a set of interventions; intuitively, these are the actions that the attacker can take. A single security property may in fact depend on the abilities of multiple attackers, or those of a single attacker acting in different capacities (e.g., observing an offline copy of the system and then acting upon the real one, or preparing an attack on the initial state and then exerting limited influence during its operation). Thus, a security policy may involve causal formulas that are relative to different sets of an interventions.

Most interesting properties of a system arise when considering its evolution from an initial state, which we represent by the assignment $\vec{u}$ to exogenous variables. We expect the system designer to specify the set of possible initial states. Attacker interventions may result in the system progressing in a way that would not have been possible in the course of its natural evolution from any admissible initial state. This may be the case even if the attacker can intervene only on the initial state (represented by an intervention $V_0 \leftarrow v'$ on an endogenous variable V_0 at time 0). Such an intervention could result in V_0 itself taking a value that the system designer never intended for it to take or, more generally, a combination of values in the initial configuration that was not part of the set of possible initial states. In that situation, an asymmetry between "regular" states (those assignments to endogenous variables that arise from some possible initial state) and "irregular" ones (those that can arise only when attacker interventions occurred) may emerge, which becomes relevant in some of the security properties we consider.

5.3 Common security properties

In this section, we review a number of common security properties and show how to interpret them in a causal framework. For the sake of clarity, we take a moment to formally define some common (but, as far as we can tell, rarely explicitly defined) terminology. For the purposes of this work, we take a *security policy* to consist of a *labelling*, which attaches certain security-relevant labels to parts of the system (such as saying that a certain piece of data is secret), and *security properties*, formulas written in some *policy language* that say that certain things should or should not happen, with reference to the labels: for example, saying that secrets should not leak to the public. Each security property itself can be interpreted as a subset of systems that satisfy it; Boolean combinations of properties can then be canonically interpreted using set operations.

A security policy is intended to bridge the gap between the system designer's intuitions about the security desiderata of a system and the unwieldy object that is the set of secure systems. We thus want the language to be both rich (so that it can express many intuitions) and simple (so that the system designer can easily anticipate the interpretation of a given policy). The primary objective of this thesis is to show that many existing security properties (specified in different policy languages) can be easily and intuitively expressed in terms of causality, and, moreover, that it is easy to find causal formulae that directly capture security-related intuitions.

Many of the security policy that we consider assume that the labeling associates each memory location $V \in \mathcal{M}$ with a *security label* $\Gamma(V) \in \Lambda$. The set Λ forms a lattice under the relation $\sqsubseteq$. We can naturally extend the labelling

function to the memory-representing variables of the causal model by taking the value of Γ on a causal model variable $W_k \in \mathcal{M}_p$ to be its value on the corresponding memory location W. (Γ is not defined on program variables P_i.)

Often, Λ is taken to be the product of two separate lattices of labels, called the confidentiality and integrity lattices, with corresponding relations $\sqsubseteq_C$ and $\sqsubseteq_I$. For many purposes, it is sufficient for each of these lattices to only consist of two related points called, respectively, public $\sqsubseteq_C$ secret and trusted $\sqsubseteq_I$ untrusted.

5.3.1 Noninterference, confidentiality, and integrity

A basic security property that uses this lattice corresponds to the idea that secret information should never be revealed to the public and untrusted information should never influence information that is trusted. To express it in the causal setting, we first model the attacker as follows.

Definition 5.3.1. The *time-0 attacker* $\mathcal{A}_0$ is the set of all interventions that only affect endogenous variables at time 0 (e.g. V_0). □

Our formal security property then says that an attacker with available actions described by $\mathcal{A}_0$ can not act upon any variable to cause an effect "further up" the lattice $\sqsubseteq$.

Definition 5.3.2. A program p satisfies *causal noninterference* if, for all initial states $\vec{u}$, whenever a variable $V_0 \in \mathcal{M}_p$ is a cause of some $W \in \mathcal{M}_p$ in $(M^p, \vec{u})$ with respect to $\mathcal{A}_0$, we have $\Gamma(V_0) \sqsubseteq \Gamma(W)$. ("Influence never flows down.") □

The special cases when $\sqsubseteq = \sqsubseteq_C$ or $\sqsubseteq = \sqsubseteq_I$ are also referred to as *confidentiality* and *integrity*, respectively. It is standard in the literature to view confidentiality

and integrity as symmetric properties; from that perspective, the fact that we are using the same attacker model seems for both seems reasonable. Moreover, for integrity, it is easy to interpret the model: we do not want the attacker to be able to affect the system by tampering with some untrusted memory before the program runs. With confidentiality, things may seem less clear. Indeed, it may seem strange to view the attacker as trying to modify (at time 0) information that is supposedly secret, the very information that he is trying to learn. We can explain this by thinking of an attacker as working offline on a system, locally simulating the effects of giving the system various secret inputs, to see how the public outputs change. If the attacker can detect a pattern, it will then be able to learn something from the public outputs observed when the system runs on actual secret data.

We can compare Defintion 5.3.2 to the following standard (non-causal) definition of noninterference:

Definition 5.3.3. For a security label ℓ, define the *ℓ-view* $\mathsf{view}_\ell(\vec{u})$ of a state of memory $\vec{u}$ to be the subset of memory locations that are "below" ℓ, that is, the set V such that $\Gamma(V) \sqsubseteq \ell$. A program satisfies *noninterference* if, for all labels ℓ and pairs of initial states of memory $\vec{u}$, $\vec{u}'$, if the initial ℓ-views agree ($\mathsf{view}_\ell(\vec{u}) = \mathsf{view}_\ell(\vec{u}')$), this remains true after any number of steps:

$$\forall n \in \mathbb{N}.\, \mathsf{view}_\ell(\mathsf{next}^n_\rightarrow(\vec{u})) = \mathsf{view}_\ell(\mathsf{next}^n_\rightarrow(\vec{u}')).$$

□

It is not hard to prove the two definitions equivalent.

Proposition 5.3.4. *A program p satisfies causal noninterference (Def. 5.3.2) iff it satisfies noninterference (Def. 5.3.3).*

Proof. Suppose that a program violates Definition 5.3.2. Then there exist variables V_0 and W_k in $\mathcal{M}_p$, $\Gamma(V) \not\sqsubseteq \Gamma(W)$, values $v \neq v', w \neq w'$ and an initial state of memory $\vec{u}$ such that we have $(M^p, \vec{u}) \models W_k = w$, $(M^p, \vec{u}) \models V_0 = v$, but $(M^p, \vec{u}) \models [V_0 \leftarrow v'](W_k = w')$. Take $\vec{u}'$ to be $\vec{u}$ with its entry corresponding to V replaced by v'. Since $\Gamma(V) \not\sqsubseteq \Gamma(W)$, the V-entry is not included in $\mathsf{view}_{\Gamma(W)}$, so $\mathsf{view}_{\Gamma(W)}(\vec{u}) = \mathsf{view}_{\Gamma(W)}(\vec{u}')$. Denote the vector of values taken by V_i ($V \in \mathcal{M}$) in $(M^p, \vec{u})$ by $\vec{u}_i$, and the corresponding vector in $M^p_{V_0 \leftarrow v'}$ by $\vec{u}'_i$. By the structural equations in both models, we have $\vec{u}_i = \mathsf{next}^i_{\rightarrow}(\vec{u}_0)$, $\vec{u}'_i = \mathsf{next}^i_{\rightarrow}(\vec{u}'_0)$. But $\vec{u}_0 = \vec{u}$, $\vec{u}'_0 = \vec{u}'$. However, $\Gamma(W) \sqsubseteq \Gamma(W)$, and we know the W-entries of $\vec{u}_k$ and $\vec{u}'_k$ are $w \neq w'$, respectively. So $\mathsf{view}_{\Gamma(W)}(\mathsf{next}^k_{\rightarrow}(\vec{u})) \neq \mathsf{view}_{\Gamma(W)}(\mathsf{next}^k_{\rightarrow}(\vec{u}'))$, and Def. 5.3.3 is violated.

Conversely, suppose that a program violates Definition 5.3.3; specifically, suppose that $\mathsf{view}_\ell(\vec{u}) = \mathsf{view}_\ell(\vec{u}')$ and $\mathsf{view}_\ell(\mathsf{next}^k_{\rightarrow}(\vec{u})) \neq \mathsf{view}_\ell(\mathsf{next}^k_{\rightarrow}(\vec{u}'))$ for some k, ℓ, $\vec{u}$, $\vec{u}'$. Let $\vec{u} = \vec{u}^0, \vec{u}^1, \ldots, \vec{u}_m = \vec{u}'$ be such that $\vec{u}^i$ and $\vec{u}^{i+1}$ differ in only one memory location for all $i < m$, and $\mathsf{view}_\ell(\vec{u}^i)$ is the same for all i. This chain must be finite, since we assume a finite number of memory locations. By transitivity of equality, there must be some i such that $\mathsf{view}_\ell(\mathsf{next}^k_{\rightarrow}(\vec{u}^i)) \neq \mathsf{view}_\ell(\mathsf{next}^k_{\rightarrow}(\vec{u}^{i+1}))$. Let W be some memory location on which $\mathsf{view}_\ell(\mathsf{next}^k_{\rightarrow}(\vec{u}^i))$ and $\mathsf{view}_\ell(\mathsf{next}^k_{\rightarrow}(\vec{u}^{i+1}))$ differ, taking values w and w', respectively. We must have $\Gamma(W) \sqsubseteq \ell$. Let V be the unique memory location on which $\vec{u}^i$ and $\vec{u}^{i+1}$ differ, say $\vec{u}^i(V) = v$, $\vec{u}^{i+1}(V) = v'$. If we had $\Gamma(V) \sqsubseteq \Gamma(W)$, then by transitivity, $\Gamma(V) \sqsubseteq \ell$, and so V is visible in the ℓ-view. But then $\mathsf{view}_\ell(\vec{u}^i) \neq \mathsf{view}_\ell(\vec{u}_{i+1})$, contradicting the construction of the $\vec{u}^i$. Hence $\Gamma(V) \not\sqsubseteq \Gamma(W)$. But, by the structural equations in the causal model, $(M^p, \vec{u}^i) \models W = w$, and $(M^p, \vec{u}^i) \models [V_0 \leftarrow v'](W = w')$. So $(M^p, \vec{u}^i) \models V_0 \rightsquigarrow W$, and Def. 5.3.2 is violated. □

While this property is simple and straightforward, few interesting programs satisfy noninterference: what's the point of keeping secrets if they can't influence your public behaviour, or maintaining trusted state if that state cannot reflect untrusted wider reality in any way? In particular, none of the following types of programs satisfy noninterference:

(1) Programs that declassify any information at all, such as a "timed-release" mechanism where we only care that a secret is not leaked until a certain point in time.

(2) Password checkers, where some secret information is used in a controlled fashion to compute a public "verdict" that may depend on public data.

(3) Programs where some data is supposed to reflect untrusted user input in a controlled fashion, such as poll systems (a user should be able to add one vote for option A or B, but not more than that) or credit systems such as bank accounts (a user should be able to cause some money to be transferred out of their account, but not adjust their balance at will).

Perhaps unsurprisingly, various refinements of noninterference have been proposed in the literature. We consider some of them here.

5.3.2 Robust Declassification

The first refinement we consider is due to Myers and Zdancewic [44]. For a program of type (1), demanding noninterference is too much, as information being disclosed necessarily violates confidentiality. However, at the minimum, we could demand that an attacker with a particular set of abilities to interfere

with the system can not influence *what* is disclosed at a given time k. Importantly, though, an attacker is free to influence *how* a secret value is disclosed at time k. For example, a program like $\mathsf{if}\ A = 1\ \mathsf{then}\ P := S\ \mathsf{else}\ Q := S$, with P and Q being public, S being secret, and A being attacker-controlled, would be secure; we could interpret this as a representation of a scenario where the "attacker" journalist gets to choose if a government disclosure request is satisfied by fax or email. This notion is called *robustness*.

We define robustness causally by giving an account of what exactly it means to have a violation of it, with a particular focus on *robust declassification*: the property that an attacker can not cause a system to declassify more information than intended. (The intuition remains broadly unchanged for integrity.) As we argued in the previous section, a secret contained in a variable V is *disclosed* via a variable W if W is public and the value of V affects (i.e., has a causal impact on) the value of W, that is, $V \rightsquigarrow_{\mathcal{A}_0} W$. Normally, this would constitute a violation of confidentiality, but here we want to stipulate that such a disclosure can be considered to have been intended by the system designer, as long as it happens during a regular run starting at a possible initial state with no attacker interventions. The statement that V was disclosed *in some way* at time k then is naturally represented by $\exists W_k.\, V \rightsquigarrow_{\mathcal{A}_0} W_k$.

Naively, we might then formalise the circumstance that this disclosure was influenced by an attacker by writing $U \rightsquigarrow_{\mathcal{I}} (\exists W_k.\, V \rightsquigarrow_{\mathcal{A}_0} W_k)$. This says that the causal relationship held because of the setting of an (untrusted) variable U by an attacker modelled by intervention set $\mathcal{I}$. This definition turns out to be adequate to capture the notion of robust declassification in [44]. However, if we inspect the definition of causality, we find that this statement formally means that (1)

V was disclosed, and (2) the attacker modelled by $\mathcal{I}$ could have intervened to prevent this. In other words, all we have established is that an attacker intervention could have caused a disclosure that normally would have happened to not happen. Is this a problem? Myers et al. [30] argue that it may not be: after all, the attacking party can make things worse only for themselves by preventing a disclosure that would normally occur from happening. Such an attacker is called *incompetent*. It therefore is actually more appropriate to consider the converse scenario: namely, one where a variable is *not* disclosed, but this is only because an attack that was available to the attacker was not performed. In other words, the attacker could have made a disclosure happen. This is one scenario in which the previously mentioned asymmetry between regular and attacker-induced states becomes relevant: if all attacker interventions have the effect of only switching between different permissible initial states, then there is no difference between the two approaches, as every context in which the attacker could prevent a disclosure is paired with one in which the attacker could cause a disclosure that normally wouldn't happen to happen. It is only when the attacker action that gives rise to the disclosure causes a transition to an "irregular" state that the difference between inept attackers (who learn less in the "irregular" scenario) and competent ones (who learn more) becomes relevant. This leads us to the following definition.

Definition 5.3.5. A program p satisfies *causal robustness* with respect to an attacker model $\mathcal{I}$ if for all initial states $\vec{u}$, times k, pairs of variables $U, V \in \mathcal{M}_p$, and sets S of interventions, it is not the case that

$$(M^p, \vec{u}) \vDash U \rightsquigarrow_{\mathcal{I}} (\neg\exists W : \Gamma(V) \not\subseteq \Gamma(W).$$
$$V \rightsquigarrow_{\mathcal{A}_0|S} W_k).$$

Just as we took existential quantification over a finite set of variables to be short-

hand for disjunction, we take $\exists W_k : \Gamma(V) \not\sqsubseteq \Gamma(W_k)$ to represent a disjunction over those variables W for which $\Gamma(V) \not\sqsubseteq \Gamma(W)$ holds. It follows that if the value of V affects the value of some *some* variable W_k at time k in a way that amounts to a violation of noninterference, this causal relationship may not be caused by an attacker action.[1] When, in addition, $\sqsubseteq$ refers only to the confidentiality component of security, we also call the resulting notion *causal robust declassification*. □

The additional quantification over S is required because *what* is disclosed may be more specific than the entirety of what is encoded in the value of a variable. For instance, it may be that the attacker can exercise control only over whether the third bit of an integer variable is disclosed. In that scenario, the secret value V may always be a cause of a public observation W_k, as changing its second bit would produce an observable change, but depending on the attacker action, a change to the third bit may or may not produce one, so we would see a difference in whether $V \rightsquigarrow_{\mathcal{A}_0|(V\in\{0,4\})} W_k$ is true. We could also interpret the role of S as conditioning in the probabilistic sense, as we did when introducing the conditioning of intervention sets: the notion of "the attacker may not cause more information to be disclosed" should hold conditional on any amount of knowledge the attacker already may have, including, for instance, the knowledge that $V \in \{0, 4\}$.

In many cases, we can simply take the set $\mathcal{I}$ of attacker interventions to consist of all the interventions that change the initial value of some attacker-controlled ("untrusted") variable. Doing so here results in a notion that is more similar to noninterference. However, it can in fact be proven [4] that if the

[1] Note that the existential quantification can be replaced by a disjunction over a finite number of statements of the form $V \rightsquigarrow^i W$, so the language does not need to be extended.

programming language is sufficiently expressive, this attacker model is no less powerful than one in which attackers can intervene in some fashion later during the execution. Intuitively, the proof proceeds by augmenting the program with attacker code that implements attacker instructions to perform particular actions (possibly depending on the state of the system), which are given from the start as part of untrusted memory, at the appropriate time.

This definition can in fact capture the security of a program that performs an intended disclosure, while flagging one where this disclosure is controllable by an attacker as insecure.

Example 5.3.6. Consider the program $B := A$, where $\Gamma(A) \not\sqsubseteq \Gamma(B)$. The causal model for this program has 4 endogenous variables A_0, A_1, B_0, B_1. This program does not satisfy (causal) noninterference (because $A_0 \rightsquigarrow B_1$), but it does satisfy causal robustness with respect to all attacker models. □

On the other hand, consider the example, first introduced by Zdancewic and Myers [44], of a password checker that an attacker might induce to copy some secret piece of information into the reference password that the attacker can then guess.

Example 5.3.7. (Adapted from [44]) Consider the system described by the program

$$\mathsf{if}\ C = 1\ \mathsf{then}\ P := S;$$

$$\mathsf{if}\ P = Q\ \mathsf{then}\ R := 1\ \mathsf{else}\ R := 0;$$

where $\Gamma(S) \not\sqsubseteq \Gamma(R)$. This program does not satisfy robustness with respect to the attacker model $\mathcal{I} = \{C \leftarrow 1, C \leftarrow 0\}$: we have

$$(M_p, C = 0) \models C \rightsquigarrow_{\mathcal{I}} (S \rightsquigarrow R). \qquad \square$$

As with noninterference, we can prove that causal robustness is equivalent to a standard notion. In this particular case, we will use the following formulation. For convenience, we adopt the simplified assumption that the attacker controls only initial states of memory, which is justified by Cecchetti, Myers, and Arden [4], but follow an older formulation of Zdancewic and Myers [44] in omitting some aspects of Cecchetti, Myers, and Arden's definition that pertain to the type system that they introduce in the paper.

Definition 5.3.8. (Adapted from [4, Def. 6.5]) A program satisfies *robust declassification* with respect to an attacker associated with an "untrusted" label ℓ' and a "public" label ℓ if, for all initial states of memory $\vec{u}$, memory locations U and V such that $\Gamma(U) = \ell'$ and $\Gamma(V) \not\sqsubseteq \ell$, and values $u^1, u^2 \in \mathcal{R}(U)$, $v^1, v^2 \in \mathcal{R}(V)$, if $\vec{u}^{ij}$ is the assignment $\vec{u}$ with U set to u^i and V set to v^j, where u^2 is a value that the attacker can write to U, then

$$\forall n \in \mathbb{N}.\mathsf{view}_\ell(\mathsf{next}^n_{\rightarrow}(\vec{u}^{11})) = \mathsf{view}_\ell(\mathsf{next}^n_{\rightarrow}(\vec{u}^{21}))$$
$$\Rightarrow \mathsf{view}_\ell(\mathsf{next}^n_{\rightarrow}(\vec{u}^{12})) = \mathsf{view}_\ell(\mathsf{next}^n_{\rightarrow}(\vec{u}^{22})). \quad \square$$

Intuitively, this definition captures the intuition that whenever two runs differ only by an attacker action, then the attacker does not gain information, although he may lose information that he would have otherwise gained. As stated earlier, the implicit attacker model is one where the attacker can control only the initial values of the "untrusted" variables, that is, those with a label $\sqsubseteq \ell'$, and observe the values of "public" variables whose label is $\sqsubseteq \ell$. At first glance this seems fairly restrictive. However, by an argument similar to the one in the appendix of [4], it actually has the same generality as the much more general definition of [30], which gives the attacker the ability to inject code into a specially marked *hole* in the code. This view-based definition is equivalent to the causal

one, if we pick the appropriate causal representation of the attacker model.

Theorem 5.3.9. *A program p satisfies causal robustness (Def. 5.3.5) with respect to the interference attacker $\mathcal{I} = \{X_0 \leftarrow v \mid \Gamma(X) = \ell', v \in \mathcal{R}(X_0)\}$ and $\sqsubseteq$ iff it satisfies robust declassification (Def. 5.3.8) with respect to an attacker associated with ℓ' and the "public, trusted" label $\ell \in \mathrm{dom}\ \sqsubseteq$.*

Proof. Suppose that a program violates Def. 5.3.5. Then there must be a context $(M^p, \vec{u})$ where the formula of that definition is true. So there must exist variables U_0 and V_0, values $u^1 \neq u^2$ of U_0, a time k, and a set S of interventions such that $(M^p, \vec{u}) \vDash U_0 = u^1$ and $(M^p, \vec{u}) \vDash [U_0 \leftarrow u^2](\exists W_k : \Gamma(V_0) \not\sqsubseteq \Gamma(W).V_0 \rightsquigarrow_{\mathcal{A}_0|S} W_k)$. In addition, $\Gamma(U) \sqsubseteq \ell'$, and $U_0 \leftarrow u^2 \in \mathcal{I}$, that is, assigning u^2 is a possible attack. Let W_k be chosen to witness the existential quantification, so $\Gamma(V) \not\sqsubseteq \Gamma(W)$. Thus, there must be values $w^1 \neq w^2$ for W_k and $v^1 \neq v^2$ for V_0 such that

1. $(M^p, \vec{u}) \vDash W_k = w^1 \wedge V_0 = v^1$;
2. $(M^p_{U_0 \leftarrow u^2}, \vec{u}) \vDash [V_0 \leftarrow v^2](W_k = w^2)$, that is, after the attack, the value of V_0 leaks in W_k;
3. for all W'_k with $\Gamma(V) \not\sqsubseteq \Gamma(W')$, there is a corresponding value w' such that for all $v' \in \mathcal{R}(V_0)$, $(M^p, \vec{u}) \vDash [V_0 \leftarrow v'](W'_k = w')$, that is, W'_k has that value regardless of how V_0 is set.

Note that by definition of the causal model M^p, the W-entry of $\mathsf{next}^k_{\rightarrow}(\vec{u})$ is the value of W_k in $(M^p, \vec{u})$, and intervening on an endogenous variable at time 0 has the same effect on the values of endogenous variables as as changing the exogenous variable that it would otherwise get its value from. So, taking $\vec{u}^{ij}$ to be as in Def. 5.3.8 with $U \in \{u^1, u^2\}$, $V \in \{v^1, v^2\}$, we have

$$\mathsf{view}_\ell(\mathsf{next}^k_{\rightarrow}(\vec{u}^{12})) \neq \mathsf{view}_\ell(\mathsf{next}^k_{\rightarrow}(\vec{u}^{22})),$$

since the two views differ on the value of W (w^1 vs. w^2), but

$$\mathsf{view}_\ell(\mathsf{next}^k_{\rightarrow}(\vec{u}^{11})) = \mathsf{view}_\ell(\mathsf{next}^k_{\rightarrow}(\vec{u}^{21})),$$

since the two views agree on the value of all memory locations W' that are in view_ℓ. It follows that $\Gamma(W') \sqsubseteq \ell$ for all W' in the view. This is because, by (3) above, the two views agree on all W' such that $\Gamma(V) \not\sqsubseteq \Gamma(W')$, and this is actually a superset of $\{W' : \Gamma(W') \sqsubseteq \ell\}$: if $\Gamma(W') \sqsubseteq \ell$ and $\Gamma(V) \sqsubset \Gamma(W')$, then by transitivity, $\Gamma(V) \sqsubseteq \ell$, contradicting the assumption that V is secret but ℓ is a public label.

Conversely, suppose a program violates Def. 5.3.8. Then we have four initial configurations $\{\vec{u}^{ij}\}_{i,j\in\{1,2\}}$ generated from a single assignment $\vec{u}$ by setting the values of memory locations U and V to $\{u^1, u^2\}$ and $\{v^1, v^2\}$, respectively, and a time k such that

$$\mathsf{view}_\ell(\mathsf{next}^k_{\rightarrow}(\vec{u}^{11})) = \mathsf{view}_\ell(\mathsf{next}^k_{\rightarrow}(\vec{u}^{21}))$$

and

$$\mathsf{view}_\ell(\mathsf{next}^k_{\rightarrow}(\vec{u}^{12})) \neq \mathsf{view}_\ell(\mathsf{next}^k_{\rightarrow}(\vec{u}^{22})).$$

We claim that for $S = \{V_0 \leftarrow v^1, V_0 \leftarrow v^2\}$, we have

$$(M^p, \vec{u}^{11}) \vDash U_0 \rightsquigarrow_I (\neg\exists W_k : \Gamma(V_0) \not\sqsubseteq \Gamma(W_k).$$
$$V_0 \rightsquigarrow_{\mathcal{A}_0|S} W_k).$$

for a particular k as required. Indeed, if W^* is an arbitrary entry on which $\mathsf{view}_\ell(\mathsf{next}^k_{\rightarrow}(\vec{u}^{12}))$ and $\mathsf{view}_\ell(\mathsf{next}^k_{\rightarrow}(\vec{u}^{22}))$ differ, then $\Gamma(W^*) \sqsubseteq \ell$ because it is in the view, but by assumption of Def. 5.3.8, $\Gamma(V) \not\sqsubseteq \ell$, and hence $\Gamma(V) \not\sqsubseteq \Gamma(W^*)$ by the contrapositive of transitivity. So we have $(M^p, \vec{u}^{12}) \vDash V_0 \rightsquigarrow W^*_k$, and $\Gamma(V_0) \not\sqsubseteq \Gamma(W^*_k)$. Also, letting w' be the value of each memory location W' in $\mathsf{next}^k_{\rightarrow}(\vec{u}^{11})$, we have $(M^p, \vec{u}^{11}) \vDash W'_k = w'$ and $(M^p, \vec{u}^{21}) \vDash W'_k = w'$ for all variables W' in the view (thus having $\Gamma(W') \sqsubseteq \ell$). So $(M^p, \vec{u}^{11}) \vDash [V_0 \leftarrow v^2](W'_k = w')$.

Since $(M^p, \vec{u}^{11}) \models V_0 = v^1$ and S was chosen so as to exclude any counterfactuals assigning values to V_0 other than v^1 and v^2, we can conclude that

$$(M^p, \vec{u}^{11}) \models \neg V_0 \rightsquigarrow_{\mathcal{A}_0|S} W'_k$$

for all W' having $\Gamma(V_0) \not\subseteq \Gamma(W'_k)$. Since setting $U_0 \leftarrow u^2$ takes us from $(M^p, \vec{u}^{11})$ to a world in which all endogenous variables have values as in $(M^p, \vec{u}^{12})$, and we have shown that the value of V_0 does not leak through any public W'_k in the former but leaks through W^*_k in the latter, we conclude that causal robustness is indeed violated as claimed. □

As we can see, neither the sort of programs outlined in (2) nor in (3) can satisfy robustness. The violation in the poll example does not even concern declassification; for the password checker example, the information that is declassified is whether the secret password matches a particular password of the untrusted user's choosing, something that is clearly controlled by the user. A proof of equivalence to the runs-and-systems based notion in the original paper proceeds along much of the same lines as the proof of Proposition 5.3.4.

To deal with programs in (2), Myers and Zdancewic [30] introduce a further refinement that they call *qualified robust declassification*. The idea there is simply that, for particular instances of low-integrity variables controlling disclosures (high-confidentiality variables affecting low-confidentiality variables), we want to enable the system designer to assert, via a dedicated syntactic construct, that this particular instance is not in fact to be considered a violation. MZ call this construct *endorsement*. In our framework, endorsement is best understood as a special case of a different class of security properties, generally referred to as *intransitive noninterference*. We consider intransitive interference next, developing machinery that will be useful for endorsement in the process.

5.3.3 Causal paths and intransitive noninterference

Not all natural security policies can be expressed using a transitive relation on labels of memory locations. Consider the following two policies:

Example 5.3.10. (a) We have a "secret" memory location U, a "public" memory location W, and a memory location U controlled by a "declassification controller" V. The declassification controller should have full discretion to declassify secret information, but care needs to be taken that information does not leak from U to W in any other way. So the program $V := U; W := V$ should be considered secure, but the program $W := U$ should not.

(b) We have two independent processes respectively operating on memory locations U and W, which should not be allowed to influence each other's data. However, both of them are allowed to operate on a shared memory location V, say, the memory of a graphics card used to accelerate part of the computation. Care needs to be taken that no information leaks from one process to the other through this shared memory location. The program $V := U; V := f(V); U := V; V := W; V := f(V); W := V$ should then be considered secure, but not the program $V := U; W := V$. □

It is easy to convince oneself that neither of these two policies can be captured using Definition 5.3.2 with a transitive relation $\sqsubseteq$ on security labels: in both cases, declaring the first program secure requires setting $\Gamma(U) \sqsubseteq \Gamma(V)$ and $\Gamma(V) \sqsubseteq \Gamma(W)$, as we have $U_i \rightsquigarrow V_j$ and $V_k \rightsquigarrow W_l$ for some i, j, k, l in each. However, by transitivity, this implies $\Gamma(U) \sqsubseteq \Gamma(W)$. Hence, in both cases the second program is also declared secure, since if $(M^p, u) \vDash X \rightsquigarrow Y$, then $X \rightsquigarrow Y$ is either of the form $U_i \rightsquigarrow V_j$, $V_i \rightsquigarrow W_j$, $U_i \rightsquigarrow W_j$, or $X_i \rightsquigarrow X_j$ for $X \in \{U, V, W\}$.

If we want to formalise either case using a definition that resembles 5.3.2, we clearly need to drop the transitivity requirement on $\sqsubseteq$. In fact, this turns out to be sufficient for Example 5.3.10(b): it is easily verified that if we define an intransitive relation $\sqsubseteq$ such that $\Gamma(U) \sqsubseteq \Gamma(V) \sqsubseteq \Gamma(U)$ and $\Gamma(W) \sqsubseteq \Gamma(V) \sqsubseteq \Gamma(W)$, but $\Gamma(U)$ and $\Gamma(W)$ are not related, and then apply Definition 5.3.2, we obtain the desired judgements about program security (as, in particular, $\Gamma(U) \not\sqsubseteq \Gamma(W)$, but U_1 is a cause of W_j for some j in the second program of the example). Essentially, this definition *blacklists* all causal relationships between variables that are not related by $\sqsubseteq$ to each other. However, this approach does not work for 5.3.10(a): in order to declare $W := U$ insecure, we would still require $\Gamma(U) \not\sqsubseteq \Gamma(W)$, but as we have seen, this results in $V := U; W := V$ being declared insecure as well.

To handle Example 5.3.10(a), we instead need an approach that is closer to *whitelisting*. We clearly want $\Gamma(U) \sqsubseteq \Gamma(V)$ and $\Gamma(V) \sqsubseteq \Gamma(W)$; otherwise, the programs $V := U$ and $W := V$ will have no chance of being declared secure. We don't want $\Gamma(U) \sqsubseteq \Gamma(W)$, lest $W := U$ be declared secure. Note that the program $V := U; W := V$, which we take to be secure, satisfies the causal formula $U_0 \rightsquigarrow W_2$, while the insecure program $W := U$ has satisfies a similar causal formula, $U_0 \rightsquigarrow W_1$. We thus need some way of distinguishing the "direct" causation in $W := U$ from the "mediated" causation that we get with $V := U; W := V$. We also need to distinguish the latter from any alternative program that leaked the contents of U to W via a location V' that is not a legitimate declassification controller. To that end, we make the following definition.

Definition 5.3.11. Let $\vec{W}$ and $\vec{V}$ be disjoint vectors of variables. Then $\vec{V} = \vec{v}$ is *a cause of φ by way of $\vec{W}$* in $(M, \vec{u})$, denoted $(M, \vec{u}) \models (V = v \overset{\vec{W}}{\rightsquigarrow} \varphi)$ or just $(M, \vec{u}) \models (V \overset{\vec{W}}{\rightsquigarrow} \varphi)$, if $\vec{V} \leftarrow \vec{v}$ is an allowed intervention, and

- $(M, \vec{u}) \models \vec{V} = \vec{v}$ and $(M, \vec{u}) \models \varphi$;
- $\exists \vec{v}'.\ (M_{\vec{V} \leftarrow \vec{v}'}, \vec{u}) \models \neg\varphi$; (i.e., $V = v$ is actually a cause of φ)
- If $\vec{w}^*$ is the actual value of $\vec{W}$ (i.e. $(M, \vec{u}) \models \vec{W} = \vec{w}^*$), then fixing it at its actual value breaks the causal relationship: that is, $\forall \vec{v}'.\ (M_{\vec{V} \leftarrow \vec{v}', \vec{W} \leftarrow w^*}, \vec{u}) \models \varphi$.

As before, we let $(V \overset{\vec{W}}{\rightsquigarrow} U)$ abbreviate $\exists v, u.\ (V = v \overset{\vec{W}}{\rightsquigarrow} U = u)$. □

Using this definition recursively, we can then define a security property that captures the idea that causal relationships are acceptable whenever they are compositions of causal relationships that are:

Definition 5.3.12. A program p satisfies *(causal) intransitive noninterference* with respect to a relation $\sqsubseteq$ if, whenever $(M^p, \vec{u}) \models \vec{U} \rightsquigarrow V$, for each $U \in \vec{U}$, the pair (U, V) is *made up of permitted flows*, defined recursively as:

- If $\Gamma(U) \sqsubseteq \Gamma(V)$, then (U, V) is made up of permitted flows.
- If $(M^p, \vec{u}) \models (U \overset{W_1, \ldots, W_n}{\rightsquigarrow} V)$ and each of $(U, W_1), \ldots, (U, W_n), (W_1, V), \ldots, (W_n, V)$ is made up of permitted flows, then (U, V) is made up of permitted flows. □

Since we wrote $\rightsquigarrow$ with no subscript in this definition, that means that, for intransitive interference, we take the set $\mathcal{I}$ of allowed interventions to include all interventions of the form $X_i \leftarrow v$: that is, the attacker is taken to be capable of interfering with the contents of memory at all times during the system's execution. We believe that intransitive interference is equivalent to a security property from the literature that turns out to assume this attacker model (see the conjecture below), although we have not yet proved this.

It is then an easy check that this notion captures the requirements of Example 5.3.10(a) with the appropriate relation $\sqsubseteq$. This is illustrated in the following example.

Example 5.3.13. Take $\sqsubseteq$ to be the reflexive relation such that $\Gamma(U) \sqsubseteq \Gamma(V)$, $\Gamma(V) \sqsubseteq \Gamma(W)$, and no other labels are related. Let p be the program $V := U; W := V$. Then the model M^p has endogenous variables $\{U_i, V_i, W_i \mid i \in \{0, 1, 2\}\}$, with $V_1 = U_0$, $W_2 = V_1$ and all other variables X_i having $X_i = X_{i-1}$. This program satisfies intransitive noninterference: we have $(M^p, \vec{u}) \vDash U_0 \rightsquigarrow W_2$ for all $\vec{u}$, but the pair (U_0, W_2) is made up of permitted flows, as we in fact have $(M^p, \vec{u}) \vDash (U_0 \overset{V_1}{\rightsquigarrow} W_2)$, and (U_0, V_1) and (V_1, W_2) are both made up of permitted flows, as $\Gamma(U_0) \sqsubseteq \Gamma(V_1)$, $\Gamma(V_1) \sqsubseteq \Gamma(W_2)$. All other causal relationships are trivially made up of permitted flows as well.

On the other hand, the program $q = W := U$ does not satisfy intransitive noninterference: in M_q, we have $W_1 = U_0$ and hence $(M^q, \vec{u}) \vDash U_0 \rightsquigarrow W_1$, but we do not have $\Gamma(U_0) \sqsubseteq \Gamma(W_1)$ and there is no variable X such that $(M^q, \vec{u}) \vDash (U_0 \overset{X}{\rightsquigarrow} W_1)$. □

However, causal intransitive noninterference, cannot be used to capture Example 5.3.10(b): since we require that both $V := U$ and $W := V$ are individually viewed as secure, and hence must be permitted, and any flow that is made up of permitted flows is again permitted, the program $V := U; W := V$ in Example 5.3.10(b) would be considered secure as well. So, to summarize, causal noninterference (with a non-transitive relationship) handles Example 5.3.10(b), but not Example 5.3.10(a), while intransitive noninterference handles Example 5.3.10(a), but not Example 5.3.10(b). We could handle both simultaneously by using the conjunction of the two security properties (with different lattice orderings $\sqsubseteq$ in

each). This would amount to constructing a security property that, rather than operating with a single set of "good" flows defined in the policy, would define both "good" flows (which make any flow that is composed from them permissible as in intransitive noninterference, unless the resulting flow is "bad") and "bad" flows (which are always considered a security violation, regardless of how they are composed).

While we are not aware of a precedent for the intransitive version of Definition 5.3.2, we believe that Definition 5.3.12 actually turns out to correspond to the notion of *IP-security* of Haigh and Young [15] (see also van der Meyden's treatment in [39]). Like our notion, IP-security can capture example 5.3.10(a), but not 5.3.10(b). IP-security from a quite different tradition of representing security policies than those that represent program runs as a sequence of state transitions and use security labels on those transitions rather than the memory they operate on. To show that our notion is equivalent, we would have to do a lot of work just to show how a security policy with labels on memory can be translated into that framework. The details of this are beyond the scope of the thesis, but we sketch a possible approach in Section 5.3.5.

5.3.4 Endorsement via intransitive noninterference

Myers and Zdancewic define endorsement as a syntactic construction of the form

$$V := \mathsf{endorse}(W, \ell),$$

where V and W are names of memory locations and ℓ is a security label. This is meant to be read as "copy the contents of W into V, endorsing them for label

ℓ''. Endorsement is peculiar in that it could really be considered a part of the security policy, rather than the specification of the program's behaviour, even though it is stated as part of the program: operationally, the above program is completely equivalent to $V := W$. The meaning of the implied policy is the following: suppose that $\Gamma(W) \not\sqsubseteq \Gamma(V)$, so normally, if the value of W is a cause of V having a particular value, this is a violation of non-interference. Then, $V := W$ is indeed considered an insecure program, as expected. On the other hand, the operationally identical $V := \mathsf{endorse}(W, \Gamma(V))$ is secure, by definition. By way of contrast, $V := \mathsf{endorse}(W, \Gamma(V)); V := W$ is, by fiat, still insecure – endorsement is *not* taken to be a *carte blanche* assertion that the contents of W may cause a particular value to manifest in a variable with label $\Gamma(V)$.

Instead of trying to interpret this feature of our policy language directly, we can actually interpret it by rewriting it in terms of existing primitives, while also rewriting the program (so the programs p that satisfy the policy featuring endorsement are then taken to be those where the corresponding rewritten program p' satisfies the rewritten policy). Specifically, we will use intransitive noninterference. In fact, the idea behind endorsement is quite similar to Example 5.3.10(a): the syntactic construction of endorsement can be seen as saying that the contents of a secret memory location are passed by a "declassification controller" into a more public location, or dually that an "endorsement controller" copies untrusted data into a trusted memory location. To interpret the endorse operation in a given security policy, it is therefore sufficient to rewrite the program with the "declassification/endorsement controller" represented by a special variable as in Example 5.3.13.

To do this formally, we first introduce special dummy memory locations $E_{\ell,\ell'}$

and corresponding labels $\epsilon_{\ell,\ell'}$ for any pair of labels ℓ, ℓ', satisfying $\Gamma(W) \sqsubseteq \epsilon_{\Gamma(W),\ell}$ and $\epsilon_{\ell,\Gamma(W)} \sqsubseteq \Gamma(W)$ for all ℓ and W (so copying from W to $E_{\Gamma(W),\ell}$ and from $E_{\ell,\Gamma(W)}$ to W is always allowed). Then we give a semantics to subprograms of the form $V := \mathsf{endorse}(W, \ell)$ by treating them as shorthand for the program $E_{\Gamma(W),\ell} := W; V := E_{\Gamma(W),\ell}$.

Example 5.3.14. Let p be the program $V := \mathsf{endorse}(W, \Gamma(V)); V := W$, with $\Gamma(V) = \alpha$ and $\Gamma(W) = \beta$, and $\alpha \not\sqsubseteq \beta$. Then M^p has endogenous variables including $\{V_i, W_i, E_{\beta,\alpha,i} \mid i \in \{0, 1, 2, 3\}\}$, and nontrivial structural equations

- $E_{\beta,\alpha,1} = V_0$,
- $W_2 = E_{\beta,\alpha,1}$,
- $W_3 = V_2$.

This program violates causal intransitive noninterference with respect to $\sqsubseteq$, but only due to the causal relationship $(M^p, \vec{u}) \models W_2 \rightsquigarrow V_3$: the other causal relationship, $(M^p, \vec{u}) \models W_0 \rightsquigarrow V_2$, is made up of permitted flows as $(M^p, \vec{u}) \models (W_0 \overset{E_{\beta,\alpha,1}}{\rightsquigarrow} V_2)$ and $\beta \sqsubseteq \epsilon_{\beta,\alpha} \sqsubseteq \alpha$. □

5.3.5 A sketch of IP-security equivalence

IP-security was first considered by Haigh and Young [15]. IP stands for "intransitive purge", where the intransitivity refers to the relationship between labels. Instead of being framed in terms of operational semantics generating runs and security labels on memory locations, IP-security and other security policies of this type (going back to Goguen and Meseguer [14]) are typically expressed by taking program executions to be a sequence of *state transitions* or *events*, each

associated with a function acting on the set of possible states. Security labels are assigned to events, and each agent is endowed with a *purge* function takes a list of events and removes all those that they are not allowed to see (because the security label on the event is above the agents' permissions), and a *view* function that restricts a state to the part the agent can actually observe. Security is then defined as the property that, in the agent's view of the state, the outcome of the true sequence of events and the sequence of events obtained by purging all those events that the agent is not allowed to see are indistinguishable.

Definition 5.3.15. (Adapted from [39]) For a sequence of actions $\alpha \in A^*$, a relation $\rightarrowtail$ on Λ and a set of labels $X \subseteq \Lambda$, recursively define $\texttt{ipurge}_X(\epsilon) = \epsilon$ and

$$\texttt{ipurge}_X(\alpha a) = \begin{cases} \texttt{ipurge}_{X \cup \Gamma(a)}(\alpha) \cdot a & \text{if } \exists \ell \in X.\, \ell \rightarrowtail \Gamma(a), \\ \texttt{ipurge}_X(\alpha) & \text{otherwise.} \end{cases}$$

Then a system with possible runs (determined by a starting state and a sequence of actions) $\mathcal{R} \subseteq \Sigma \times A^*$ is *IP-secure* if for all initial segments of runs (s_0, α') (i.e., (s_0, α') such that α' is a prefix of some α such that $(s_0, \alpha) \in R$), and labels ℓ, we have

$$\texttt{view}_\ell(s_0 \cdot \texttt{ipurge}_{\{\ell\}}(\alpha)) = \texttt{view}_\ell(s_0 \cdot \alpha). \quad \square$$

In order to compare this notion to causal intransitive noninterference, we need to be able to apply the two notions to the same programs and attacker models. This requires some work, as the way that parts of the system are associated with different agents (the system designer, the public, or any attackers) in security policies of the type assumed by Haigh and Young differs from policies that assume labelings of memory as those used by Myers et al., and the details

are beyond the scope of this thesis. An embedding of a data-protection model into a similar event-based framework is presented in [13]. Here, we will sketch a way in which a similar embedding could be performed for an event-based framework of the form used in [39] to establish the equivalence we want.

Intuitively, we will want to break up every step of the program that touches memory locations into a "read" transition, any computations that are performed, and a "write" transition. We can then associate security labels with transitions by taking the "read" and "write" transitions to have the labels of the memory locations they operate on. The splitting into "input" and "output" labels $\lambda^{\leftarrow}$ and $\lambda^{\rightarrow}$ is necessary for technical reasons. Specifically, one issue with using the HY framework in this fashion is that it necessitates specifying an ordering of actions (by writing out the list α) but can not represent the circumstance that subsequent actions in this ordering were causally independent, as is the case when two different variables are read without an intervening write action. Consider a defective version of the secure program in the "declassification controller" example 5.3.10(a), $V := U; W := V + U$: in addition to using the controller-declassified version of the value that was originally in U, W reads U directly. Suppose that the transition representation of the second assignment is such that U is read first, then V, then the addition is performed, and finally the result is written to W. The agent associated with the label of W should not be able to observe the outcome of this read of U (it could for instance well be that a new secret value was written to U after the declassification controller decided to declassify the previous one!). However, with the given definition of $\texttt{ipurge}_{\{\Gamma(W)\}}$, if we used the label set Λ^0 and defined both read and write actions to have the label of the memory location they involve, this read action of U will not actually be purged, because it is followed by a read of V, and actions with label $\Gamma(U)$ that

are followed by an action with label $\Gamma(V)$ are not purged if $\Gamma(U) \rightarrowtail \Gamma(V)$. With our construction, this problem does not arise, because no two distinct "read" labels $\lambda^{\rightarrow} \neq \lambda'^{\rightarrow}$ are related by $\rightarrowtail$.

Conjecture: A program p with a security labelling Γ and relation $\sqsubseteq$ satisfies causal intransitive noninterference iff its transition representation satisfies IP-security.

To prove this conjecture, we would show that in the transition representation, the intransitive purge $\texttt{ipurge}_{\{\ell\}}(\alpha)$ of a sequence α of transitions ending in a write to a variable V_i with $\Gamma(V) = \ell$ consists exactly of those read and write operations that precede it and affect variables W such that if $W \rightsquigarrow V$ in any context, then (W, V) is made of permitted flows. Then, we can introduce a special value $\perp$ in the state of the transition representation, which every register is initialised to, and all computations that involve at least one $\perp$ are taken to evaluate to $\perp$ themselves. This way, if the "read" action of any memory location W that the value of V_i depends on is purged by the `ipurge` operator (as must be the case if (W, V) is not made of permitted flows), then the value of V in $\texttt{view}_\ell(s_0 \cdot \texttt{ipurge}_{\{\ell\}}(\alpha))$ must also become $\perp$, rather than its actual value in $\texttt{view}_\ell(s_0 \cdot \alpha)$, and IP-security is violated.

CHAPTER 6

CAUSAL GRAPHS

The causal model representing a program can be arbitrarily large, and checking whether a given security property is satisfied in it is computationally costly. With a naive computation, we arrive at the following lower bound:

Proposition 6.0.1. *Suppose M is a causal model in n variables, the size of whose ranges $|\mathcal{R}(V)|$ is bounded above by k. Then directly checking whether there exists a $\vec{u} \in \mathcal{S}$ such that $(M, \vec{u}) \models A \rightsquigarrow B$ for endogenous variables A and B of the model takes $\Omega(|\mathcal{S}| \cdot n \cdot k)$ operations.*

Proof. For each of $|\mathcal{S}|$ possible $\vec{u}$, if $(M, \vec{u}) \models B = b$, for each of k possible values $a \in \mathcal{R}(A)$, we need to check whether $(M, \vec{u}) \models [A \leftarrow a]\neg(B = b)$. This might entail having to reevaluate the structural equations of all n variables. □

This complexity is often intractable, as $|\mathcal{S}|$ is generally exponential in n (every variable can take any value). It is not hard to show that the problem of checking whether there exists a $\vec{u}$ such that $(M, \vec{u}) \models A \rightsquigarrow B$ is in fact NP-complete by reducing from SAT. Given a Boolean formula φ in variables $C_1, \ldots, C_n$, we can construct a model with two Boolean endogenous variables A and B and n Boolean exogenous variables $C_1, \ldots, C_n$, and set structural equations to be $A = 1$, $B = A \vee \neg\varphi(C_1, \ldots, C_n)$. Then $A \rightsquigarrow B$ iff some assignment $\vec{u}$ to exogenous variables satisfies φ (so $(M, \vec{u}) \models [A \leftarrow 0]B = 0$. Many related problems in model-checking of causal formulae are known to be similarly complex; see [17], section 5.3, for a discussion.

We would therefore like to investigate an alternative means of proving pro-

grams secure which has better complexity characteristics. Due to the NP-completeness of the causality checking problem, it is inevitable that some information is lost in the process; the algorithm we present, therefore, will be sound (in that it will never declare a program that does not satisfy a security property in question secure), but not complete (in that some secure programs will not be declared secure).

6.1 Definition

6.1.1 Basic definition

Causal graphs, or causal networks, are an object frequently used in the study of causality to qualitatively represent causal models [17]. We want to use causal graphs to analyse the causal models we generate to represent programs, and compute guarantees of security, more efficiently.

The idea behind the causal graph is to represent the variables of a causal model as vertices in a graph, and introduce a directed edge from some variable A to a variable B if A is an *immediate* dependency of B. Formally, we can give a basic definition as follows.

Definition 6.1.1. Let M be a causal model with exogenous variables $\mathcal{U}$ and endogenous variables $\mathcal{V}$. The *causal graph* $G(M)$ is the graph with vertices $\mathcal{U} \cup \mathcal{V}$, and a directed edge from W to V whenever the structural equation for V can change value in response to a change to the value of W: there exist assignments $\vec{u}$, $\vec{u}'$ to the variables not including V such that $\vec{u}$ and $\vec{u}'$ only differ in the W-entry and $F_V(\vec{u}) \neq F_V(\vec{u}')$.

Causal graphs are relevant for the satisfaction of security properties because of the following basic fact, of which we will in fact go on to prove a more powerful version later.

Theorem 6.1.2. *If* $(M, \vec{u}) \models U \rightsquigarrow V$ *for any* $\vec{u}$*, then there is a directed path from the vertex* U *to the vertex* V *in* $G(M)$*.*

In the contrapositive, this means that when there is *no* directed path, then there is no $\vec{u}$ such that $(M, \vec{u}) \not\models U \rightsquigarrow V$. By leveraging this theorem, we can establish that programs satisfy any security property of the form "there exist no U, V such that **Bad**(U, V) (where **Bad** is some property of pairs of variables) and $(M, \vec{u}) \models U \rightsquigarrow V$" by just verifying absence of paths for bad pairs in the causal graphs.

6.1.2 Slicing

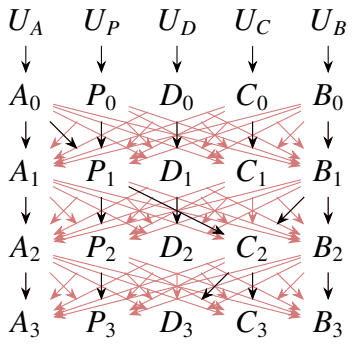
We can consider how our definition plays out for a simple program. We will take the small-step semantics to be such that evaluating the condition of an if statement takes one step and reduces the program to the appropriate branch, so for instance

$$\langle \mathsf{if\ true\ then}\ p_1\ \mathsf{else}\ p_2\ \mathsf{end}, \sigma\rangle \to \langle p_1, \sigma\rangle.$$

Example 6.1.3. Consider the program

$$\mathsf{if}\ A = 1\ \mathsf{then}\ C := B\ \mathsf{else}\ C := 0\ \mathsf{end}; D := C.$$

The causal graph $G(M^p)$ for this program is:



What happened here? We observe (pale red) directed edges connecting basically all variables to all variables at the respective next step, rendering the causal graph plainly uninformative. This is actually meaningful, and an inevitable consequence of the semantics, which determines the value of all variables at time i in terms of the program and the contents of memory at time $i-1$, by computing the next step. If the program at, say, step 2 somehow changed (perhaps by attacker action, or an intended behaviour that completely overwrote it), then the new program could have any effect on any location of memory at step 3. Replacing the semantics which stores the entire remaining program in a variable with one that merely keeps track of a "program counter" would not remedy this problem either: a sufficiently large program could reasonably be expected to contain statements affecting any given variable *somewhere*, and so an attacker who can change the program counter at time i could affect any variable at time $i+1$ by just picking a program counter pointing to a location at which that particular variable is written to.

However, the existence of these edges poses a problem for our agenda of deriving a meaningful sufficient criterion for security by analysing causal graphs. We hope to make use of the statement that absence of a path from U to V in the causal graph means that U cannot be a cause of V; conversely, if a path does

exist, that means we cannot rule out that U could in fact be a cause of V. But in most programs, our present approach would give rise to the latter case far too often. Whenever a program contains conditionals, the remaining program (or the program counter, should that implementation be chosen) after the conditional depends on whatever the truth value of the conditioning expression depends on. In our example, this is reflected by the presence of an edge from A_0 to P_1, as whether the branch is taken or not depends on the value of A. As a result, we obtain causal paths from A_0 to V_i for *any* V and $i \geq 2$!

Not only does this likely preclude us from giving any security guarantees, it is also not meaningful in most cases: after all, we do not often consider attackers that can change the program in arbitrary ways at any point in time; and indeed, such an attacker would be basically omnipotent for information flow purposes (so no meaningful security could be attained). We would therefore do well to create a definition of the causal graph that is restricted to having only edges that actually represent *relevant* dependencies, that is, ones that actually could come into play under the set of contexts and attacker interventions that we consider. This is captured by the following definition.

Definition 6.1.4. Let M be a causal model with exogenous variables $\mathcal{U}$ and endogenous variables $\mathcal{V}$. Given a set $\mathcal{S}$ of assignments to exogenous variables and $\mathcal{I}$ of interventions for M, the $(\mathcal{S}, \mathcal{I})$-*slice causal graph* $G_{(\mathcal{S},\mathcal{I})}(M)$ is the graph with vertices $\mathcal{U} \cup \mathcal{V}$ and a directed edge $W \to V$ whenever

SC1. there exist assignments $\vec{v}$, $\vec{v}'$ to the variables not including V such that $\vec{v}$ and $\vec{v}'$ only differ in the W-entry, say $\vec{v}(W) = w$, $\vec{v}'(W) = w'$ and $F_V(\vec{v}) \neq F_V(\vec{v}')$ (as before)

SC2. there exist assignments to exogenous variables $\vec{u}, \vec{u}' \in \mathcal{S}$ and interven-

tions $\vec{Z} \leftarrow \vec{z}, \vec{Z}' \leftarrow \vec{z}' \in \mathcal{I}$ such that $(M, \vec{u}) \models [\vec{Z} \leftarrow \vec{z}]W = w$ and $(M, \vec{u}') \models [\vec{Z}' \leftarrow \vec{z}']W = w'$ (the difference in W can be achieved using the given contexts and interventions).

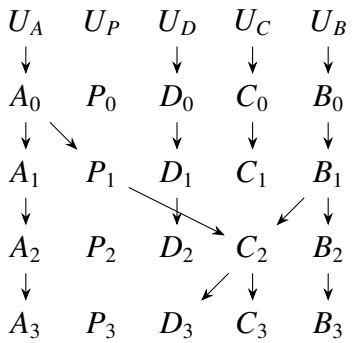
Armed with this new definition, we can now try to revisit the previous example with a slice representing more realistic assumptions. In particular, we will assume an attacker can not actually change the program, though the attacker may change anything else:

Definition 6.1.5. Given a program p, let $\mathcal{S}_p$ denote the set of all assignments to exogenous variables of M^p that set U_P to p, and $\mathcal{I}_V$ the set of all interventions upon variables excluding the program variables P_i.

Example 6.1.6. Once again, consider the program p:

$$\mathsf{if}\ A = 1\ \mathsf{then}\ C := B\ \mathsf{else}\ C := 0\ \mathsf{end};\ D := C.$$

Then the $(\mathcal{S}_p, \mathcal{I}_V)$-slice causal graph of M^p is



Note in particular that not only have we lost the objectionable directed edges linking all variables at every step, but not even P_{i+1} seems to depend on P_i anymore! This is because as we assume that the program is fixed and can't be intervened upon, the remaining program at every step has become essentially a

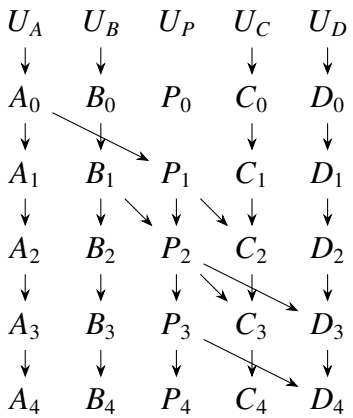
constant. Only the program P_1 at step 1 can vary at all, depending on the value of A: if A_0 was 1, then we take the first branch and so P_1 is $C := B; D := C$, and otherwise we take the second branch and P_1 is $C := 0; D := C$. Since both forks of the if take the same number of steps, one step later the program P_2 becomes $D := C$ regardless of the side taken.

It may be worth looking at some more involved examples, with nested if statements.

Example 6.1.7. Consider the program

$$\begin{aligned}&\text{if } A = 1 \text{ then}\\ &\quad \text{if } B = 1 \text{ then } C := 1 \text{ else } C := 0 \text{ end}\\ &\text{else } C := 2 \text{ end}; D := 1.\end{aligned}$$

This program has the following $(\mathcal{S}_p, \mathcal{I}_V)$-slice causal graph:



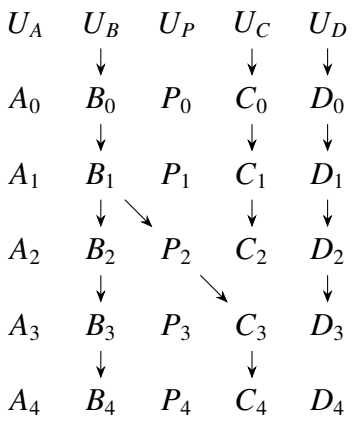
In this program, the timing depends on whether A_0 is 1, as the first branch of the if takes 2 steps whereas the second one only takes one step to evaluate. It is therefore not surprising that we see more paths (after all, whether we execute the final $D := 1$ after the second or after the third timestep depends on A).

However, a spurious path is also created: since P_2 may depend on B_1 (if $A = 1$ and $B = 1$, then it is $C := 1; D := 1$, otherwise it is $C := 0; D := 1$), we have a directed edge between the two, and subsequent P_i also depend on their predecessor programs due the timing information. However, there is not actually a causal relationship between D_3 and B_1: whether $D := 1$ is executed at the second or the third step has nothing to do with B, and if the conditional on B becomes relevant at all, $D := 1$ will be executed at the third step regardless of the value of B.

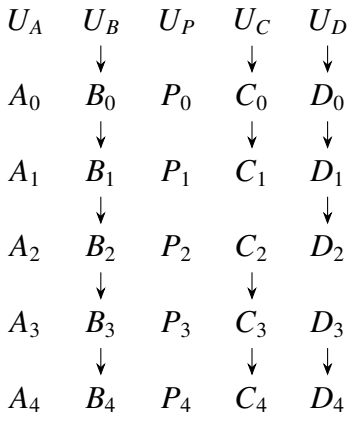
This example illustrates a general class of issues that often arise when forming causal graphs, and may frustrate the attempt to understand the underlying causal model by looking at the graph. In general terms, what happened here is that a single variable, P_2, encodes what are really two independent aspects of the scenario: whether the outer if's first branch was taken, and whether the inner if's first branch was taken. Some dependencies of P_2 (here, P_1) only affect the first aspect, whereas others (here, B_1) only affect the second; and on the other hand, P_3's dependency on P_2 is in reality only a dependency on the former aspect. However, the causal graph does not encode the circumstance that P_3 does not in fact indirectly depend on B_1 via P_2, as its dependency ignores the encoding of the aspect of P_2 that depended on B_1. One way we can recover sufficient information about the causal model is by refining our slice causal graph by taking separate, complementary slices:

Example 6.1.8. Let $\mathcal{S}_0$ and $\mathcal{I}_0$ be the subsets of $\mathcal{S}$ and $\mathcal{I}$ respectively that set A to 0, and $\mathcal{S}_1$ and $\mathcal{I}_1$ be likewise for A set to 1.

Then $G_{(\mathcal{S}_0,\mathcal{I}_0)}(M^p)$ is



and $G_{(\mathcal{S}_1,\mathcal{I}_1)}(M^p)$ is



6.2 Relating paths to causality

We are now ready to prove the "real" version of Theorem 6.1.2, which applies to all slice causal graphs, as long as the causal relationships we are interested in are actually included in the relevant slice. The original statement of Theorem 6.1.2 is an immediate corollary if we apply this version to the slice for all assignments and interventions that are possible with the given ranges for each variable.

Theorem 6.2.1. *If* $(M, \vec{u}) \models U \rightsquigarrow_{\mathcal{I}} V$ *for some* $\vec{u} \in \mathcal{S}$*, then there is a directed path from the vertex* U *to the vertex* V *in* $G_{(\mathcal{S},\mathcal{I})}(M)$*.*

Proof. Let $\geq$ be a linear extension of the "depends on" order of the variables of M, that is, a topological sort of the causal graph, which must exist since M is acyclic. Then $\leq$ must be well-founded, since every variable can depend only on variables with a strictly lower timestamp and so we can pick a variable with minimal timestamp in any set to get a minimal element.

Say that a variable W *changes under* $U \leftarrow u'$ if $(M, \vec{u}) \models W = w^*$ and $(M_{U \leftarrow u'}, \vec{u}) \models W = w'$ for some $w' \neq w^*$. If $(M, \vec{u}) \models U \rightsquigarrow V$, there must be a u' such that V changes under $U \leftarrow u'$, by definition. Set $V_0 = V$. We build up a $\leq$-descending sequence of variables $V_0 \geq V_1 \geq V_2 \geq \ldots$ that change under $U \leftarrow u'$ such that there is a directed edge from V_{i+1} to V_i for all i, that is, $V_0 \leftarrow V_1 \leftarrow V_2 \leftarrow \ldots$ is a directed path in the causal graph for M. We claim that this sequence is finite and ends in U. Indeed, the value of V_i is determined by its structural equations from the values of the variables it depends on. Either (1) $V_i = U$, or (2) the immediate dependencies in both M and $M_{U \leftarrow u'}$ are a superset of the vertices that have a directed edge to V_i in the causal graph (for either of M and $M_{U \leftarrow u'}$).

In case (2), consider the immediate dependencies of V_i in M. Not all of these variables can take the same value when $U = u$ and under $U \leftarrow u'$, or V_i, which is completely determined by its immediate dependencies, would not actually change under $U \leftarrow u'$. So at least one immediate dependency W must have a directed edge to V_i: the value of u' must change at some point if we change the values of the immediate dependencies from the ones they take under $U = u$ to the ones they take under $U \leftarrow u'$ one by one (so there exists a W for which

$W \to V_i$ satisfies SC1), and both the old and the new value are attainable as $U \leftarrow u' \in \mathcal{I}$ (so SC2 is satisfied for the same edge). Then, by definition, $V_i \geq W$, and we can extend the path by setting $V_{i+1} := W$. Since $\leq$ is well-founded, this sequence must terminate with some final variable V_k, beyond which it could not be extended. This variable can't have a constant structural equation in M or be exogenous, since then it wouldn't change under $U \leftarrow u'$. Therefore, V_i must have been an instance of case (1), so it must be U itself, as claimed. □

Hence, we can check that $(M, \vec{u}) \models \neg(U \rightsquigarrow V)$ by establishing that there is no directed path from U to V. Often, it is most efficient to compute all shortest paths using an algorithm such as Floyd-Warshall and then establish that the entries for all pairs of vertices between which no causal relationship should hold do not have a finite distance. To get an idea of what kinds of programs can be accurately proved secure using slice causal graphs, and in what settings the loss of information due to the abstraction results in this no longer being possible, we consider a few examples.

Example 6.2.2. (i) Consider the program

$$\begin{aligned}&\mathsf{if}\ A = 1\ \mathsf{then}\\ &\quad \mathsf{if}\ B = 1\ \mathsf{then}\ C := 1\ \mathsf{else}\ C := 0\ \mathsf{end}\\ &\mathsf{else}\ C := 2\ \mathsf{end};\ D := 1.\end{aligned}$$

of Example 6.1.7. Suppose the labelling is $\Gamma(A) = \Gamma(B) = \Gamma(C) = \text{secret}$ and $\Gamma(D) = \text{public}$, with $\text{secret} \not\sqsubseteq \text{public}$. As we saw in the previous example, the graph $G_{\mathcal{S}_p, \mathcal{I}_V}(M^p)$ has a directed path $A_0 \to P_1 \to P_2 \to D_3$, and hence we can not rule out that there may exist a $\vec{u} \in \mathcal{S}_p$ s.t. $(M^p, \mathcal{S}_p) \models A_0 \rightsquigarrow_{\mathcal{I}_0} D_3$. Indeed, the program does in fact not satisfy confidentiality with respect to

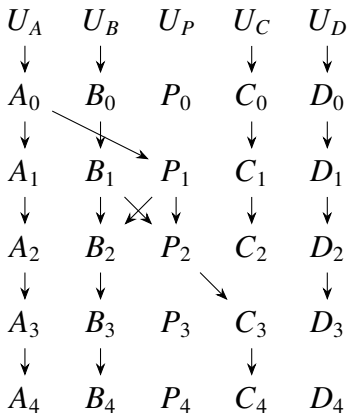
this ordering, and a causal relationship of this form holds: depending on whether $A = 1$, the assignment $D := 1$ either gets executed after two or three timesteps, and so checking whether the public D_3 is 1 allows us to infer whether the secret A_0 was 1.

(ii) Consider the same program and ordering of labels, but take the labelling to instead be $\Gamma(B) = \Gamma(C) = \text{secret}$ and $\Gamma(A) = \Gamma(D) = \text{public}$. This program does in fact satisfy confidentiality with respect to the labelling: the public memory location A never gets modified, and when and how the public memory location D gets modified only depends on A, which is also public, so no secrets leak. However, in the slice causal graph $G_{\mathcal{S}_p, \mathcal{I}_V}(M^p)$, we have a directed path $B_0 \to B_1 \to P_2 \to D_3$. Therefore, we can't rule out that $(M^p, \mathcal{S}_p) \vDash B_0 \rightsquigarrow_{\mathcal{I}_0} D_3$, that is, the secret value of B leaking. This is due to the phenomenon we discussed earlier, where the separation between the impact of A and B upon control flow is lost in the causal graph. Therefore, even though this program is secure in the sense of the property we care about, we cannot prove this via the causal graph.

(iii) Consider the program

$$\begin{aligned}&\mathsf{if}\ A = 1\ \mathsf{then}\\ &\quad \mathsf{if}\ B = 1\ \mathsf{then}\ C := 1\ \mathsf{else}\ C := 0\ \mathsf{end}\\ &\mathsf{else}(B := 0; C := 2)\ \mathsf{end}; D := 1,\end{aligned}$$

differing from the previous one in that both branches of the outer if now take an equal number of steps. This program has the following slice causal graph $G_{\mathcal{S}_p, \mathcal{I}_V}(M^p)$:



Take the labelling to be $\Gamma(A) = \Gamma(B) = \Gamma(C) =$ secret and $\Gamma(D) =$ public as in (i). Since there is no directed path from any A_i, B_i or C_i to any D_i, we can conclude by Theorem 6.2.1 that no secret cause can have a public effect, and so the program satisfies confidentiality. Indeed, it is easy to convince ourselves that D will receive a constant value in the third step regardless of the values of secret variables, and so no secrets leak through it.

Besides this application to but-for causality between individual variables, paths in causal graphs allow us to efficiently inspect the ways in which causal relationships between variables are mediated.

Theorem 6.2.3. *Suppose that* $(M, \vec{u}) \models U \rightsquigarrow_{\mathcal{I}} V$ *for some recursive model* M *and* $\vec{u} \in \mathcal{S}$*, and all directed paths from* U *to* V *in* $G_{(\mathcal{S},\mathcal{I})}(M)$ *contain a vertex from* S*. Then* $(M, \vec{u}) \models (U \overset{S}{\rightsquigarrow} V)$*.*

Proof. Repeat the proof of Theorem 6.2.1 with the additional assumption that the structural equations of all variables in S have been set to their actual values in $(M, \vec{u})$. Since, by assumption, if we walk backwards on directed edges, we will reach a variable in S *before* reaching U, all descending sequences of dependencies must actually terminate at a variable that has a constant structural

equation or is exogenous. Hence, the value of V can't change in response to U being intervened upon. Therefore, having intervened to set $S \leftarrow s^*$, we have $\neg(U \rightsquigarrow V)$, and hence can conclude $(U \overset{S}{\rightsquigarrow} V)$. □

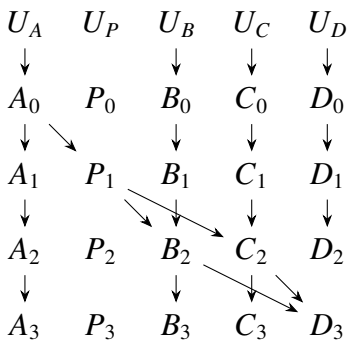
Corollary 6.2.4. *If every pair of vertices $(W, W') \in \mathcal{M}_p^2$ of $G_{S,I}(M^p)$ connected by a directed path from W to W' either satisfies $\Gamma(W) \sqsubseteq \Gamma(W')$ or passes through another vertex $\in \mathcal{M}_p^2$, then whenever $(M^p, \vec{u}) \vDash U_i \rightsquigarrow V_j$, (U_i, V_j) is made of permitted flows in the sense of Definition 5.3.12.*

As before, we can present an example where this allows us to correctly identify a program as secure, and another one where loss of information due to abstraction leaves us unable to establish that a secure program actually is secure.

Example 6.2.5. (i) Consider the program

$$\text{if } A = 1 \text{ then } B := 1 \text{ else } C := 2 \text{ end}; D := B + C$$

evaluated against an intransitive security policy as in Definition 5.3.12, where $\Gamma(A) \sqsubseteq \Gamma(B), \Gamma(C)$, $\Gamma(B), \Gamma(C) \sqsubseteq \Gamma(D)$ and no other relations hold. This program has the following slice causal graph $G_{S_p,I_V}(M^p)$:



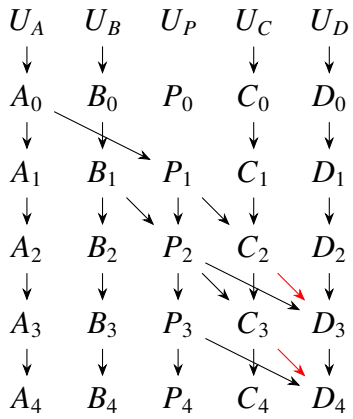
Since all directed paths from A_0 to D_3 traverse either B_2 or C_2, we conclude that if $(M^p, \vec{u}) \vDash A_0 \rightsquigarrow D_3$, then $(M^p, \vec{u}) \vDash (A_0 \overset{B_2,C_2}{\rightsquigarrow},)D_3$. Therefore, the

pair (A_0, D_3) is made up of permitted flows. By checking off all the other instances of $V \rightsquigarrow W$, we can conclude that causal intransitive noninterference must indeed be satisfied.

(ii) Consider the following slight modification (marked in red) of the program from Example 6.1.7.

$$\begin{array}{l}\mathsf{if}\ A = 1\ \mathsf{then}\\ \quad \mathsf{if}\ B = 1\ \mathsf{then}\ C := 1\ \mathsf{else}\ C := 0\ \mathsf{end}\\ \mathsf{else}\ C := 2\ \mathsf{end};\ D := C.\end{array}$$

This program has the following $(\mathcal{S}_p, \mathcal{I}_V)$-slice causal graph:



Suppose that our intransitive noninterference policy is such that all flows except for those from B to D are allowed: $\Gamma(B) \not\sqsubseteq \Gamma(D)$. Then this program is in fact secure with respect to the policy. However, the directed path $B_1 \to P_2 \to P_3 \to D_4$ suggests that it may not be: we can't rule out that the information from B did not only get conveyed to D via C (as witnessed by the path $B_1 \to P_2 \to C_3 \to D_4$), but also directly via the timing of the final assignment. (In reality, this can not happen: whenever B is read at all, D is only assigned to at the last step.)

CHAPTER 7

CONCLUSIONS

We have introduced a way of representing computer programs as causal models in the Halpern-Pearl framework and shown that several security properties, including noninterference, robust declassification, endorsement and intransitive properties such as a data-protection version of IP-security, can be expressed as causal formulae. Beyond these formally specified properties, we demonstrated that many natural security-relevant scenarios also admit intuitive causal formulations, and the formal properties that we discussed can be evaluated in the context of natural scenarios. In our view, the causal formulae that we have presented for these properties are quite intuitive, and this fact alone is a point in favour of the causal approach. The connection between causality and security also enabled new insights into causality, as the scenarios we considered necessitated extensions of the HP framework to make allowance for nesting. We suspect that many other security properties admit a natural formulation in terms of causality: for instance, the notion of separation of duties [3] suggests that certain sensitive effects need to be caused by multiple independent principals, with the objective of preventing fraud and rogue actions from single compromised components or agents.

We have introduced a refinement of the standard notion of causal graphs, the slice causal graph, as a tool to efficiently compute sound guarantees of program security. In future work, we hope to develop this aspect of the theory further. We are currently investigating a type system that can directly compute a supergraph of the slice causal graph for a program, obviating the need to consider the causal model at all to achieve the sound guarantees and tying our approach

into the tradition of language-based security. Several extensions of the causal graph also appear to be of interest; on one hand, we can improve computational efficiency further by forming graph quotients to collapse redundant parts of the causal graph, and on the other we are interested in hypergraph-based formalisms that would better capture the higher-order structure of a causal model, with a look to being able to obtain similar sound guarantees for nested properties such as robust declassification.

Part II

Information Acquisition Under Resource Limitations in a Noisy Environment

CHAPTER 8

INTRODUCTION

Decision-making is typically subject to resource constraints. However, an agent may be able to choose how to allocate his resources. We consider a simple decision-theoretic framework in which to examine this resource-allocation problem. Our framework is motivated by a variety of decision problems in which multiple noisy signals are available for sampling, such as the following:

- An animal must decide whether some food is safe to eat. We assume that "safe" is characterised by a Boolean formula φ, which involves variables that describe (among other things) the presence of unusual smells or signs of other animals consuming the same food. The animal can perform a limited number of tests of the variables in φ, but these tests are noisy; if a test says that a variable v is true, that does not mean that v is true, but only that it is true with some probability. After the agent has exhausted his test budget, he must either guess the truth value of φ or choose not to guess. Depending on his choice, he gets a payoff. In this example, guessing that φ is true amounts to guessing that the food is safe to eat. There will be a small positive payoff for guessing "true" if the food is indeed safe, but a large negative payoff for guessing "true" if the food is not safe to eat. In this example we can assume a payoff of 0 if the agent guesses "false" or does not guess, since both choices amount to not eating the food.
- A quality assurance team needs to certify a modular product, say a USB memory stick, or send it back to the factory. Some subsystems, such as the EEPROM cells, are redundant to an extent, and a limited number of them not working is expected and does not stop the product from functioning.

Others, such as the USB controller chip, are unique; the device will not work if they are broken. Whether the device is good can be expressed as a Boolean combination of variables that describe the goodness of its components. Time and financial considerations allow only a limited number of tests to be performed, and tests themselves have a probability of false negatives and positives. What parts should be tested and how often?

- A data scientist wants to perform a complex query on a very big database. A certain error rate is acceptable; in any case, executing the query exactly is infeasible with the available hardware. The selection criterion itself is a Boolean combination of some atomic predicates on the entries of the database, which can be evaluated only using heuristics (which are essentially probabilistic algorithms). Given a query that asks for rows that, for instance, satisfy the criterion $P_1 \wedge (P_2 \vee P_3)$ in three predicates P_i, which heuristics should be run and how often should they be run to attain the desired error rate?

We are interested in optimal strategies for each of these problems; that is, what tests should the agent perform and in what order. Unfortunately (and perhaps not surprisingly), as we show, finding an optimal strategy (i.e., one that obtains the highest expected payoff) is infeasibly hard. We provide a heuristic that guarantees a positive expected payoff whenever the optimal strategy gets a positive expected payoff. Our analysis of this strategy also gives us the tools to examine two other problems of interest.

The first is *rational inattention*, the notion that in the face of limited resources it is sometimes rational to ignore certain sources of information completely. There has been a great deal of interest recently in this topic in economics [37, 43].

Here we show that optimal testing strategies in our framework exhibit what can reasonably be called rational inattention (which we typically denote RI from now on). Specifically, our experiments show that for a substantial fraction of formulae, an optimal strategy will hardly ever test variables that are clearly relevant to the outcome. (Roughly speaking, "hardly ever" means that as the total number of tests goes to infinity, the fraction of tests devoted to these relevant variables goes to 0.) For example, consider the formula $v_1 \vee v_2$. Suppose that the tests for v_1 and v_2 are equally noisy, so there is no reason to prefer one to the other for the first test. But for certain choices of payoffs, we show that if we start by testing v_2, then all subsequent tests should also test v_2 as long as v_2 is observed to be true (and similarly for v_1). Thus, with positive probability, the optimal strategy either ignores v_1 or ignores v_2. Our formal analysis allows us to conclude that this is a widespread phenomenon.

The second problem we consider is what makes a concept (which we can think of as being characterised by a formula) hard. To address this, we use our framework to define a notion of hardness. Our notion is based on the minimum number of tests required to have a chance of making a reasonable guess regarding whether the formula is true. We show that, according to this definition, XORs (i.e., formulae of the form $v_1 \oplus \cdots \oplus v_n$, which are true exactly if an odd number of the v_i's are true) and their negations are the hardest formulae. We compare this notion to other notions of hardness of concepts considered in the cognitive psychology literature (e.g., [11, 27, 36]).

Organisation. The rest of this part is organized as follows. In Chapter 9, we formally define the games that we use to model our decision problem and analyse the optimal strategies for a simple example. The detailed calculations for

this example can be found in Section 9.3. In Section 9.2, we look at the problem of determining optimal strategies more generally. We discuss the difficulty of this problem and analyse a simple heuristic, developing our understanding of the connection between payoffs and certainty in the process. In Chapter 10, we formally define rational inattention and discuss the intuition behind our definition. After considering some examples of when RI occurs under our definition, we show that there is a close connection between rational inattention and particular sequences of observations (*optimal test outcome sequences*) that may occur while testing. We use this connection to obtain a quantitative estimate of how common RI is in formulae involving up to 10 variables. The theory behind this estimate is presented in Section 10.2.2, where we relate the optimal test outcome sequences to the solution polytope of a particular linear program (LP). While we are not aware of any explicit connections, our method should be seen in a broader tradition of applying LPs to decision problems such as multi-armed bandits [5], and may be of independent interest for the analysis of information acquisition. Finally, in Chapter 11, we introduce our notion of test complexity, prove that XORs are the formulas of greatest test complexity (the details of the proof are in Section 11.1), and discuss the connections to various other notions of formula complexity in the cognitive and computational science literature.

CHAPTER 9

INFORMATION-ACQUISITION GAMES

9.1 Definition

We model the *information-acquisition game* as a single-player game against nature, that is, one in which actions that are not taken by the player are chosen at random. The game is characterised by five parameters:

- a Boolean formula φ over variables $v_1, \ldots, v_n$ for some $n > 0$;
- a probability distribution D on truth assignments to $\{v_1, \ldots, v_n\}$;
- a bound k on the number of tests;
- an *accuracy vector* $\vec{\alpha} = (\alpha_1, \ldots, \alpha_n)$, with $0 \leq \alpha_i \leq 1/2$ (explained below);
- payoffs (g, b), where $g > 0 > b$ (also explained below).

We denote this game as $G(\varphi, D, k, \vec{\alpha}, g, b)$.

In the game $G(\varphi, D, k, \vec{\alpha}, g, b)$, nature first chooses a truth assignment to the variables $v_1, \ldots, v_n$ according to distribution D. While the parameters of the game are known to the agent, the assignment chosen by nature is not. For the next k rounds, the agent then chooses one of the n variables to test (possibly as a function of history), and nature responds with either T or F. The agent then must either guess the truth value of φ or choose not to guess.

We view a truth assignment A as a function from variables to truth values ($\{T, F\}$); we can also view a formula as a function from truth assignments to truth values. If the agent chooses to test v_i, then nature returns $A(v_i)$ (the right answer)

with probability $1/2 + \alpha_i$ (and thus returns $\neg A(v_i)$ with probability $1/2 - \alpha_i$).[1] Thus, outcomes are independent, conditional on a truth assignment. Finally, if the agent choses not to guess at the end of the game, his payoff is 0. If he chooses to guess, then his payoff is g (good) if his guess coincides with the actual truth value of φ on assignment A (i.e., his guess is correct) and b (bad) if his guess is wrong. It is occasionally useful to think of a formula φ as a function from assignments to truth values; we thus occasionally write $\varphi(A)$ to denote the truth value of φ under truth assignment A. A strategy for an agent in this game can be seen as a pair of functions: one that determines which test the agent performs after observing a given sequence of test outcomes of length $< k$, and one that determines the whether to make a guess and, if so, which guess to make, given all k test outcomes.

Example 9.1.1. Consider the information-acquisition game over the formula $v_1 \vee v_2$, with $k = 2$ tests, a uniform distribution on truth assignments, accuracy vector $(1/4, 1/4)$, correct-guess reward $g = 1$ and wrong-guess penalty $b = -16$. As we show (see Section 9.3) this game has two optimal strategies:

1. test v_1 twice, guess T if both tests came out T, and make no guess otherwise;
2. test v_2 twice, guess T if both tests came out T, and make no guess otherwise. □

Thus, in this game, an optimal strategy either ignores v_1 or ignores v_2. As we show in Section 9.3, the strategy "test v_1 and then v_2, then guess T if both tests

[1] Note that this means that the probability of a false positive and that of a false negative are the same. While we could easily extend the framework so as to allow the accuracy in a test on a variable v to depend on whether $A(v)$ is T or F, doing so would complicate notation and distract from the main points that we want to make.

came out T'' is strictly worse than these two; in fact, its expected payoff is negative! This was (to us, at least) surprising: among other things, it implies that optimal strategies do not form a convex set, so a convex combination of two optimal testing strategies is not necessarily optimal.

If we increase k, the situation becomes more nuanced. For instance, if $k = 4$, an optimal strategy tests v_1 once, and if the test comes out F, tests v_2 three times and guesses T if all three tests came out T. However, it always remains optimal to keep testing one variable as long as the tests keep coming out true. That is, all optimal strategies exhibit RI in the sense that there are test outcomes that result in either v_1 never being tested or v_2 never being tested, despite their obvious relevance to $v_1 \vee v_2$.

For our results, we need to analyze the probability of various events related to the game. Many of the probabilities that we care about depend on only a few parameters of the game. Formally, we put a probability on *histories* of an information-acquisition game. A history is a tuple of the form (A, S, a), where A is the assignment of truth values to the n variables chosen by nature, $S = (v_{i_1} \approx b_1, \ldots, v_{i_k} \approx b_k)$ is a *test-outcome sequence* in which $v_{i_j} \approx b_j$ indicates that the jth test was performed on variable v_{i_j} and that nature responded with the test outcome b_j, and a is the final agent action of either making no guess or guessing some truth value for the formula. A game $G(\varphi, D, k, \vec{\alpha}, g, b)$ and agent strategy σ for this game then induce a probability $\Pr_{G,\sigma}$ on this sample space.

Example 9.1.2. In Example 9.1.1, $\Pr_{G,\sigma}(\varphi)$ is 3/4, as we know only that there is a probability of 3/4 that nature picked a satisfying assignment. After observing a single test outcome suggesting that v_1 is false, the posterior probability $\Pr_{G,\sigma}(\varphi \mid (v_1 \approx F))$ drops to 5/8. If the same test is performed and the outcome is again F,

the posterior drops further to $\Pr_{G,\sigma}(\varphi \mid (v_1 \approx F, v_1 \approx F)) = 11/20$. □

The only features of the game G that affect the probability are the prior distribution D and the accuracy vector α, so we write $\Pr_{D,\alpha,\sigma}(\varphi)$ rather than $\Pr_{G,\sigma}(\varphi)$. If some component of the subscript does not affect the probability, then we typically omit it. In particular, we will show (Lemma 10.2.6) that the strategy σ does not affect $\Pr_{G,\sigma}(\varphi \mid S)$, so we write $\Pr_{D,\vec{\alpha}}(\varphi \mid S)$. Finally, the utility (payoff) received by the agent at the end of the game is a real-valued random variable that depends on parameters b and g. We can define the expected utility $\mathbb{E}_{G,\sigma}(\text{payoff})$ as the expectation of this random variable.

9.2 Determining optimal strategies

It is straightforward to see that the game tree[2] for the game $G(\varphi, D, k, \vec{\alpha}, g, b)$ has $3(2^n)(2n)^k$ leaves: there is a branching factor of 2^n at the root (since there are 2^n truth assignments) followed by k branching factors of n (for the n variables that the agent can choose to test) and 2 (for the two possible outcomes of a test). At the end there are three choices (don't guess, guess T, and guess F). A straightforward backward induction can then be used to compute the optimal strategy. Unfortunately, the complexity of this approach is polynomial in the number of leaves, and hence grows exponentially in k even for a fixed number of variables n, quickly becoming infeasible.

In general, it is unlikely that the dependency on 2^n can be removed. In the special case that $b = -\infty$ and $\alpha_i = \frac{1}{2}$ for all i (so tests are perfectly accurate, but the

[2]For the one-player games that we are considering, a game tree is a graph whose nodes consist of all valid partial sequences of actions in the game, including the empty sequence, and two nodes have an edge between them if they differ by appending one action.

truth value of the formula must be established for sure), determining whether there is a strategy that gets a positive expected payoff when the bound on tests is k reduces to the problem of finding a conjunction of length k that implies a given Boolean formula. Umans ([38]) showed that this problem is Σ_2^p-complete, so it lies in a complexity class that is at least as hard as both NP and co-NP.

A simple heuristic (whose choice of variables is independent of φ) would be to simply test each variable in the formula k/n times, and then choose the action that maximises the expected payoff given the observed test outcomes. We can calculate in time polynomial in k and n the expected payoff of a guess, conditional on a sequence of test outcomes. Since determining the best guess involves checking the likelihood of each of the 2^n truth assignments conditional on the outcomes, this approach takes time polynomial in k and 2^n. We are most interested in formulae where n is small (note that k still can be large, since we can test a variable multiple times!), so this time complexity would be acceptable. However, this approach can be arbitrarily worse than the optimum. As we observed when discussing Example 9.1.1, the expected payoff of this strategy is negative, while there is a strategy that has positive expected payoff.

An arguably somewhat better heuristic, which we call the *random-test heuristic*, is to choose, at every step, the next variable to test uniformly at random, and again, after k observations, choosing the action that maximises the expected payoff. This heuristic clearly has the same time complexity as the preceding one, while working better in information-acquisition games that require an unbalanced approach to testing.

Proposition 9.2.1. *If there exists a strategy that has positive expected payoff in the information-acquisition game G, then the random-test heuristic has positive expected*

payoff.

To prove Proposition 9.2.1, we need a preliminary lemma. Intuitively, an optimal strategy should try to generate test-outcome sequences S that maximise $|\Pr_{D,\vec{\alpha}}(\varphi \mid S) - 1/2|$, since the larger $|\Pr_{D,\vec{\alpha}}(\varphi \mid S) - 1/2|$ is, the more certain the agent is regarding whether φ is true or false. The following lemma characterises how large $|\Pr_{D,\vec{\alpha}}(\varphi \mid S) - 1/2|$ has to be to get a positive expected payoff.

Definition 9.2.2. The *threshold* associated with payoffs b, g is $q(b, g) = \frac{b+g}{2(b-g)}$. □

Lemma 9.2.3. *The expected payoff of $G(\varphi, D, k, \vec{\alpha}, g, b)$ when making a guess after observing a sequence S of test outcomes is positive iff*

$$\left|\Pr_{D,\vec{\alpha}}(\varphi \mid S) - 1/2\right| > q(b, g). \tag{9.1}$$

Proof. The expected payoff when guessing that the formula is true is

$$g \cdot \Pr_{D,\vec{\alpha}}(\varphi \mid S) + b \cdot (1 - \Pr_{D,\vec{\alpha}}(\varphi \mid S)).$$

This is greater than zero iff

$$(g - b) \Pr_{D,\vec{\alpha}}(\varphi \mid S) + b > 0,$$

that is, iff

$$\Pr_{D,\vec{\alpha}}(\varphi \mid S) - 1/2 > \frac{b}{b-g} - \frac{1}{2} = q(b, g).$$

When guessing that the formula is false, we simply exchange $\Pr_{D,\vec{\alpha}}(\varphi \mid S)$ and $1 - \Pr_{D,\vec{\alpha}}(\varphi \mid S)$ in the derivation. So the payoff is then positive iff

$$(1 - \Pr_{D,\vec{\alpha}}(\varphi \mid S)) - \frac{1}{2} = -(\Pr_{D,\vec{\alpha}}(\varphi \mid S) - \frac{1}{2}) > q(b, g).$$

Since $|x| = \max\{x, -x\}$, at least one of these two inequalities must hold if (9.1) does, so the corresponding guess will have positive expected payoff. Conversely, since $|x| \geq x$, either inequality holding implies (9.1). □

Proof of Proposition 9.2.1. Suppose that σ is a strategy for G with positive expected payoff. The test-outcome sequences of length k partition the space of paths in the game tree, so we have

$$\mathbb{E}_{G,\sigma}(\text{payoff}) = \sum_{\{S:|S|=k\}} \Pr_{D,\vec{\alpha},\sigma}(S)\, \mathbb{E}_{G,\sigma}(\text{payoff} \mid S).$$

Since the payoff is positive, at least one of the summands on the right must be, say the one due to the sequence S^*. By Lemma 9.2.3, $\left|\Pr_{D,\vec{\alpha}}(\varphi \text{ is true} \mid S^*) - 1/2\right| > q(b,g)$.

Let τ denote the random-test heuristic. Since τ chooses the optimal action after making k observations, it will not get a negative expected payoff for any sequence S of k test outcomes (since it can always obtain a payoff of 0 by choosing not to guess). On the other hand, with positive probability, the variables that make up the sequence S^* will be chosen and the outcomes in S^* will be observed for these tests; that is $\Pr_{D,\vec{\alpha},\tau}(S^*) > 0$. It follows from Lemma 9.2.3 that $\mathbb{E}_{G,\tau}(\text{payoff} \mid S^*) > 0$. Thus, $\mathbb{E}_{G,\tau}(\text{payoff}) > 0$, as desired. □

9.3 An example calculation of the optimal strategies

In this section, we fill in the details of the calculations for Example 9.1.1. We abuse notation by also viewing formulas, assignments, and test-outcome sequences as events in (i.e., subsets of) the space of histories of the information-acquisition game. Specifically,

- we identify a truth asignment A to the n variables in the game with the event consisting of all histories where A is the assignment chosen by nature;

- we identify the formula φ with the event consisting of all histories where φ is true under the assignment A chosen by nature; thus, φ is the disjoint union of all events A such that $\varphi(A) = T$;
- we identify a test-outcome sequence $S = (v_{i_1} \approx b_1, \ldots, v_{i_k} \approx b_k)$ of length k with the event consisting of all histories where at least k tests are performed, and the outcomes of the first k are described by S.

Observe that with the "good" payoff being +1 and the "bad" payoff being −16, the expected payoff from guessing that the formula is true after observing S is $\Pr_{D,\vec{\alpha}}(\varphi \mid S) \cdot 1 - \Pr_{D,\vec{\alpha}}(\neg\varphi \mid S) \cdot 16$, so it is greater than 0 if and only if $\Pr_{D,\vec{\alpha}}(\varphi \mid S) > 16/17$.

Henceforth, for brevity, let $A_{bb'}$ $(b, b' \in \{\mathrm{T}, \mathrm{F}\})$ refer to the assignment $\{v_1 \mapsto b, v_2 \mapsto b'\}$. By assumption, all test outcomes are independent conditional on a fixed assignment. Suppose first the player tests the same variable twice, say v_1. Then, for the "ideal" test outcome sequence $S = (v_1 \approx \mathrm{T}, v_1 \approx \mathrm{T})$, the conditional probability of S given that nature picked A is $(3/4) \cdot (3/4)$ if $A(v_1) = \mathrm{T}$, and $(1/4) \cdot (1/4)$ otherwise. It follows that

$$
\begin{aligned}
& \Pr_{D,\vec{\alpha}}(v_1 \vee v_2 \mid S) \\
= \; & \Pr_{D,\vec{\alpha}}(A_{\mathrm{TT}} \mid S) + \Pr_{D,\vec{\alpha}}(A_{\mathrm{TF}} \mid S) + \Pr_{D,\vec{\alpha}}(A_{\mathrm{FT}} \mid S) \\
= \; & \frac{\Pr_{D,\vec{\alpha}}(S \mid A_{\mathrm{TT}}) \Pr_{D,\vec{\alpha}}(A_{\mathrm{TT}}) + \ldots + \Pr_{D,\vec{\alpha}}(S \mid A_{\mathrm{FT}}) \Pr_{D,\vec{\alpha}}(A_{\mathrm{FT}})}{\Pr_{D,\vec{\alpha}}(S)} \\
= \; & \frac{\Pr_{D,\vec{\alpha}}(S \mid A_{\mathrm{TT}}) \Pr_{D,\vec{\alpha}}(A_{\mathrm{TT}}) + \ldots + \Pr_{D,\vec{\alpha}}(S \mid A_{\mathrm{FT}}) \Pr_{D,\vec{\alpha}}(A_{\mathrm{FT}})}{\sum_A \Pr_{D,\vec{\alpha}}(S \mid A) \Pr_{D,\vec{\alpha}}(A)} \\
= \; & \frac{((3/4)\cdot(3/4)+(3/4)\cdot(3/4)+(1/4)\cdot(1/4))\cdot(1/4)}{((3/4)\cdot(3/4)+(3/4)\cdot(3/4)+(1/4)\cdot(1/4)+(1/4)\cdot(1/4))\cdot(1/4)} \\
= \; & \frac{(19/16)\cdot(1/4)}{(20/16)\cdot(1/4)} \\
= \; & 19/20 > 16/17.
\end{aligned}
$$

Thus, the agent will guess true after observing S, and get a positive expected payoff (since S will be observed with positive probability) as a consequence of

testing v_1 twice. Symmetrically, testing v_2 twice gives a positive expected payoff.

On the other hand, suppose the player tests two different variables. The best case would be to get $S = (v_1 \approx \mathrm{T}, v_2 \approx \mathrm{T})$. As before, the probability of S conditioned on some assignment is the product of the probabilities for each of its entries being observed; for instance, $\Pr_{D,\vec{\alpha}}(S \mid A_{\mathrm{TF}}) = (3/4) \cdot (1/4)$. So we get

$$\begin{aligned}
& \Pr_{D,\vec{\alpha}}(v_1 \vee v_2 \mid S) \\
= \; & \Pr_{D,\vec{\alpha}}(A_{\mathrm{TT}} \mid S) + \Pr_{D,\vec{\alpha}}(A_{\mathrm{TF}} \mid S) + \Pr_{D,\vec{\alpha}}(A_{\mathrm{FT}} \mid S) \\
= \; & \frac{\Pr_{D,\vec{\alpha}}(S|A_{\mathrm{TT}}) \Pr_{D,\vec{\alpha}}(A_{\mathrm{TT}}) + \ldots + \Pr_{D,\vec{\alpha}}(S|A_{\mathrm{FT}}) \Pr_{D,\vec{\alpha}}(A_{\mathrm{FT}})}{\Pr_{D,\vec{\alpha}}(S)} \\
= \; & \frac{\Pr_{D,\vec{\alpha}}(S|A_{\mathrm{TT}}) \Pr_{D,\vec{\alpha}}(A_{\mathrm{TT}}) + \ldots + \Pr_{D,\vec{\alpha}}(S|A_{\mathrm{FT}}) \Pr_{D,\vec{\alpha}}(A_{\mathrm{FT}})}{\sum_A \Pr_{D,\vec{\alpha}}(S|A) \Pr_{D,\vec{\alpha}}(A)} \\
= \; & \frac{((3/4)\cdot(3/4)+(3/4)\cdot(1/4)+(1/4)\cdot(3/4))\cdot(1/4)}{((3/4)\cdot(3/4)+(3/4)\cdot(1/4)+(1/4)\cdot(3/4)+(1/4)\cdot(1/4))\cdot(1/4)} \\
= \; & \frac{(15/16)\cdot(1/4)}{(16/16)\cdot(1/4)} \\
= \; & 15/16 < 16/17.
\end{aligned}$$

An analogous calculation shows that if either of the tests comes out false, the conditional probability is even lower. Thus, after testing different variables, the agent will not make a guess, no matter what the outcome, and so has an expected payoff of 0.

So, indeed, measuring the same variable twice is strictly better than measuring each of them once.

CHAPTER 10

RATIONAL INATTENTION

10.1 Defining rational inattention

We might think that an optimal strategy for learning about φ would test all variables that are relevant to φ (given a sufficiently large test budget). As shown in Example 9.1.1, this may not be true. For example, an optimal k-step strategy for $v_1 \vee v_2$ can end up never testing v_1, no matter what the value of k, if it starts by testing v_2 and keeps discovering that v_2 is true. It turns out that RI is quite widespread.

It certainly is not surprising that if a variable v does not occur in φ, then an optimal strategy would not test v. More generally, it would not be surprising that a variable that is not particularly relevant to φ is not tested too often, perhaps because it makes a difference only in rare edge cases. In the foraging animal example from the introduction, the possibility of a human experimenter having prepared a safe food to look like a known poisonous plant would impact whether it is safe to eat, but is unlikely to play a significant role in day-to-day foraging strategies. What might seem more surprising is if a variable v is (largely) ignored while another variable v' that is no more relevant than v is tested. This is what happens in Example 9.1.1; although we have not yet defined a notion of relevance, symmetry considerations dictate that v_1 and v_2 are equally relevant to $v_1 \vee v_2$, yet an optimal strategy might ignore one of them.

The phenomenon of rational inattention observed in Example 9.1.1 is surprisingly widespread. To make this claim precise, we need to define "rele-

vance". There are a number of reasonable ways of defining it; we focus on one below.[1] The definition of the relevance of v to φ that we use counts the number of truth assignments for which changing the truth value of v changes the truth value of φ.

Definition 10.1.1. Define the relevance ordering $\leq_\varphi$ on the variables in φ by taking

$$\begin{aligned} & v \leq_\varphi v' \text{ iff} \\ & |\{A : \varphi(A[v \mapsto \mathrm{T}]) \neq \varphi(A[v \mapsto \mathrm{F}])\}| \\ \leq\ & |\{A : \varphi(A[v' \mapsto \mathrm{T}]) \neq \varphi(A[v' \mapsto \mathrm{F}])\}|, \end{aligned}$$

where $A[v \mapsto b]$ is the assignment that agrees with A except that it assigns truth value b to v. □

Thus, rather than saying that v is or is not relevant to φ, we can say that v is (or is not) at least as relevant to φ as v'. Considering the impact of a change in a single variable to the truth value of the whole formula in this fashion has been done both in the cognitive science and the computer science literature: for example, Vigo ([41]) uses the *discrete (partial) derivative* to capture this effect, and Lang et al. ([25]) define the related notion of *Var-independence*.

We could also consider taking the probability of the set of truth assignments where a variable's value makes a difference, rather than just counting how many such truth assignments there are. This would give a more detailed quantitative view of relevance, and is essentially how relevance is considered in Bayesian networks. Irrelevance is typically identified with independence. Thus, v is relevant to φ if a change to v changes the probability of φ. (See Druzdzel and Suermondt ([10]) for a review of work on relevance in the context of Bayesian

[1] We checked various other reasonable definitions experimentally; qualitatively, it seems that our results continue to hold for all the variants that we tested.

networks.) We did not consider a probabilistic notion of relevance because then the relevance order would depend on the game (specifically, the distribution D, which is one of the parameters of the game). Our definition makes the relevance order depend only on φ. That said, we believe that essentially the same results as those that we prove could be obtained for a probabilistic notion of relevance ordering.

Roughly speaking, φ exhibits RI if, for all optimal strategies σ for the game $G(\varphi, D, k, \vec{\alpha}, b, g)$, σ tests a variable v' frequently while hardly ever testing a variable v that is at least as relevant to φ as v'. We still have to make precise "hardly ever", and explain how the claim depends on the choice of D, $\vec{\alpha}$, k, b, and g. For the latter point, note that in Example 9.1.1, we had to choose b and g appropriately to get RI. This turns out to be true in general; given D, k, and $\vec{\alpha}$, the claim holds only for an appropriate choice of b and g that depends on these. In particular, for any fixed choice of b and g that depends only on k and $\vec{\alpha}$, there exist choices of priors D for which the set of optimal strategies is fundamentally uninteresting: we can simply set D to assign a probability to some truth assignment A that is so high that the rational choice is always to guess $\varphi(A)$, regardless of the test outcomes.

Another way that the set of optimal strategies can be rendered uninteresting is when, from the outset, there is no hope of obtaining sufficient certainty of the formula's truth value with the k tests available. Similarly to when the truth value is a foregone conclusion, in this situation, an optimal strategy can perform arbitrary tests, as long as it makes no guess at the end. More generally, even when in general the choice of variables to test does matter, a strategy can reach a situation where there is sufficient uncertainty that no future test outcome could

affect the final choice. Thus, a meaningful definition of RI that is based on the variables tested by optimal strategies must consider only tests performed in those cases in which a guess actually should be made (because the expected payoff of the optimal strategy is positive).[2] We now make these ideas precise.

Definition 10.1.2. A function $f : \mathbb{N} \to \mathbb{N}$ is *negligible* if $f(k) = o(k)$, that is, if $\lim_{k\to\infty} f(k)/k = 0$. □

The idea is that φ exhibits RI if, as the number k of tests allowed increases, the fraction of times that some variable v is tested is negligible relative to the number of times that another variable v' is tested, although v is at least as relevant to φ as v'. We actually require slightly more: we want v' to be tested a linear number of times (i.e., at least ck times, for some constant $c > 0$). (Note that this additional requirement makes it harder for a variable to exhibit RI.)

Since we do not want our results to depend on correlations between variables, we restrict attention to probability distributions D on truth assignments that are product distributions.

Definition 10.1.3. A probability distribution D on truth assignments to $v_1, \ldots, v_n$ is a *product distribution* if $\Pr_D(A) = \Pr_D(v_1 = A(v_1)) \cdots \Pr_D(v_n = A(v_n))$ (where, for an arbitrary formula φ, $\Pr_D(\varphi) = \sum_{\{A:\, A(\varphi)=\mathrm{T}\}} \Pr_D(A)$). □

As discussed earlier, to get an interesting notion of RI, we need to allow the choice of payoffs b and g to depend on the prior distribution D; for fixed b, g, and testing bound k, if the distribution D places sufficiently high probability

[2] One way to avoid these additional requirements is to modify the game so that performing a test is associated has a small but positive cost, so that an optimal strategy avoids frivolous testing when the conclusion is foregone. The definitions we use have essentially the same effect, and are easier to work with.

on a single assignment, no k outcomes can change the agent's mind. Similarly, assigning prior probability 1 to any one variable being true or false means that no tests will change the agent's mind about that variable, and so testing it is pointless (and the game is therefore equivalent to one played on the formula in $n - 1$ variables where this variable has been replaced by the appropriate truth value). We say that a probability distribution that gives all truth assignments positive probability is *open-minded*.

With all these considerations in hand, we can finally define RI formally.

Definition 10.1.4. The formula φ *exhibits rational inattention* if, for all open-minded product distributions D and uniform accuracy vectors $\vec{\alpha}$ (those with $(\alpha_1 = \ldots = \alpha_n)$), there exists a negligible function f and a constant $c > 0$ such that for all k, there are payoffs b and g such that all optimal strategies in the information-acquisition game $G(\varphi, D, k, \vec{\alpha}, b, g)$

have positive expected payoff and, in all histories of the game, either make no guess or

- test a variable v' at least ck times, but
- test a variable v such that $v' \leq_\varphi v$ at most $f(k)$ times. □

Our definition of RI is quite strong; for instance, as we discuss at the end of Section 10, we could obtain a plausible weakening by requiring that a variable v' at least as relevant as a variable v is tested far less frequently than v in a set of histories with positive probability rather than in *all* histories. But even with our strong requirements, we find that rational inattention in the given strong sense is quite widespread.

To get an intuition for this definition of RI, we will first directly check whether some natural classes of formulae satisfy it.

Example 10.1.5. (Rational inattention)

1. Conjunctions $\varphi = \bigwedge_{i=1}^{N} \ell_i$ and disjunctions $\varphi = \bigvee_{i=1}^{N} \ell_i$ of $N \geq 2$ literals (variables $\ell_i = v_i$ or their negations $\neg v_i$) exhibit RI. In each case, we can pick b and g such that all optimal strategies pick one variable and focus on it, either to establish that the formula is false (for conjunctions) or that it is true (for disjunctions). By symmetry, all variables v_i and v_j are equally relevant, so $v_i \leq_\varphi v_j$.

2. The formulae v_i and $\neg v_i$ do not exhibit RI. There is no variable $v \neq v_i$ such that $v_i \leq_{(\neg)v_i} v$, and for all choices of b and g, the strategy of testing only v_i and ignoring all other variables (making an appropriate guess in the end) is clearly optimal for $(\neg)v_i$.

3. More generally, we can say that all XORs in ≥ 0 variables do not exhibit RI. For the constant formulae T and F, any testing strategy that "guesses" correctly is optimal; for a XOR in more than one variable, an optimal strategy must test all of the variables about the same number of times, as any remaining uncertainty about the truth value of some variable leads to at least equally great uncertainty about the truth value of the whole formula. Similarly, negations of XORs do not exhibit RI. Together with the preceding two points, this means that the only formulae in 2 variables exhibiting rational inattention are those equivalent to one of the four conjunctions $\ell_1 \wedge \ell_2$ or the four disjunctions $\ell_1 \vee \ell_2$ in which each variable occurs exactly once and may or may not be negated.

4. For $n > 2$, formulae φ of the form $v_1 \vee (\neg v_1 \wedge v_2 \wedge \ldots \wedge v_n)$) do not exhibit RI.

Optimal strategies that can attain a positive payoff at all will start by testing v_1; if the tests come out true, it will be optimal to continue testing v_1, ignoring $v_2 \ldots v_n$. However, for formulae φ of this form, v_1 is strictly more relevant than the other variables: there are only 2 assignments where changing v_i flips the truth value of the formula for $i > 1$ (the two where $v_1 \mapsto F$ and $v_j \mapsto T$ for $j \notin \{1, i\}$) but $2^n - 2$ assignments where changing v_1 does (all but the two where $v_j \mapsto T$ for $j \neq 1$). Hence, in the event that all these tests actually succeed, the only variables that are ignored are not at least as relevant as the only one that isn't, so φ does not exhibit RI.

5. For $n > 4$, formulae φ of the form $(v_1 \vee v_2) \wedge (v_3 \oplus \ldots \oplus v_n)$ exhibit RI. Optimal strategies split tests between v_1 and v_2, and try to establish that both are false, and hence that φ is. To establish that φ is true would require showing that the XOR is true. This, in turn, would require testing all of $v_3, \ldots, v_n$; since $n > 4$, there are at least three variables to test. As we noted earlier, an optimal strategy must test each of the variables in the XOR about the same number of times to establish the truth (or falsity) of the XOR. It thus requires signficantly more tests to gain a given level of confidence that the XOR is true (or false) than it does to gain that level of confidence that $v_1 \vee v_2$ is false. (In Section 11, we show that XORs are the formulae that we learn the least about in a given number of tests among all formulas with a fixed number of variables.) The variable v_1 determines whether is true in only φ in 1/4 of the assignments (when the XOR is true, which it is in half the assignments, and v_2 is false); similarly for v_2. On the other hand, $v_3, \ldots, v_n$ determine the truth value of φ in 3/4 of all assignments (all assignments where $v_1 \vee v_2$ is true). Thus, this family of formulae (and other similar families) satisfy an even stronger definition of RI, as a strictly less relevant variable is preferred. □

10.2 A sufficient criterion for rational inattention

Unfortunately, as far as we know, determining the optimal strategies is hard in general. To be able to reason about whether φ exhibits RI in a tractable way, we find it useful to consider optimal test-outcome sequences.

Definition 10.2.1. A sequence S of test outcomes is *optimal* for a formula φ, prior D, and accuracy vector $\vec{\alpha}$ if it minimises the conditional uncertainty about the truth value of φ among all test-outcome sequences of the same length. That is, $\left|\Pr_{D,\vec{\alpha}}(\varphi \mid S) - \frac{1}{2}\right| \geq \left|\Pr_{D,\vec{\alpha}}(\varphi \mid S') - \frac{1}{2}\right|$ for all S' with $|S'| = |S|$. □

It turns out that for a formula to exhibit rational inattention, it is sufficient (but not necessary!) for just the optimal test-outcome sequences to be "inattentive", because we can set up the payoffs in such a way that only the very best test-outcome sequences ever become relevant (by possibly leading to a non-negative payoff). By doing this, we avoid having to deal with the complicated quantification over all histories in the definition of RI. With the appropriate payoffs, each history either has to end in no guess or contain an optimal test-outcome sequence. We will see that we can reason about optimal test-outcome sequences without having to worry about the structure of arbitrary optimal strategies.

Proposition 10.2.2. *Suppose that, for a given formula φ, for all open-minded product distributions D and uniform accuracy vectors $\vec{\alpha}$, there exists a negligible function f and a constant $c > 0$ such that for all testing bounds k, the test-outcome sequences S optimal for φ, D, and $\vec{\alpha}$ of length k have the following two properties:*

- *S has at least ck tests of some variable v', but*
- *S has at most $f(k)$ tests of some variable $v \geq_\varphi v'$.*

Then φ exhibits RI.

Proof. Let $P(\varphi, D, \vec{\alpha}, f, c, k)$ denote the statement that for all test-outcomes sequences S that are optimal for φ, D, and $\vec{\alpha}$, there exist variables $v \geq_\varphi v'$ such that S contains $\geq ck$ tests of v' and $\leq f(k)$ tests of v. We now prove that for all φ, D, $\vec{\alpha}$, f, c, and k, $P(\varphi, D, \vec{\alpha}, f, c, k)$ implies the existence of b and g such that φ exhibits RI in the game $G(\varphi, D, k, m, b, g)$. It is easy to see that this suffices to prove the proposition.

Fix φ, D, $\vec{\alpha}$, f, c, and k, and suppose that $P(\varphi, D, \vec{\alpha}, f, c, k)$ holds. Let

$$q^* = \max_{\{S : |S| = k\}} \left| \Pr_{D,\vec{\alpha}}(\varphi | S) - \frac{1}{2} \right|.$$

Assume for now that $q^* > 0$. Since there are only finitely many test-outcome sequences of length k, there must be some ε with $q^* > \varepsilon > 0$ sufficiently small such that for all S with $|S| = k$, $|\Pr_{D,\vec{\alpha}}(\varphi|S) - \frac{1}{2}| > q^* - \varepsilon$ iff $|\Pr_{D,\vec{\alpha}}(\varphi|S) - \frac{1}{2}| = q^*$. Choose the payoffs b and g such that the threshold $q(b, g)$ is $q^* - \varepsilon$. We show that φ exhibits RI in the game $G(\varphi, D, k, m, b, g)$.

Let $\mathcal{S}_k = \{S : |S| = k \text{ and } |\Pr_{D,\vec{\alpha}}(\varphi|S) - \frac{1}{2}| = q^*\}$ be the set of test-outcome sequences of length k optimal for φ, D, and $\vec{\alpha}$. If σ is an optimal strategy for the game $G(\varphi, D, k, \vec{\alpha}, g, b)$, the only sequences of test outcomes after which σ makes a guess are the ones in $\mathcal{S}_k$. For if a guess is made after seeing some test-outcome sequence $S^* \notin \mathcal{S}_k$, by Lemma 9.2.3 and the choice of b and g, the expected payoff of doing so must be negative, so the strategy σ' that is identical to σ except that it makes no guess if S^* is observed is strictly better than σ, contradicting the optimality of σ. So whenever a guess is made, it must be after a sequence $S \in \mathcal{S}_k$ was observed. Since sequences in $\mathcal{S}_k$ are optimal for φ, D, and $\vec{\alpha}$, and $P(\varphi, D, \vec{\alpha}, f, c, k)$ holds by assumption, this sequence S must contain $\geq ck$ test of

v' and $\leq f(k)$ test of v.

All that remains to show that φ exhibits RI in the game $G(\varphi, D, k, \vec{\alpha}, g, b)$ is to establish that all optimal strategies have positive expected payoff. To do this, it suffices to show that there is a strategy that has positive expected payoff. Let S be an arbitrary test-outcome sequence in $\mathcal{S}_k$. Without loss of generality, we can assume that $\Pr_{D,\vec{\alpha}}(\varphi \mid S) > 1/2$. Let σ_S be the strategy that tests every variable the number of times that it occurs in S in the order that the variables occur in S, and guesses that the formula is true iff S was in fact the test-outcome sequence observed (and makes no guess otherwise). Since S will be observed with positive probability, it follows from Lemma 9.2.3 that σ_S has positive expected payoff.

It remains to address the case where $q^* = 0$, that is, the number of tests k is insufficient to learn anything about the truth value of the formula. In this case, we can simply set $b = -1$ and $g = 2$, and proceed as before, needing to show only that some strategy attains a positive payoff. Indeed, take σ_T to be a strategy that repeatedly tests some variable v and then guesses T regardless of outcomes. Since $q^* = 0$, so we have that $\Pr_{D,\vec{\alpha}}(\varphi \mid S) = 1/2$ for all test-outcome sequences, there is a probability $1/2$ of getting payoff -1 and a probability $1/2$ of getting payoff 2, so the expected payoff is positive. This completes the proof. □

10.2.1 Characteristic fractions and traces

Applying Proposition 10.2.2 to test whether a formula exhibits RI is not trivial. It is easy to show that all that affects $\Pr_{D,\vec{\alpha}}(\varphi \mid S)$ is the number of times that each variable is tested and the outcome of the test, not the order in which the tests

were made. We establish this formally in Lemma 10.2.6; the following notation, which we use throughout the rest of this section and in Section 10.2.2, implicitly assumes this fact.

Definition 10.2.3. (General notation)

- $o_i = \frac{1/2+\alpha_i}{1/2-\alpha_i}$. We can think of o_i as the odds of making a correct observation of v_i; namely, the probability of observing $v_i \approx b$ conditional on v_i actually being b divided by the probability of observing $v_i \approx b$ condition on v_i not being b.
- $n^+_{S,A,i} = |\{j : S[j] = (v_i \approx A(v_i))\}|$. Thus, $n^+_{S,A,i}$ is the number of times that v_i is observed to have the correct value according to truth assignment A in test-outcome sequence S.
- $r_{D,\vec{\alpha}}(A, S) = \Pr_{D,\vec{\alpha}}(A) \prod_{\{i:v_i \text{ is in the domain of } A\}} o_i^{n^+_{S,A,i}}$ □

Our goal is to show that a large proportion of Boolean formulae exhibit RI. To this end, we would like a method to establish that a particular formula exhibits RI that is sufficiently efficient that we can run it on all formulae of a given size, or at least a statistically significant sample. Throughout this section, we focus on some arbitrary but fixed formula φ in n variables $v_1, \ldots, v_n$. Proposition 10.2.2 gives a sufficient criterion for φ to exhibit RI in terms of the structure of the optimal sequences of test outcomes of each length. To make use of this criterion, we introduce some machinery to reason about optimal sequences of test outcomes. The key definition turns out to be that of the *characteristic fraction* of S for φ, denoted $\mathrm{cf}(\varphi, s)$, which is a quantity that is inversely ordered to $\Pr_{D,\vec{\alpha}}(\varphi \mid S)$ (Lemma 10.2.5) (so the probability is maximised iff the characteristic fraction is minimised and vice versa), while exhibiting several convenient properties that

enable the subsequent analysis. Let o_i represent the odds of making a correct observation of v_i, namely, the probability of observing $v_i \approx b$ conditional on v_i actually being b divided by the probability of observing $v_i \approx b$ conditional on v_i not being b. If we assume that $o_i = o_j$ for all variables i and j, and let o represent this expression, then $\mathrm{cf}(\varphi, S)$ is the quotient of two polynomials, and has the the form

$$\frac{c_1 o^{d_1|S|} + \ldots + c_{2^n} o^{d_{2^n}|S|}}{e_1 o^{f_1|S|} + \ldots + e_{2^n} o^{f_{2^n}|S|}},$$

where c_j, d_j, e_j, and f_j are terms that depend on the truth assignment A_j, so we have one term for each of the 2^n truth assignments, and $0 \leq d_j, f_j \leq 1$. For a test-outcome sequence S that is optimal for φ, we can show that $f_j = 1$ for some j. Thus, the most significant term in the denominator (i.e., the one that is largest, for $|S|$ sufficiently large) has the form $eo^{|S|}$. We call the factor d_i before $|S|$ in the exponent of the leading term of the numerator the *max-power* (Definition 10.2.15) of the characteristic function. We can show that the max-power is actually independent of S (if S is optimal for φ). Since we are interested in the test-outcome sequence S for which $\mathrm{cf}(\varphi, S)$ is minimal (which is the test-outcome sequence for which $\mathrm{Pr}_{D,\vec{\alpha}}(\varphi|S)$ is maximal), for each k, we want to find that S of length k whose max-power is minimal. As we show, we can find the sequence S whose max-power is minimal by solving a linear program (Definition 10.2.17).

Formally, we define the characteristic fraction as follows.

Definition 10.2.4. The *characteristic fraction* of a test-outcome sequence S for φ is

$$\mathrm{cf}(\varphi, S) = \frac{\sum_{\{A:\, \varphi(A)=\mathrm{F}\}} r_{D,\vec{\alpha}}(A, S)}{\sum_{\{A:\, \varphi(A)=\mathrm{T}\}} r_{D,\vec{\alpha}}(A, S)}.$$

□

The importance of this quantity is due to the following:

Lemma 10.2.5. $\Pr_{D,\vec{\alpha}}(\varphi \mid S) > \Pr_{D,\vec{\alpha}}(\varphi \mid S')$ *iff* $\mathrm{cf}(\varphi, S) < \mathrm{cf}(\varphi, S')$.

To prove this lemma, we first start with another lemma that gives a straightforward way of calculating $\Pr_{D,\vec{\alpha}}(A \mid S)$ for an assignment A and a test-outcome sequence S. The lemma also shows that, as the notation suggests, the probability is independent of the strategy σ.

Lemma 10.2.6. *For all accuracy vectors $\vec{\alpha}$, product distributions D, assignments A, and test-outcome sequences S,*

$$\Pr_{D,\vec{\alpha}}(A \mid S) = \frac{r_{D,\vec{\alpha}}(A, S)}{\sum_{\text{truth assignments } A'} r_{D,\vec{\alpha}}(A', S)}.$$

Thus,

$$\Pr_{D,\vec{\alpha}}(\varphi \mid S) = \sum_{\{A:\, \varphi(A)=\mathrm{T}\}} \Pr_{D,\vec{\alpha}}(A \mid S) = \frac{\sum_{\{A:\, \varphi(A)=\mathrm{T}\}} r_{D,\vec{\alpha}}(A, S)}{\sum_{A'} r_{D,\vec{\alpha}}(A', S)}.$$

These probabilities do not depend on the strategy σ.

Proof. By Bayes' rule, for all truth assignments A and sequences $S = [v_{i_1} \approx b_1, \ldots, v_{i_k} \approx b_k]$ of test outcomes, we have

$$\begin{aligned} \Pr_{D,\vec{\alpha},\sigma}(A \mid S) &= \frac{\Pr_{D,\vec{\alpha},\sigma}(S \mid A) \Pr_{D,\vec{\alpha}}(A)}{\Pr_{D,\vec{\alpha},\sigma}(S)} \\ &= \frac{\Pr_{D,\vec{\alpha},\sigma}(S \mid A) \Pr_{D,\vec{\alpha}}(A)}{\sum_{\text{truth assignments } A'} \Pr_{D,\vec{\alpha},\sigma}(S \mid A') \Pr_{D,\vec{\alpha}}(A')}. \end{aligned} \tag{10.1}$$

Suppose that $S = (v_{i_1} \approx b_1, \ldots, v_{i_k} \approx b_k)$. We want to compute $\Pr_{D,\vec{\alpha},\sigma}(S \mid A')$ for an arbitrary truth assignment A'. Recall that a strategy σ is a function from test-outcome sequences to a distribution over actions. We write $\sigma_S(\text{test } v)$ to denote the probability that σ tests v given test-outcome sequence S and use $()$ for the empty sequence; more generally, we denote by $\text{test}_j(v)$ the event that the jth variable chosen was v. Then,

$$\begin{aligned} \Pr_{D,\vec{\alpha},\sigma}(S \mid A') = {} & \sigma_{()}(\text{test}_1(v_{i_1})) \Pr_{D,\vec{\alpha},\sigma}((v_{i_1} \approx b_1) \mid \text{test}_1(v_{i_1}), A') \ldots \\ & \sigma_{(v_{i_1} \approx b_1, \ldots, v_{i_{k-1}} \approx b_{k-1})}(\text{test}_k(v_{i_k})) \Pr_{D,\vec{\alpha},\sigma}((v_{i_k} \approx b_k) \mid \text{test}_k(v_{i_k}), A'). \end{aligned}$$

Here, we were able to write $\Pr_{D,\vec{\alpha},\sigma}((v_{i_j} \approx b_j) \mid \text{test}_j(v_{i_j}), A')$ without conditioning on the entire test-outcome sequence up to $v_{i_{j-1}}$ because by the definition of the information-acquisition game, all observations are independent of each other conditioned on the assignment A'. Observe that the terms $\sigma_{()}(\text{test}_1(v_{i_1}))$, ..., $\sigma_{(v_{i_1} \approx b_1, \ldots, v_{i_{k-1}} \approx b_{k-1})}(\text{test}_k(v_{i_k}))$ are common to $\Pr_{D,\vec{\alpha},\sigma}(S \mid A')$ for all truth assignments A', so we can pull them out of the numerator and denominator in (10.1) and cancel them. Moreover, probabilities of the form $\Pr_{D,\vec{\alpha},\sigma}((v_{i_j} \approx b_j) \mid \text{test}_j(v_{i_j}), A')$ do not depend on the strategy σ, so we can drop it from the subscript of $\Pr_{D,\vec{\alpha},\sigma}$; the probability also does not depend on the results of earlier tests (since, by assumption, test outcomes are independent, conditional on the truth assignment). Thus, it follows that

$$\begin{aligned}
&\Pr_{D,\vec{\alpha},\sigma}(A \mid S) \\
&= \frac{\left[\prod_{j=1}^{k} \Pr_{D,\vec{\alpha}}(v_{i_j} \approx b_j \text{ observed} \mid v_{i_j} \text{ chosen}, A)\right] \Pr_{D,\vec{\alpha}}(A)}{\sum_{\text{truth assignments } A'} \left[\prod_{j=1}^{k} \Pr_{D,\vec{\alpha}}(v_{i_j} \approx b_j \text{ observed} \mid v_{i_j} \text{ chosen}, A')\right] \Pr_{D,\vec{\alpha}}(A')}.
\end{aligned}$$

Next, we multiply both the numerator and the denominator of this fraction by $\prod_{j=1}^{k} \frac{1}{1/2-\alpha_{i_j}}$. This amounts to multiplying the jth term in each product by $\frac{1}{1/2-\alpha_{i_j}}$. Thus, in the numerator, if $b_j = A(v_{i_j})$, then the jth term in the product equals o_{i_j}; if $b_j = \neg A(v_{i_j})$, then the jth term in the product is 1. It easily follows that this expression is just $r_{D,\vec{\alpha}}(A, S)$. A similar argument shows that the denominator is $\sum_{\text{truth assignments } A'} r_{D,\vec{\alpha}}(A', S)$. This proves the first and third statements in the lemma. The second statement is immediate from the first. □

The next lemma gives an intuitive property of those test-outcome sequences S that are *optimal* for φ, D, and $\vec{\alpha}$.

Lemma 10.2.7. *If S is a test-outcome sequence that is optimal for φ, D, and $\vec{\alpha}$, and $\Pr_{D,\vec{\alpha}}(\varphi|S) \neq \Pr_{D,\vec{\alpha}}(\varphi) > 0$, then S does not contain observations both of the form*

$v_i \approx T$ *and of the form* $v_i \approx F$ *for any* v_i.

Proof. Suppose that S is optimal for φ, D, and $\vec{\alpha}$, $\Pr_{D,\vec{\alpha}}(\varphi \mid S) \neq \Pr_{D,\vec{\alpha}}(\varphi)$, there are $n_1 > 0$ instance of $v_i \approx T$ in S, and $n_2 > 0$ instances of $v_i \approx F$ in S. Without loss of generality, suppose that $n_1 > n_2$. Let S_0 be the sequence that results from S by removing the n_2 occurrences of $v_i \approx F$ and the last n_2 occurrences of $v_i \approx T$. Thus, $|S_0| = |S| - 2n_2 < |S|$. It is easy to see that, for each truth assignment A, we have $n^+_{S,A,i} = n^+_{S_0,A,i} + n_2$. It thus follows from Lemma 10.2.6 that $\Pr_{D,\vec{\alpha}}(\varphi \mid S) = \Pr_{D,\vec{\alpha}}(\varphi \mid S_0)$. We can similarly remove all other "contradictory" observations to get a sequence S_0 that does not contradict itself such that $|S_0| < |S|$ and $\Pr_{D,\vec{\alpha}}(\varphi \mid S) = \Pr_{D,\vec{\alpha}}(\varphi \mid S_0)$.

Suppose without loss of generality that $\Pr_{D,\vec{\alpha}}(\varphi) - 1/2 \geq 0$. Since it cannot be the case that for every test-outcome sequence S_0 of length $|S|$ we have $\Pr_{D,\vec{\alpha}}(\varphi \mid S_0) - 1/2 < \Pr_{D,\vec{\alpha}}(\varphi) - 1/2$, and S is optimal for φ, D, and $\vec{\alpha}$, we must have

$$\Pr_{D,\vec{\alpha}}(\varphi \mid S) - 1/2 \geq |\Pr_{D,\vec{\alpha}}(\varphi) - 1/2|. \tag{10.2}$$

We want to show that we can add tests to S_0 to get a sequence S^* with $|S^*| = |S|$ such that $\Pr_{D,\vec{\alpha}}(\varphi \mid S^*) > \Pr_{D,\vec{\alpha}}(\varphi \mid S_0) = \Pr_{D,\vec{\alpha}}(\varphi \mid S)$. This will show that S is not optimal for φ, D, and $\vec{\alpha}$, giving us the desired contradiction.

Suppose that $S_0 = (v_{i_1} \approx b_1, \ldots, v_k \approx b_k)$. Define test-outcome sequences $S_1, \ldots, S_k$ inductively by taking S_j to be S_{j-1} with $v_{i_j} \approx b_j$ removed if $\Pr_{D,\vec{\alpha}}(\varphi \mid S_{j-1}) \leq \Pr_{D,\vec{\alpha}}(\varphi \mid S_{j-1} \setminus (v_{i_j} \approx b_j))$ and otherwise taking $S_j = S_{j-1}$. It is immediate from the construction that $\Pr_{D,\vec{\alpha}}(\varphi \mid S_k) \geq \Pr_{D,\vec{\alpha}}(\varphi \mid S_0) = \Pr_{D,\vec{\alpha}}(\varphi \mid S)$ and $|S_k| \leq |S_0| < |S|$. It cannot be the case that $|S_k| = 0$, for then $\Pr_{D,\vec{\alpha}}(\varphi) \geq \Pr_{D,\vec{\alpha}}(\varphi \mid S)$. Since $\Pr_{D,\vec{\alpha}}(\varphi) \neq \Pr_{D,\vec{\alpha}}(\varphi \mid S)$ by assumption, we would have $\Pr_{D,\vec{\alpha}}(\varphi) > \Pr_{D,\vec{\alpha}}(\varphi \mid S)$, contradicting (10.2).

Suppose that $v_i \approx b$ is the last test in S_k. Let $S_k^- = S_k \setminus (v_i \approx b)$, so that $S_k = S_k^- \cdot (v_i \approx b)$. By construction, $\Pr_{D,\vec{\alpha}}(\varphi \mid S_k) > \Pr_{D,\vec{\alpha}}(\varphi \mid S_k^-)$. That is, observing $v \approx b$ increased the conditional probability of φ. We now show that observing $v \approx b$ more often increases the conditional probability of φ further; that is, for all m, $\Pr_{D,\vec{\alpha}}(\varphi \mid (S_k \cdot (v_i \approx b)^m) > \Pr_{D,\vec{\alpha}}(\varphi \mid S_k)$. We can thus take $S^* = (S_k \cdot (v_i \approx b)^{|S|-|S_k|})$.

It follows from Lemma 10.2.6 that

$$\Pr_{D,\vec{\alpha}}(\varphi \mid S_k) = \sum_{\{A:\ \varphi(A)=T\}} \Pr_{D,\vec{\alpha}}(A \mid S_k) = \frac{\sum_{\{A:\ \varphi(A)=T\}} r_{D,\vec{\alpha}}(A, S_k)}{\sum_{\text{truth assignments } A'} r_{D,\vec{\alpha}}(A', S_k)}$$

$$\text{and } \Pr_{D,\vec{\alpha}}(\varphi \mid S_k^-) = \sum_{\{A:\ \varphi(A)=T\}} \Pr_{D,\vec{\alpha}}(A \mid S_k^-) \frac{\sum_{\{A:\ \varphi(A)=T\}} r_{D,\vec{\alpha}}(A, S_k^-)}{\sum_{\text{truth assignments } A'} r_{D,\vec{\alpha}}(A', S_k^-)}.$$

Note that for all truth assignments A', $r_{D,\vec{\alpha}}(A', S_k) = r_{D,\vec{\alpha}}(A', S_k^-)$ if $A'(v_i) \neq b$, and $r_{D,\vec{\alpha}}(A', S_k) = o_i r_{D,\vec{\alpha}}(A', S_k^-)$ if $A'(v_i) = b$. Thus, there exist x_1, x_2, y_1, y_2 such that $\Pr_{D,\vec{\alpha}}(\varphi \mid S_k^-) = \frac{x_1+x_2}{y_1+y_2}$ and $\Pr_{D,\vec{\alpha}}(\varphi \mid S_k) = \frac{o_i x_1+x_2}{o_i y_1+y_2}$. Indeed, we can take

$$x_1 = \sum_{\{A:\varphi(A)=T, A(v_i)=b\}} r_{D,\vec{\alpha}}(S_k, A), \qquad x_2 = \sum_{\{A:\varphi(A)=T, A(v_i)\neq b\}} r_{D,\vec{\alpha}}(S_k, A),$$

$$y_1 = \sum_{\{A:A(v_i)=b\}} r_{D,\vec{\alpha}}(S_k, A), \text{ and} \qquad y_2 = \sum_{\{A:A(v_i)\neq b\}} r_{D,\vec{\alpha}}(S_k, A).$$

Since $\Pr_{D,\vec{\alpha}}(\varphi \mid S_k) > \Pr_{D,\vec{\alpha}}(\varphi \mid S_k^-)$, we must have

$$\frac{o_i x_1 + x_2}{o_i y_1 + y_2} > \frac{x_1 + x_2}{y_1 + y_2}. \tag{10.3}$$

Since $x_1, x_2, y_1, y_2 \geq 0$, crossmultiplying shows that (10.3) holds iff

$$x_2 y_1 + o_i x_1 y_2 > x_1 y_2 + o_i x_2 y_1.$$

Similar manipulations show that

$$\begin{aligned} & \Pr_{D,\vec{\alpha}}(\varphi \mid S_k \cdot (v_i \approx b) > \Pr_{D,\vec{\alpha}}(\varphi \mid S_k) \\ \text{iff} \quad & \frac{o_i^2 x_1 + x_2}{o_i^2 y_1 + y_2} > \frac{o_i x_1 + x_2}{o_i y_1 + y_2} \\ \text{iff} \quad & x_2 y_1 + o_i x_1 y_2 > x_1 y_2 + o_i x_2 y_1. \end{aligned}$$

Thus, $\Pr_{D,\vec{\alpha}}(\varphi \mid S_k \cdot (v_i \approx b)) > \Pr_{D,\vec{\alpha}}(\varphi \mid S_k)$. A straightforward induction shows that $\Pr_{D,\vec{\alpha}}(\varphi \mid S_k \cdot (v_i \approx b)^h) > \Pr_{D,\vec{\alpha}}(\varphi \mid S_k)$ for all h, so $\Pr_{D,\vec{\alpha}}(\varphi \mid S^*) > \Pr_{D,\vec{\alpha}}(\varphi \mid S_k) = \Pr_{D,\vec{\alpha}}(\varphi \mid S)$, as desired. □

We conclude with a proof of the lemma that relates the ordering of probabilities to that of characteristic fractions.

Proof. (Lemma 10.2.5) Since, for $x, y > 0$, we have that $x > y$ iff $(1/x) < (1/y)$, it follows from Lemma 10.2.6 that $\Pr_{D,\vec{\alpha}}(\varphi \mid S) > \Pr_{D,\vec{\alpha}}(\varphi \mid S')$ iff

$$\frac{\sum_A r_{D,\vec{\alpha}}(A,S)}{\sum_{\{A:\,\varphi(A)=\mathrm{T}\}} r_{D,\vec{\alpha}}(A,S)} < \frac{\sum_A r_{D,\vec{\alpha}}(A,S'))}{\sum_{\{A:\,\varphi(A)=\mathrm{T}\}} r_{D,\vec{\alpha}}(A,S')\}},$$

which is true iff

$$< \frac{\frac{\sum_{\{A:\,\varphi(A)=\mathrm{T}\}} r_{D,\vec{\alpha}}(A,S)+\sum_{\{A:\,\varphi(A)=\mathrm{F}\}} r_{D,\vec{\alpha}}(A,S)}{\sum_{\{A:\,\varphi(A)=\mathrm{T}\}} r_{D,\vec{\alpha}}(A,S)}}{\frac{\sum_{\{A:\,\varphi(A)=\mathrm{T}\}} r_{D,\vec{\alpha}}(A,S')+\sum_{\{A:\,\varphi(A)=\mathrm{F}\}} r_{D,\vec{\alpha}}(A,S')}{\sum_{\{A:\,\varphi(A)=\mathrm{T}\}} r_{D,\vec{\alpha}}(A,S')}},$$

that is, if and only if

$$\frac{\sum_{\{A:\,\varphi(A)=\mathrm{F}\}} r_{D,\vec{\alpha}}(A,S)}{\sum_{\{A:\,\varphi(A)=\mathrm{T}\}} r_{D,\vec{\alpha}}(A,S)} < \frac{\sum_{\{A:\,\varphi(A)=\mathrm{F}\}} r_{D,\vec{\alpha}}(A,S')}{\sum_{\{A:\,\varphi(A)=\mathrm{T}\}} r_{D,\vec{\alpha}}(A,S')}.$$

The statement of the lemma follows. □

Consider the following example of the characteristic fraction of a formula.

Example 10.2.8. Let $\varphi = (v_1 \wedge v_2) \vee (\neg v_2 \wedge \neg v_3)$ and $S = (v_2 \approx F, v_1 \approx T)$, and suppose that the prior D is uniform and the testing accuracy is the same for all variables, so $o_1 = \ldots = o_n = o$. This formula has four satisfying assignments, namely $\{TTT, TFF, TTF, FFF\}$ (letting xyz denote the assignment $\{v_1 \mapsto x, v_2 \mapsto y, v_3 \mapsto z\}$, for brevity). The other four assignments, namely $\{FFT, TFT, FTT, FTF\}$, make the formula false. For each assignment A, the corresponding summand $r_{D,\vec{\alpha}}(A,S)$ is $\Pr_{D,\vec{\alpha}}(A)$ times a factor of o for every test outcome in S that is compatible with A, where a test outcome $v_i \approx b$ is *compatible*

with A if $b = A(v_i)$. For instance, the falsifying assignment FFT is compatible with $v_2 \approx F$ but not $v_1 \approx T$, so it gives rise to a summand of $\Pr_{D,\vec{\alpha}}(A) \cdot o$ in the numerator of the characteristic fraction of S. On the other hand, if A is the the satisfying assignment TFF, then both $v_1 \approx T$ and $v_2 \approx F$ are compatible with A, yielding $\Pr_{D,\vec{\alpha}}(A) \cdot o^2$ in the denominator. Performing the same analysis for the other assignments and cancelling the common factors of $\Pr_{D,\vec{\alpha}}(A)$ (as the prior is uniform), we find that

$$\mathrm{cf}(\varphi, S) = \frac{o^1 + o^2 + o^0 + o^0}{o^1 + o^2 + o^1 + o^1}.$$

For a more general example, suppose that $S = ((v_1 \approx T)^{c_1 k}, (v_2 \approx F)^{c_2 k}, (v_3 \approx F)^{c_3 k})$, where k is the total number of tests in S and real constants $0 \leq c_1, c_2, c_3 \leq 1$ with $c_1 + c_2 + c_3 = 1$ representing the fraction of times that each of the three test outcomes occurs in it. Then

$$\mathrm{cf}(\varphi, S) = \frac{o^{c_2 k} + o^{c_1 k + c_2 k} + o^0 + o^{c_3 k}}{o^{c_1 k} + o^{c_1 k + c_2 k + c_3 k} + o^{c_1 k + c_3 k} + o^{c_2 k + c_3 k}}.$$

□

In the second example above, the characteristic fraction of S depends only on the integers $c_1 k$, $c_2 k$, and $c_3 k$. The factor k is common to all the exponents, and so we can pull it out to rewrite $\mathrm{cf}(\varphi, S)$ as a fraction of sums of powers of o^k :

$$\mathrm{cf}(\varphi, S) = \frac{(o^k)^{c_2} + (o^k)^{c_1 + c_2} + (o^k)^0 + (o^k)^{c_3}}{(o^k)^{c_1} + (o^k)^{c_1 + c_2 + c_3} + (o^k)^{c_1 + c_3} + (o^k)^{c_2 + c_3}}.$$

They key point is that we can view $\mathrm{cf}(\varphi, S)$ as a function of the vector (c_1, c_2, c_3) that describes the relative makeup of the test-outcome sequence and a parameter o^k that depends on the test accuracy and the number of tests k. This can be shown to be true in general. Since we are interested in the behaviour of the

information-acquisition game as k, and hence o^k, goes to infinity, it will turn out to be useful to consider the asymptotic behaviour of cf as $o^k \to \infty$ as a function of (c_1, c_2, c_3). Indeed, the rest of our approach can be seen as the result of an attempt to make this idea rigorous. As a starting point, we define the relative-makeup vector for all test-outcome sequences.

Definition 10.2.9. Given a test-outcome sequence S and truth assignment A, the *A-trace* of S, denoted $\mathrm{Tr}_A(S)$, is the vector $\mathrm{Tr}_A(S) = (n^+_{S,A,1}/|S|, \ldots, n^+_{S,A,n}/|S|)$. □

Example 10.2.10. Consider the sequence of test outcomes

$$S_1 = (v_1 \approx T, v_2 \approx T, v_1 \approx T, v_1 \approx T, v_1 \approx F).$$

S_1 has three instances of $v_1 \approx T$, one instance of $v_1 \approx F$ and one instance of $v_2 \approx T$. So the $\{v_1 \mapsto T, v_2 \mapsto T\}$-trace of S_1 is $(\frac{3}{5}, \frac{1}{5})$; the $\{v_1 \mapsto F, v_2 \mapsto T\}$-trace of S_1 is $(\frac{1}{5}, \frac{1}{5})$. If the last test is $v_2 \approx F$ rather than $v_1 \approx F$, giving us the test-outcome sequence

$$S_2 = (v_1 \approx T, v_2 \approx T, v_1 \approx T, v_1 \approx T, v_2 \approx F),$$

then the $\{v_1 \mapsto T, v_2 \mapsto T\}$-trace of S_2 is also $(\frac{3}{5}, \frac{1}{5})$. The sequence

$$S_3 = (v_1 \approx T, v_2 \approx F, v_1 \approx T, v_1 \approx T, v_1 \approx T)$$

has 4 instances of $v_1 \approx T$ and 1 of $v_2 \approx F$, so the $\{v_1 \mapsto T, v_2 \mapsto F\}$-trace of S_3 is $(\frac{4}{5}, \frac{1}{5})$. □

It may seem that counting only the test outcomes that agree with A results in an unacceptable loss of information: indeed, as the example above illustrates, the A-traces of the two distinct test-outcome sequences S_1 and S_2 can be the same, even though S_1 and S_2 will in general lead to different posterior probabilities of a formula being true. However, it turns out that if a test-outcome

sequence is optimal, there must be some assignment A for which, in a sense, we do not lose information by taking the A-trace.

Definition 10.2.11. The test-outcome sequence S and the assignment A are *compatible* if all test outcomes in S are compatible with A: that is, S contains no observations of the form $v_i \approx \neg A(v_i)$. □

Lemma 10.2.12. *Every optimal test-outcome sequence S is compatible with some assignment A.*

Proof. Immediate from Lemma 10.2.7. □

We can now define a counterpart to the earlier definition of a characteristic fraction that uses only the information that is given by an A-trace.

Definition 10.2.13. If $\vec{c} = (c_1, \ldots, c_n)$, φ is a formula in the n variables $v_1, \ldots, v_n$, and A is a truth assignment, then the *characteristic fraction of the A-trace* is the function cf_A, where

$$\mathrm{cf}_A(\varphi, \vec{c}, k) \quad = \quad \frac{\sum_{\{B:\varphi(B)=\mathrm{F}\}} \mathrm{Pr}_{D,\vec{\alpha}}(B) \prod_{\{v_i:A(v_i)=B(v_i)\}} o_i^{c_i k}}{\sum_{\{B:\varphi(B)=\mathrm{T}\}} \mathrm{Pr}_{D,\vec{\alpha}}(B) \prod_{\{v_i:A(v_i)=B(v_i)\}} o_i^{c_i k}}.$$

□

As expected, the quantities $\mathrm{cf}(\varphi, S)$ and $\mathrm{cf}_A(\varphi, \vec{c}, k)$ are closely related. The following lemma makes precise when we do not lose information by considering the appropriate A-trace.

Lemma 10.2.14. *For all truth assignments A compatible with S, we have*

$$\mathrm{cf}(\varphi, S) = \mathrm{cf}_A(\varphi, \mathrm{Tr}_A(S), |S|).$$

Proof. If A is compatible with S, then $(\mathrm{Tr}_A(S))_i = n^+_{S,A,i}/|S|$ for all i, so the result is immediate from the definition. □

Recall that our goal is to show that the optimal test-outcome sequences for φ, that is, sequences S that maximise $|\Pr_{D,\vec{\alpha}}(\varphi \mid S) - \frac{1}{2}|$, satisfy a particular property. By Lemma 10.2.5, the optimal test-outcome sequences either minimise $\mathrm{cf}(\varphi, S)$ or $\mathrm{cf}(\neg\varphi, S) = 1/\mathrm{cf}(\varphi, S)$. By Lemma 10.2.12, there must be some assignment A that S is compatible with; by Lemma 10.2.14, we can equivalently minimise $\mathrm{cf}_A(\varphi, S)$ or $\mathrm{cf}_A(\neg\varphi, S)$ for a truth assignment A compatible with S. In Section 10.2.2 we show that if S is sufficiently long and compatible with A and $\varphi(A) = T$, then we must have $\Pr_{D,\vec{\alpha}}(\varphi|S) \geq \Pr_{D,\vec{\alpha}}(\neg\varphi|S)$, while if $\varphi(A) = F$, the opposite inequality must hold (see Lemma 10.2.6). So we need to minimise $\mathrm{cf}_A(\varphi, S)$ if $\varphi(A) = T$ and to minimise $\mathrm{cf}_A(\neg\varphi, S)$ if $\varphi(A) = F$. It suffices to find a sequence S and a truth assignment A that is compatible with S for which the appropriate cf_A is minimal.

10.2.2 The polytope of optimal A-traces

What does this view actually gain us over the naive undertaking to identify the optimal test-outcome sequences directly? Recall that our definition of rational inattention depends on the asymptotic behaviour of information-acquisition games as the number k of tests goes to infinity. Even without the exponential dependency of the search space on k, it is not clear how to analyze large games.

Here, the machinery of A-traces, which do not depend on k at all, comes in helpful. As part of the statement of Theorem 10.2.20, we provide an A-trace analogue (that is thus independent of k) of the criterion that Proposition 10.2.2 gives

for rational inattention in test-outcome sequences. In the course of the proof of this theorem, we will see that, with certain caveats, we can say enough about the A-traces of all optimal test-outcome sequences to be able to check whether they satisfy the analogous criterion. Roughly speaking, the space of A-traces represents something like a continuous generalisation of (a subspace of) test-outcome sequences. We show that inside this space, there is a convex polytope of what can be viewed as "optimal A-traces" (cf. Lemma 10.2.25). As a convex polytope, it is amenable to well-known and well-behaved optimisation techniques. Moreover, as the number k of tests grows, the A-traces of actual optimal test-outcome sequences get progressively closer to points in this polytope. It follows that if all the points in the polytope of "optimal A-traces" satisfy the criterion for rational inattention, then all the A-traces of actual optimal test-outcome sequences are a negligible function away from satisfying them.

We present the definition of this convex polytope and the formal statement of the A-trace analogue criterion here, but relegate the technical details of the rest of the proof to Section 10.2.2.

Assumption: We assume for ease of exposition for the remainder of the thesis that the measurement accuracy of each variable is the same, that is, $\alpha_1 = \cdots = \alpha_n$. This implies that $o_1 = \cdots = o_n$; we use o to denote this common value. While we do not need this assumption for our results, allowing non-uniform measurement vectors $\vec{\alpha}$ would require us to parameterize RI by the measurement accuracy; the formulae that exhibit $(0.1, 0.1)$-RI might not be the same as those that exhibit $(0.1, 0.3)$-RI.

With this assumption, we can show that $\mathrm{cf}_A(\varphi, S)$ is essentially characterised by the terms in its numerator and denominator with the largest exponents. Ev-

ery optimal test-outcome sequence S is compatible with some assignment A. Since all test outcomes in S are compatible with A, if $\varphi(A) = T$, the summand due to A in the denominator of $\mathrm{cf}(\varphi, S) = \mathrm{cf}_A(\varphi, \mathrm{Tr}_A(S), |S|)$ is of the form $\Pr_{D,\vec{\alpha}}(A)o^{|S|}$. This term must be the highest power of o that occurs in the denominator. The highest power of o in the numerator of $\mathrm{cf}_A(S)$, which is due to some assignment B for which $\varphi(B) = F$, will in general be smaller than $1 \cdot |S|$, and depends on the structure of φ. On the other hand, if $\varphi(A) = F$, we want to minimise the characteristic fraction for $\neg\varphi$, for which the sets of satisfying and falsifying assignments are the opposite of those with A. So, in either case, the greatest power in the numerator of the characteristic fraction we care about will be due to an assignment B for which $\varphi(B) \neq \varphi(A)$. As Lemma 10.2.16 below shows, we can formalise the appropriate highest power as follows:

Definition 10.2.15. The *max-power* of a vector $\vec{c} \in \mathbb{R}^n$ is

$$\mathrm{maxp}_{\varphi,A}(\vec{c}) = \max_{\{B:\varphi(B)\neq\varphi(A)\}} \sum_{\{i:A(v_i)=B(v_i)\}} c_i.$$

□

Lemma 10.2.16. *If S is a test-outcome sequence compatible with A and $\varphi(A) = T$ (resp., $\varphi(A) = F$), then the highest power of o that occurs in the numerator of* $\mathrm{cf}(\varphi, S)$ *(resp.,* $\mathrm{cf}(\neg\varphi, S)$ *) is* $|S|\mathrm{maxp}_{\varphi,A}(\mathrm{Tr}_A(S))$.

Proof. This follows from the definition of $\mathrm{cf}_A(\varphi, \mathrm{Tr}_A(S), |S|)$, the observation that all entries in $\mathrm{Tr}_A(S)$ are non-negative, and Lemma 10.2.14. □

We now show that the search for the max-power can be formulated as a linear program (LP). Note that if R is a compact subset of $\mathbb{R}$, finding a maximal element of the set is equivalent to to finding a minimal upper bound for it:

$$\max R = \min\{m \mid \forall r \in R : r \leq m\}.$$

Hence, finding the vector $\vec{c}$ with $\sum_i c_i = 1$ and $c_i \geq 0$ that attains the greatest max-power, that is, that maximises $\max_{\{B:\varphi(B)\neq\varphi(A)\}}(\sum_{\{i:A(v_i)=B(v_i)\}} c_i)$ is equivalent to finding the $\vec{c}$ and max-power m that minimise m subject to $\max_{\{B:\varphi(B)\neq\varphi(A)\}} \sum_{\{i:A(v_i)=B(v_i)\}} c_i \leq m$, $\sum_i c_i = 1$, and $c_i \geq 0$ for all i. These latter constraints are captured by the following LP.

Definition 10.2.17. Given a formula φ and truth assignment A, define the *conflict LP* $L_A(\varphi)$ to be the linear program

$$\begin{aligned} &\text{minimise } m \\ &\text{subject to } \sum_{\{i:A(v_i)=B(v_i)\}} c_i \leq m \text{ for all } B \text{ such that } \varphi(B) \neq \varphi(A) \\ &\sum_i c_i = 1 \\ &c_i \geq 0 \text{ for } i = 1, \ldots, n \\ &0 \leq m \leq 1. \end{aligned}$$

□

The constraint $0 \leq m \leq 1$ is not necessary; since the c_i's are non-negative and $\sum_i c_i = 1$, the minimum m that satisfies the constraints must be between 0 and 1. However, adding this constraint ensures that the set of tuples $(c_1, \ldots, c_n, m)$ that satisfy the constraints form a compact (i.e., closed and bounded) set. It is almost immediate from the definitions that the solution to $L_A(\varphi)$ is $\sup_{\vec{c}:\sum_{i=1}^n c_i=1,\ c_i\geq 0} \mathrm{maxp}_{\varphi,A}(\vec{c})$.

We call $L_A(\varphi)$ a *conflict LP* because we are considering assignments B that *conflict* with A, in the sense that φ takes a different truth value on them than it does on A. To reason about conflict LPs, we first introduce some notation.

Definition 10.2.18. Suppose that L is a linear program in n variables minimising an objective function $f : \mathbb{R}^n \to \mathbb{R}$ subject to some constraints.

- The *feasible set* of L, $Feas(L) \subseteq \mathbb{R}^n$, is the set of points that satisfy all the constraints of L.
- The *minimum* of the LP, $\mathrm{MIN}(L)$, is the infimum $\inf_{p \in Feas(L)} f(p)$ attained by the objective function over the points in $Feas(L)$.
- The *solution polytope* of L, $\mathrm{OPT}(L) \subseteq Feas(L) \subseteq \mathbb{R}^n$, is the set of all feasible points at which f attains the minimum, that is, $\mathrm{OPT}(L) = \{p \in Feas(L) : f(p) = \mathrm{MIN}(L)\}$.

□

It now follows that if $(\vec{d}, m) \in \mathrm{OPT}(L_A(\varphi))$, then $\mathrm{maxp}_{\varphi,A}(\vec{d}) = m = \mathrm{MIN}(L_A(\varphi))$. Our goal is to show that the solutions to the conflict LPs tell us enough about the structure of optimal test-outcome sequences to derive a sufficient condition for a formula to exhibit RI.

Roughly speaking, $\mathrm{MIN}(L_A(\varphi))$ tells us how well any sequence of test outcomes compatible with the assignment A can do. Since every optimal sequence is compatible with some assignment, we therefore can find the max-power of optimal sequences by considering the minimum of the minima of all LPs:

Definition 10.2.19. For a formula φ, the *minimax power* $\mathrm{MIN}^*(\varphi)$ is the minimum of minima:

$$\mathrm{MIN}^*(\varphi) = \min_{\text{assignments } A} \mathrm{MIN}(L_A(\varphi)).$$

An assignment A, and the LP $L_A(\varphi)$, are *relevant* if $\mathrm{MIN}(L_A(\varphi)) = \mathrm{MIN}^*(\varphi)$. □

With this definition, we can finally state the core theorem of this section.

Theorem 10.2.20. *If there exists a constant $C > 0$ such that for all relevant truth assignments A and all solution points $\vec{c} = (c_1, \ldots, c_n, m) \in \mathrm{OPT}(L_A(\varphi))$, there exist indices i and j such that $v_i \leq_\varphi v_j$, $c_i \geq C$, and $c_j = 0$, then φ exhibits RI.*

As the conflict LP is of polynomial size, we can solve for a point in the solution polytope in polynomial time. Some additional subtleties are involved in making sure that the criteria we imposed on the coordinates, which represent the A-trace counterpart of the criteria of Proposition 10.2.2 for optimal test-outcome sequences, are satisfied for all such points. In Section 10.2.4, we will see that we can do this by solving a polynomial number of LPs derived from the conflict LP. Thus, we obtain a polynomial-time (in 2^n, where n is the number of variables) algorithm for evaluating a sufficient criterion for a formula to exhibit rational inattention.

10.2.3 Proof of Theorem 10.2.20

To prove Theorem 10.2.20, we show that the antecedent of the theorem implies the antecedent of Proposition 10.2.2. The next lemma is a first step towards this goal. Proposition 10.2.2 involves a condition on test sequences that intuitively says that some variable is tested often, but another variable that is at least as important is tested very little. This condition arises repeatedly in the following proof, so we attach a name to it.

Definition 10.2.21. Given a constant c and negligible function f, a test-outcome sequence S is (f, c, φ)-*good* if there exist variables v_i and v_j such that $v_i \geq_\varphi v_j$,

S contains at least $c|S|$ tests of v_j, and S contains at most $f(|S|)$ tests of v_i. S is (f, c, φ)-*bad* if it is not (f, c, φ)-good. □

Using this notation, Proposition 10.2.2 says that a formula φ exhibits RI if for all open-minded product distributions D and accuracy vectors $\vec{\alpha}$, there exists a negligible function f and $c > 0$ such that all test-outcome sequences optimal for φ, D, and $\vec{\alpha}$ are (f, c, φ)-good. The contrapositive of Proposition 10.2.2 says that if a formula does not exhibit RI, then for all f and c, there is an (f, c, φ)-bad test-outcome sequence optimal for φ, D, and $\vec{\alpha}$. Bad test-outcome sequences are counterexamples to RI. The next lemma allows us to "boost" such counterexamples if they exist: whenever we have a single bad test-outcome sequence, we in fact have an infinite family of arbitarily long bad test-outcome sequences that can be considered refinements of the same counterexample.

Lemma 10.2.22. *If, for all negligible functions f and constants $c > 0$, there exists an (f, c, φ)-bad test-outcome sequence that is optimal for φ, D, and $\vec{\alpha}$, then for all f and c, there exists an infinite sequence $\{S_k\}$ of (f, c, φ)-bad optimal test-outcome sequences of increasing length (so that $|S_{k+1}| > |S_k|$), all optimal for φ, D, and $\vec{\alpha}$.*

Proof. We show the contrapositive. Fix φ, D, and $\vec{\alpha}$. We show that if there exist f and c for which there is no infinite sequence $\{S_k\}$ of (f, c, φ)-bad test-outcome sequences optimal for φ, D, and $\vec{\alpha}$, then, for all D and $\vec{\alpha}$, there exist f'' and c'' for which there is not even a single (f'', c'')-bad test-outcome sequence that is optimal for φ, D, and $\vec{\alpha}$.

Choose f and c such that the premises of the contrapositive hold. Let $\mathcal{S}_{f,c}$ be the set of all (f, c, φ)-bad test-outcome sequences that are optimal for φ, D and $\vec{\alpha}$. We can assume $\mathcal{S}_{f,c}$ is nonempty; otherwise the claim trivially holds. If there

exist arbitrarily long sequences $S \in \mathcal{S}_{f,c}$, then we can pick a sequence $\{S_k\}$ of test-outcome sequences in $\mathcal{S}_{f,c}$ of increasing length from them, contradicting the assumption. In fact, this must be the case. For suppose, by way of contradiction, that it isn't. Then there must be an upper bound $\hat{k}$ on the lengths of test-outcome sequences in $\mathcal{S}_{f,c}$. Moreover, since there are only finitely many test-outcome sequences of a given length, $\mathcal{S}_{f,c}$ itself must also be finite. Thus,

$$c' = \min_{S \in \mathcal{S}_{f,c}} \max_{\text{variables } v_i \text{ in } \varphi} |(\mathrm{Tr}_A(S))_i|$$

is finite and greater than zero (as every sequence must test at least one variable and not contradict itself, so we are taking the minimum over finitely many terms greater than zero). Hence, $c'' = \min\{c, c'\}$ is also greater than 0. Let

$$f''(k) = \begin{cases} k & \text{if } k \leq \hat{k} \\ f(k) & \text{otherwise.} \end{cases}$$

Since f is negligible and f'' agrees with f for all $k > \hat{k}$, f'' is also negligible.

We claim that no test-outcome sequence S optimal for φ, D, and $\vec{\alpha}$ is (f'', c'')-bad. Indeed, all candidate sequences of length $|S| \leq \hat{k}$ are ruled out, because setting both v_i and v_j to be whatever variable is tested the most in S discharges the existential quantification of Definition 10.2.21 (note $\leq_\varphi$ is a partial order, so $v_i \leq_\varphi v_i$ for all v_i) as the number of tests is bounded below by the minimum $c'|S|$ and above by the length $|S|$. Any test-outcome sequence S of length $|S| > \hat{k}$ must also be (f'', c'')-good. Indeed, by choice of $\hat{k}$, S is (f, c, φ)-good. Therefore, there must be a variable pair $v_i \geq_\varphi v_j$ such that S contains $\geq c|S|$ tests of v_j and $\leq f(|S|)$ tests of v_i. But $c'' \leq c$ by definition and $f''(|S|) = f(|S|)$, so v_i and v_j also bear witness to S being (f'', c'')-good. This gives the desired contradiction.

Thus, we have shown that there exists a sequence $\{S_k\}$ of bad test-sequence outcomes in $\mathcal{S}_{f,c}$ of increasing length. □

In the following, we use the standard notion of *1-norm*, where the 1-norm of a real-valued vector $\vec{v} = (v_1, \ldots, v_n)$ is

$$\|\vec{v}\|_1 = \sum_{i=1}^{n} |\vec{v}_i|,$$

the sum of absolute values of the entries of $\vec{v}$. We often consider the 1-norm of the difference of two vectors. Although the difference of vectors is defined only if they have same length, we occasionally abuse notation and write $\|\vec{v} - \vec{w}\|_1$ even when $\vec{v}$ and $\vec{w}$ are vectors of different lengths. In that case, we consider only the common components of the vectors. For example, if $\vec{v} = (v_1, \ldots, v_n)$ and $\vec{w} = (w_1, \ldots, w_m)$, then

$$\|\vec{v} - \vec{w}\|_1 = |v_1 - w_1| + \cdots + |v_{\min\{n,m\}} - w_{\min\{n,m\}}|.$$

We further facilitate working with different-length vectors by using $(\vec{v}, \vec{w})$ to denote vector concatenation, so $(\vec{v}, \vec{w})$ denotes the vector $(v_1, \ldots, v_n, w_1, \ldots, w_m)$.

The following fact about LPs will prove useful.

Lemma 10.2.23. *If L is an LP with objective function f such that $Feas(L)$ is compact, then for all $\varepsilon > 0$, there exists an $\varepsilon' > 0$ such that all feasible points $\vec{p} \in Feas(L)$, either $\vec{p}$ is within ε of a solution point, that is,*

$$\exists \vec{o} \in \mathrm{OPT}(L)(\|\vec{p} - \vec{o}\|_1 < \varepsilon),$$

or $f(\vec{p})$ is more than ε' away from the optimum, that is,

$$f(\vec{p}) - \mathrm{MIN}(L) > \varepsilon'.$$

Proof. We will argue by contradiction. Suppose that the claim does not hold, and let Q be the set of all points in $Feas(L)$ that do not satisfy the first inequality; that is,

$$Q = \{\vec{p} \in Feas(L) : \forall \vec{o} \in \mathrm{OPT}(L)(\|\vec{p} - \vec{o}\|_1 \geq \varepsilon)\}.$$

This set is bounded and closed, hence compact. If $\inf_{\vec{q} \in Q}(f(\vec{q}) - \text{MIN}(L)) > 0$, then we can take $\epsilon' = \inf_{\vec{q} \in Q}(f(\vec{q}) - \text{MIN}(L))/2$ since then, for every point $\vec{p} \in \mathit{Feas}(L)$, if $f(\vec{p}) - \text{MIN}(L) \leq \epsilon'$, then $\vec{p} \notin Q$, and hence by definition of Q, $\vec{p}$ must be within ϵ of some solution point.

So suppose that $\inf_{\vec{q} \in Q}(f(\vec{q}) - \text{MIN}(L)) = 0$. Then there exists a sequence $(\vec{q}_i)_{i=1}^{\infty}$ of points in Q such that $\lim_{i \to \infty} f(\vec{q}) = \text{MIN}(L)$. By the Bolzano-Weierstrass Theorem, this sequence must have a convergent subsequence $(\vec{q}'_i)_{i=1}^{\infty}$. Write $\vec{q}^*$ for $\lim_{i \to \infty} \vec{q}'_i$. This limit point is still in Q, as Q is compact. Since f is linear, hence continuous,

$$\begin{aligned} f(\vec{q}^*) &= f(\lim_{i \to \infty} \vec{q}'_i) \\ &= \lim_{i \to \infty} f(\vec{q}'_i) \\ &= \lim_{i \to \infty} f(\vec{q}_i) \\ &= \text{MIN}(L). \end{aligned}$$

Thus, $\vec{q}^* \in \text{OPT}(L)$ and $\vec{q}^* \in Q$, which is incompatible with the definition of Q. This gives the desired contradiction. □

We have seen how to distill the information in a test-outcome sequence for a formula in n variables into a vector in $\mathbb{R}^n$ by taking A-traces. The following lemma is to be understood as an approximate converse of this process: given a vector in $\mathbb{R}^n$, we construct a test-outcome sequence of a given length k whose A-trace is close (within an error term of $2n/k$) to that vector.

Lemma 10.2.24. *If A is an assignment to the n variables of φ and $\vec{d} \in \mathbb{R}^n$ is such that all coordinates are non-negative and sum to 1, then for all $k \in \mathbb{N}$, there exists a test-outcome sequence $S_{k,\vec{d},A}$ of length k compatible with A such that $|\text{maxp}_{\varphi,A}(\text{Tr}_A(S_{k,\vec{d},A}) - \text{maxp}_{\varphi,A}(\vec{d})| < 2n/k$.*

Proof. Define

$$S_{k,\vec{d},A} = ((v_1 \approx A(v_1))^{\lfloor d_1 k \rfloor}, \ldots, (v_n \approx A(v_n))^{\lfloor d_n k \rfloor}, (v_n \approx A(v_n))^{e}),$$

where $\lfloor x \rfloor$ is the floor of x (i.e., the largest integer n such that $n \leq x$) and $e = k - (\sum_{i-1}^{n} \lfloor d_i k \rfloor)$ is whatever is needed to pad the sequence to having length k. (e.g., if $\vec{d} = (0.3, 0.7)$ and $k = 2$, then although the d_is sum to 1, $\lfloor d_1 k \rfloor = 0$ and $\lfloor d_2 k \rfloor = 1$, so we would have $e = 1$.)

Since $\sum_i d_i k = k$, $\sum_i \lfloor d_i k \rfloor \leq k$, and hence $e \geq 0$. Also, $\mathrm{Tr}_A(S_{k,\vec{d},A})$ differs from $\vec{d}$ by at most $1/k$ in the first $n-1$ coordinates (as $|d_1 k - \lfloor d_1 k \rfloor| \leq 1$) and by at most n/k in the final coordinate (as $e \leq n$). Hence, for each assignment B,

$$\left| \sum_{\{i:A(v_i)=B(v_i)\}} d_i - \sum_{\{i:A(v_i)=B(v_i)\}} (Tr_A(S_{k,\vec{d},A}))_i \right| \leq (n-1)\frac{1}{k} + \frac{n}{k} < \frac{2n}{k}.$$

Recalling the definition of the max-power for a vector $\vec{c}$,

$$\mathrm{maxp}_{\varphi,A}(\vec{c}) = \max_{\{B:\varphi(B)\neq\varphi(A)\}} \sum_{\{i:A(v_i)=B(v_i)\}} c_i,$$

it follows that $|\mathrm{maxp}_{\varphi,A}(\mathrm{Tr}_A(S_{k,\vec{d},A)}) - \mathrm{maxp}_{\varphi,A}(\vec{d})| < 2n/k$, as desired. □

We can finally relate the solutions of the conflict LP $L_A(\varphi)$ to the traces of optimal test-outcome sequences. While the traces of optimal sequences may not be in $\mathrm{OPT}(L_A(\varphi))$, they must get arbitrarily close to it as the length of the sequence gets larger.

Lemma 10.2.25. *If D is open-minded, then there exists a function $\delta : \mathbb{N} \to \mathbb{R}$, depending only on φ, D, and $\vec{\alpha}$, such that*

- $\lim_{k\to\infty} \delta(k) = 0$ *and*

- *for all assignments A and test-outcome sequences S compatible with A that are optimal for φ, D, and $\vec{\alpha}$, the A-trace of S is within $\delta(|S|)$ of some solution $(\vec{d}, m) \in \mathrm{OPT}(L_A(\varphi))$, that is,*

$$\exists(\vec{d}, m) \in OPT(L_A(\varphi)). \|\vec{d} - \mathrm{Tr}_A(S)\|_1 < \delta(|S|).$$

Proof. Fix φ, D, and $\vec{\alpha}$. Given $\varepsilon > 0$, we show that there exists a constant k_ε such that for all truth assignments A and all test-outcome sequences S compatible with A such that $|S| > k_\epsilon$ and

$$\forall(\vec{d}, m) \in \mathrm{OPT}(L_A(\varphi)). \|\mathrm{Tr}_A(S) - \vec{d}\|_1 \geq \varepsilon, \tag{10.4}$$

S is not optimal for φ, D, and $\vec{\alpha}$. This suffices to prove the result, since we can then choose any descending sequence $\varepsilon_0, \varepsilon_1, \ldots$ and define $\delta(n) = \varepsilon_n$ for all $k_{\varepsilon_n} < n \leq k_{\varepsilon_{n+1}}$.

Fix $\varepsilon > 0$ and A. Choose an arbitrary test-outcome sequence S compatible with A satisfying (10.4). Without loss of generality, we can assume that $\varphi(A) = T$. (If $\varphi(A) = F$, then the lemma follows from applying the argument below to $\neg\varphi$ and the observation that sequences are optimal for φ iff they are optimal for $\neg\varphi$.) The feasible set of the LP $L_A(\varphi)$ is compact by construction. Therefore, we can invoke Lemma 10.2.23 to obtain a constant $\varepsilon_A > 0$, depending on ε and the LP (and hence A), such that for all feasible points $p = (\vec{c}, m) \in \mathit{Feas}(L_A(\varphi))$, either $\|\vec{c} - \vec{d}\|_1 < \epsilon$ for some $\vec{d} \in \mathrm{OPT}(L)$, or $|m - \mathrm{MIN}(L_A(\varphi))| > \epsilon_A$. Set

$$k_{\epsilon,A} = \max\left(\frac{4n}{\epsilon_A}, \frac{2}{\varepsilon_A}\log_o\left(\frac{2^{2n}}{\mathrm{Pr}_{D,\vec{\alpha}}(A)\min_B \mathrm{Pr}_{D,\vec{\alpha}}(B)}\right)\right).$$

(Since D is open-minded, $\min_B \mathrm{Pr}_{D,\vec{\alpha}}(B) > 0$, so this is well defined.)

We now show that if $|S| > k_{\epsilon,A}$, then S is not optimal for φ, D, and $\vec{\alpha}$. We can then take $k_\epsilon = \max_A k_{\epsilon,A}$ to complete the proof. By assumption, S satisfies (10.4).

Since appending another entry to a vector can only make its 1-norm greater, we also have $\|(\mathrm{Tr}_A(S), \mathrm{maxp}_{\varphi,A}(\mathrm{Tr}_A(S))) - \vec{d}\|_1 \geq \varepsilon$ for all $\vec{d} \in \mathrm{OPT}(L_A(\varphi))$. The point $(\mathrm{Tr}_A(S), \mathrm{maxp}_{\varphi,A}(\mathrm{Tr}_A(S)))$ is in the feasible set of $L_A(\varphi)$; this contradicts the first option in the disjunction provided by Lemma 10.2.23. Therefore, the second option must be true:

$$|\mathrm{maxp}_{\varphi,A}(\mathrm{Tr}_A(S)) - \mathrm{MIN}(L_A(\varphi))| \quad > \quad \varepsilon_A. \tag{10.5}$$

Since all the entries in $\mathrm{Tr}_A(S)$ are non-negative, it follows from the definition that

$$\begin{aligned} & \mathrm{cf}_A(\varphi, \mathrm{Tr}_A(S), |S|) \\ = \; & \frac{\sum_{\{B:\varphi(B)=\mathrm{F}\}} \Pr_{D,\vec{\alpha}}(B) o^{\sum_{\{v_i:A(v_i)=B(v_i)\}} \mathrm{Tr}_A(S)_i |S|}}{\sum_{\{B:\varphi(B)=\mathrm{T}\}} \Pr_{D,\vec{\alpha}}(B) o^{\sum_{\{v_i:A(v_i)=B(v_i)\}} \mathrm{Tr}_A(S)_i |S|}} \\ \geq \; & \frac{\min_B \Pr_{D,\vec{\alpha}}(B) o^{\mathrm{maxp}_{\varphi,A}(\mathrm{Tr}_A(S))|S|}}{2^n o^{|S|}} \qquad \text{[see below]}. \end{aligned} \tag{10.6}$$

The inequality holds because, as we observed before, the term in the numerator with the greatest exponent has exponent $\mathrm{maxp}_{\varphi,A}(\mathrm{Tr}_A(S))|S|$. Its coefficient is at least $\min_B \Pr_{D,\vec{\alpha}}(B)$. The remaining terms in the numerator (if any) are non-negative. Thus, the numerator is at least as large as $\min_B \Pr_{D,\vec{\alpha}}(B) o^{\mathrm{maxp}_{\varphi,A}(\mathrm{Tr}_A(S))|S|}$. There are 2^n terms in the denominator, each of which is at most $o^{|S|}$, since, as we observed earlier, $\sum_i \mathrm{Tr}_A(S)_i = 1$ (since S is compatible with A). Thus, the denominator is at most $2^n o^{|S|}$.

Fix $(\vec{d}, m) \in \mathrm{OPT}(L_A(\varphi))$. By Lemma 10.2.24, there exists a test-outcome sequence $S_{|S|,\vec{d},A}$ such that $|\mathrm{maxp}_{\varphi,A}(\mathrm{Tr}_A(S_{|S|,\vec{d},A})) - \mathrm{maxp}_{\varphi,A}(\vec{d})| < 2n/|S|$. For brevity, set $\vec{d}' = \mathrm{Tr}_A(S_{|S|,\vec{d},A})$. So if $|S| > k_{\epsilon,A} \geq 4n/\epsilon_A$, then $|\mathrm{maxp}_{\varphi,A}(\vec{d}') - \mathrm{maxp}_{\varphi,A}(\vec{d})| < \epsilon_A/2$. Since $(\vec{d}, m) \in \mathrm{OPT}(L_A(\varphi))$, we have that $\mathrm{maxp}_{\varphi,A}(\vec{d}) = m = \mathrm{MIN}(L_A(\varphi))$, so $|\mathrm{maxp}_{\varphi,A}(\vec{d}') - \mathrm{MIN}(L_A(\varphi))| < \varepsilon_A/2$. Now using (10.5) and applying the triangle inequality gives us that

$$\mathrm{maxp}_{\varphi,A}(\mathrm{Tr}_A(S)) - \mathrm{maxp}_{\varphi,A}(\vec{d}') > \varepsilon_A/2. \tag{10.7}$$

Much as above, we can show that

$$\begin{aligned} & \mathrm{cf}_A(\varphi, \vec{d}', |S|) \\ = \; & \frac{\sum_{\{B:\varphi(B)=\mathrm{F}\}} \Pr_{D,\vec{\alpha}}(B) o^{\sum_{\{v_i:A(v_i)=B(v_i)\}} d'_i |S|}}{\sum_{\{B:\varphi(B)=\mathrm{T}\}} \Pr_{D,\vec{\alpha}}(B) o^{\sum_{\{v_i:A(v_i)=B(v_i)\}} d'_i |S|}} \\ \leq \; & \frac{2^n o^{\mathrm{maxp}_{\varphi,A}(\vec{d}')|S|}}{\Pr_{D,\vec{\alpha}}(A) o^{|S|}}, \end{aligned} \tag{10.8}$$

where now the inequality follows because we have replaced every term $\Pr_{D,\vec{\alpha}}(B)$ in the numerator by 1 and there are at most 2^n of them, and the fact that $\Pr_{D,\vec{\alpha}}(A) o^{|S|}$ is one of the terms in the denominator and the rest are non-negative.

Now observe that

$$\begin{aligned} & \Pr_{D,\vec{\alpha}}(\varphi \mid S_{|S|,\vec{d},A}) > \Pr_{D,\vec{\alpha}}(\varphi \mid S) \\ \text{iff} \quad & \mathrm{cf}(\varphi, S_{|S|,\vec{d},A}) < \mathrm{cf}(\varphi, S) && \text{[by Lemma 10.2.5]} \\ \text{iff} \quad & \mathrm{cf}_A(\varphi, \vec{d}', |S|) < \mathrm{cf}_A(\varphi, \mathrm{Tr}_A(S), |S|) && \text{[by Lemma 10.2.14]} \\ \text{if} \quad & \frac{2^n o^{\mathrm{maxp}_{\varphi,A}(\vec{d}')|S|}}{\Pr_{D,\vec{\alpha}}(A) o^{|S|}} < \frac{\min_B \Pr_{D,\vec{\alpha}}(B) o^{\mathrm{maxp}_{\varphi,A}(\mathrm{Tr}_A(S))|S|}}{2^n o^{|S|}} && \text{[by (10.6) and (10.8)]} \\ \text{iff} \quad & \frac{\min_B \Pr_{D,\vec{\alpha}}(B) o^{\mathrm{maxp}_{\varphi,A}(\mathrm{Tr}_A(S))|S|}}{2^n o^{|S|}} - \frac{2^n o^{\mathrm{maxp}_{\varphi,A}(\vec{d}')|S|}}{\Pr_{D,\vec{\alpha}}(A) o^{|S|}} > 0 \\ \text{iff} \quad & \frac{\Pr_{D,\vec{\alpha}}(A) \min_B \Pr_{D,\vec{\alpha}}(B) o^{\mathrm{maxp}_{\varphi,A}(\mathrm{Tr}_A(S))|S|} - 2^{2n} o^{\mathrm{maxp}_{\varphi,A}(\vec{d}')|S|}}{\Pr_{D,\vec{\alpha}}(A) 2^n o^{|S|}} > 0 \\ \text{iff} \quad & \Pr_{D,\vec{\alpha}}(A) \min_B \Pr_{D,\vec{\alpha}}(B) o^{\mathrm{maxp}_{\varphi,A}(\mathrm{Tr}_A(S))|S| - \mathrm{maxp}_{\varphi,A}(\vec{d}')|S|)} - 2^{2n} > 0 \\ \text{iff} \quad & (\mathrm{maxp}_{\varphi,A}(\mathrm{Tr}_A(S)) - \mathrm{maxp}_{\varphi,A}(\vec{d}'))|S| > \log_o \left(\frac{2^{2n}}{\Pr_{D,\vec{\alpha}}(A) \min_B \Pr_{D,\vec{\alpha}}(B)} \right). \end{aligned} \tag{10.9}$$

By assumption, $|S| > \frac{2}{\varepsilon_A} \log_o \frac{2^n 2^n}{\Pr_{D,\vec{\alpha}}(A) \min_B \Pr_{D,\vec{\alpha}}(B)}$; by (10.7), $(\mathrm{maxp}_{\varphi,A}(\mathrm{Tr}_A(S)) - \mathrm{maxp}_{\varphi,A}(\vec{d}')) > \varepsilon_A/2$. It follows that the last line of (10.9) is in fact satisfied. Thus S is not optimal, as desired. □

Moreover, unless the sequence in question is short, any optimal sequence of test outcomes must be compatible with an LP that actually attains the minimax power.

Lemma 10.2.26. *There exists a constant k_0, depending only on φ, D and $\vec{\alpha}$, such that if a sequence S of length $|S| \geq k_0$ is compatible with an assignment A, then either S is*

not optimal or A is relevant.

Proof. The proof reuses many of the core ideas of Lemma 10.2.25 in a simplified setting. For contradiction, suppose that A is not relevant, but S is optimal. Let B be an arbitrary relevant assignment. Then

$$\mathrm{MIN}(L_A) - \mathrm{MIN}(L_B) = \epsilon > 0.$$

We show that we can choose a k_0 such that if $|S| > k_0$, then there is a test-outcome sequence S' of the same length supporting B that is actually better, contradicting the optimality of S.

Indeed, set

$$k_0 = \max\left\{4n/\epsilon, \frac{2}{\epsilon}\log_o\left(\frac{2^{2n}}{\Pr_{D,\vec{\alpha}}(B)\min_C \Pr_{D,\vec{\alpha}}(C)}\right)\right\}.$$

Since $\mathrm{Tr}_A(S)$ is a feasible point of L_A, we have $\mathrm{maxp}_{\varphi,A}\mathrm{Tr}_A(S) \geq \mathrm{MIN}(L_A) \geq \mathrm{MIN}(L_B) + \epsilon$. On the other hand, let $(\vec{d}, m) \in \mathrm{OPT}(L_B(\varphi))$ be arbitrary. Since $|S| > 4n/\epsilon$, the B-trace $\vec{d}' = \mathrm{Tr}_B(S_{k,\vec{d},B})$ of the sequence $S_{k,\vec{d},B}$ of Lemma 10.2.24 satisfies

$$|\mathrm{maxp}_{\varphi,A}(\vec{d}') - \mathrm{maxp}_{\varphi,A}(\vec{d})| = |\mathrm{maxp}_{\varphi,A}(\vec{d}') - \mathrm{MIN}(L_B)| < \epsilon/2.$$

So $\mathrm{maxp}_{\varphi,A}(\mathrm{Tr}_A(S)) - \mathrm{maxp}_{\varphi,A}(\vec{d}') > \epsilon/2$. As in the proof of Lemma 10.2.25, we have

$$\mathrm{cf}_A(\varphi, \mathrm{Tr}_A(S), |S|) \geq \frac{\min_B \Pr_{D,\vec{\alpha}}(B) o^{\mathrm{maxp}_{\varphi,A}(\mathrm{Tr}_A(S))|S|}}{2^n o^{|S|}}$$

for S and

$$\mathrm{cf}_B(\varphi, \vec{d}', |S|) \leq \frac{2^n o^{\mathrm{maxp}_{\varphi,A}(\vec{d}')|S|}}{\Pr_{D,\vec{\alpha}}(B) o^{|S|}}$$

for the sequence that approximates $\vec{d}$, and hence

$$\begin{aligned}
& \Pr_{D,\vec{\alpha}}(\varphi \mid S_{|S|,\vec{d},B}) > \Pr_{D,\vec{\alpha}}(\varphi \mid S) \\
\text{iff} \quad & \mathrm{cf}(\varphi, S_{|S|,\vec{d},B}) < \mathrm{cf}(\varphi, S) \\
\text{if} \quad & (\ldots) \\
\text{iff} \quad & (\mathrm{maxp}_{\varphi,A}(\mathrm{Tr}_A(S)) - \mathrm{maxp}_{\varphi,A}(\vec{d}\,'))|S| > \log_o\left(\frac{2^{2n}}{\Pr_{D,\vec{\alpha}}(B)\min_C \Pr_{D,\vec{\alpha}}(C)}\right).
\end{aligned}$$

But $|S| > k_0 \geq \frac{2}{\epsilon}\log_o\left(\frac{2^{2n}}{\Pr_{D,\vec{\alpha}}(B)\min_C \Pr_{D,\vec{\alpha}}(C)}\right)$, and hence S is indeed not optimal. □

With these pieces, we can finally prove Theorem 10.2.20.

Proof (of Theorem 10.2.20). Suppose, by way of contradiction, that the antecedent of Theorem 10.2.20 holds, but φ does not exhibit RI. Let δ be the function of Lemma 10.2.25 and let C be the constant that is assumed to exist in the statement of Theorem 10.2.20. Define f by taking $f(k) = \delta(k)k$. Since $\lim_{k\to\infty} f(k)/k = \lim_{k\to\infty}\delta(k) = 0$, f is negligible. By Proposition 10.2.2, there exists an open-minded product distribution D and accuracy vector α such that for all c, there exists an (f,c)-bad test-outcome sequence optimal for φ, D, and $\vec{\alpha}$. So by Lemma 10.2.22, taking $c = C/2$, there exists an infinite sequence $\{S_k\}$ of $(f, C/2, \varphi)$-bad test-outcome sequences that are optimal for φ, D, and $\vec{\alpha}$ and are of increasing length. Thus,

$$\begin{aligned}
&\text{for all } k\text{, there are no variables } v_j \geq_\varphi v_i \text{ such that } v_j \text{ is tested at most} \\
&f(|S_k|) \text{ times, but } v_i \text{ is tested at least } C|S_k|/2 \text{ times.}
\end{aligned} \tag{10.10}$$

We can assume without loss of generality that all the sequences S_k are compatible with the same assignment A, since there must be an assignment A that infinitely many of the sequences S_k are compatible with, and we can consider the subsequence consisting just of these test-outcomes sequences that are compatible with A. Moreover, by Lemma 10.2.26, we can assume that A is relevant, since all but finitely many of the S_k must be sufficiently long.

Let k_1 be sufficiently large that $\delta(k) < C/2$ for all $k > k_1$. By Lemma 10.2.25, for all $k > k_1$, we must have

$$\|\vec{d} - \mathrm{Tr}_A(S_k)\|_1 < \delta(k) < C/2$$

for some solution $(\vec{d}, m)$ to the LP $L_A(\varphi)$. Since A is relevant by construction, the assumptions of the theorem guarantee that there exist i and j such that $v_i \leq_\varphi v_j$, $d_i > C$, and $d_j = 0$. Since $\|\vec{d} - \mathrm{Tr}_A(S_k)\|_1 < \delta(|S_k|)$, it follows that $(\mathrm{Tr}_A(S_k))_i > C - \delta(|S_k|) > C/2$ and $(\mathrm{Tr}_A(S_k))_j < \delta(|S_k|)$. Since each sequence S_k is compatible with A, for each variable v_h, $n^+_{S_k,A,h}$ is just the number of times that v_h is tested in S_k, so $(\mathrm{Tr}_A(S_k))_h$ is the number of times that v_h is tested divided by $|S_k|$. This means that we have a contradiction to (10.10). □

10.2.4 Using LPs for nonconvex properties

Theorem 10.2.20 gives us a criterion that is sufficient to conclude that a given formula exhibits rational inattention: there exists a C such that for all relevant assignments A, and all $\vec{c}$ such that $(\vec{c}, m) \in \mathrm{OPT}(L_A(\varphi))$ for some m, there exist entries c_i and c_j such that $v_i \leq_\varphi v_j$, $c_i \geq C$, and $c_j = 0$. We call this property P_C, and write $P_C(\vec{c})$ if $\vec{c}$ satisfies the property. To compute how many formulae exhibit RI, we want an efficient algorithm that evaluates P_C. Since the quantification over C is existential, and $C < C' \Rightarrow (P_C(\vec{c}) \Rightarrow P_{C'}(\vec{c}))$, all choices of C makes P_C a sufficient condition for RI, with smaller C giving rise to stronger conditions. We found no difference for the formulae that we tested between taking C to be 10^{-4}, 10^{-5}, or 10^{-6}, and hence arbitrarily chose $C = 10^{-5}$ for our computations.

LPs (such as $L_A(\varphi)$) are known to be solvable in polynomial time (see, e.g., [23]). However, rather than finding a description of the entire solution polytope

OPT($L_A(\varphi)$), standard linear programming algorithms such as that of Karmarkar [23] compute only a single point inside the polytope. Since we are interested in whether *all* points in the polytope satisfy P_C, we have to do some additional work before we can leverage standard LP solvers.

In general, to establish whether all points in OPT(L) for a minimising LP L satisfy a property P, we can compare the objective values attained at feasible points for which P is true to those attained at points for which P is false. If some point where P is true attains a smaller value than all points where P is false, then no $\neg P$ point can be optimal, and so P is true for all optimal points; likewise, if some point where P is false attains a smaller value than all points where P is true, then P is false for all optimal points. If neither relationship holds, then points in OPT(L) can satisfy both P and its negation. Of course, it is unclear how we would compare all pairs of points from two infinite sets in general. However, if we know the minimum objective value m^+ across all feasible points that satisfy P, then the first property above can be simplified to say: is m^+ alone smaller than the objective at any point where P is false? (If the minimum exists, it is by definition attained at some point. Conversely, if the objective at some point that satisfies P is smaller than the objective at all points that don't satisfy P, then m^+ must also be smaller than all these points by transitivity of $<$.)

It would be convenient if we could indeed determine such an m^+ for P_C, for instance by solving another LP. However, the subset of feasible points that satisfy P_C may not actually be a convex polytope. Indeed, a priori it may not even be well-defined, as the minimum of a linear function may not be attained on a set that is not closed. In fact, the property that we care about, that is, the existence of indices i and j such that $v_i \leq_\varphi v_j$, $c_i \geq C$, and $c_j = 0$, is not even closed

under convex combinations, let alone expressible as a set of linear inequalities. For example, if $v_i \leq_\varphi v_j$ and $v_j \leq_\varphi v_i$ are two variables of equal relevance, and $C_1 = 0.15$, then the points $(\ldots, 0, \ldots, 0.2, \ldots)$ and $(\ldots, 0.2, \ldots, 0, \ldots)$ (the filled-in entries correspond to coordinates i and j) satisfy the property for i and j, but their average $(\ldots, 0.1, \ldots, 0.1, \ldots)$ does not. However, for fixed i and j, the condition that $c_i \geq C$ and $c_j = 0$ can be imposed easily on a feasible solution by adding the two inequalities in question to the LP. The set of points that satisfy the existentially quantified condition therefore can be covered by a $O(n^2)$-sized family of convex closed polytopes, over which we can minimise m as a linear program, and determine the overall minimum m^+ by taking the minimum over the individual minima.

Definition 10.2.27. For all variables $v_i \neq v_j$ with $v_i \leq_\varphi v_j$, define

$$L^+_{A,i,j}(\varphi, C) = L_A(\varphi) \cup \{c_j = 0, c_i \geq C\}$$

(so, roughly speaking, in solutions to $L^+_{A,i,j}(\varphi, C)$, variable v_j is ignored while v_i is tested in a constant fraction of the tests). □

Clearly, $\bigcup L^+_{A,i,j}(\varphi, C) = \{\vec{p} \in \mathit{Feas}(L_A(\varphi)) : P_C(\vec{p})\}$, so $\min_{i,j} \mathrm{MIN}(L^+_{A,i,j}) = m^+$. It would be convenient if we could similarly determine a minimum m^- for all points where $\neg P_C$, and thereby get rid of the quantification over those points as well and simply compare m^+ to m^-. The negation of P_C is equivalent to

$$\forall j(\, c_j = 0 \Rightarrow \forall i(\, v_i \leq_\varphi v_j \Rightarrow c_i < C)) :$$

if a variable v_j is ignored, then every variable v_i that is no more important than v_j is also "pretty much ignored", that is, not even given a C-fraction of tests for the cutoff constant C we picked. Analogously to before, we can now define a collection of convex polytopes whose union is the set of points on which P_C

does not hold. For each polytope, we just fix the (not necessarily single) most important variable v_j that is ignored. Let

$$S^-_{A,j} = \{(c_1, \ldots, c_n, m) \in \mathit{Feas}(L_A(\varphi)) : c_i < C \ \forall i : v_i \leq_\varphi v_j, c_i > 0 \ \forall i : v_i \not\leq_\varphi v_j\}.$$

Observe then that, indeed, $\bigcup_j S^-_{A,j}(\varphi, C) \supseteq \{\vec{p} \in \mathit{Feas}(L_A(\varphi)) : \neg P_C(\vec{p})\}$. Unfortunately, the definition of each $S^-_{A,j}$ involves some strict inequalities, so the sets are not closed. Therefore, we can't use standard LP techniques to find m^-, and indeed, the minimum might not even be attained on the set.

While we may not be able to minimise linear functions over non-closed convex polytopes, we can make use of a standard trick to determine whether such a polytope is at least nonempty. This turns out to be sufficient for our purposes.

Proposition 10.2.28. *We can decide, in time polynomial in the number of variables and the number of bits required to describe the inequalities, whether a set that is defined by linear inequalities (strict or non-strict) is empty.*

Proof. Take the inequalities defining a set U to be $f_1(\vec{x}) \leq c_1, \ldots, f_n(\vec{x}) \leq c_n$ and $g_1(\vec{x}) < d_1, \ldots, g_m(\vec{x}) < d_m$. Then the LP

$$\begin{array}{lrl} \text{minimise} & s & \\ \text{subject to} & f_1(\vec{x}) & \leq c_1 \\ & & \vdots \\ & f_n(\vec{x}) & \leq c_n \\ & g_1(\vec{x}) & \leq c_1 + s \\ & & \vdots \\ & g_m(\vec{x}) & \leq c_m + s \end{array}$$

has a solution $s^* < 0$ iff U is nonempty: If the LP has a solution $s^* < 0$ then the solution point $\vec{x}^*$ also satisfies the inequalities defining U; conversely, all

points $x \in U$ must satisfy all inequalities, so the non-strict inequalities in the LP are immediately satisfied. For the strict ones, there exist $s_1, \ldots, s_m < 0$ such that $g_1(x) = c_1 + s_1 < c_1, \ldots, g_m(x) = c_m + s_m < c_m$. Hence, taking $s = \max_i s_i < 0$, (x, s) satisfies the corresponding non-strict LP inequalities as well, and the solution $s^* \leq s < 0$.

This LP has one more variable than the original set of inequalities, and clearly can be described using at most a polynomially greater number of bits than the original under any reasonable encoding. The result follows by using Karmarkar's algorithm [23]. □

As we noted earlier, we can establish that P_C is true if $m^+ < m$ for all m such that $(x, m) \in \mathit{Feas}(L_A(\varphi))$ and $\neg P_C(x, m)$. This is true iff *no* point that satisfies $\neg P_C$ attains a value $m \leq m^+$; and each such point must come from at least one of the $S^-_{A,j}$. In other words, defining

$$T^-_{A,j}(\varphi, C, m^+) = \{(c_1, \ldots, c_n, m) \in S^-_{A,j}(\varphi, C) : m \leq m^+\} = \varnothing,$$

we require all the sets $T^-_{A,j}$ to be empty. By the proposition above, we can decide this in polynomial time.

Theorem 10.2.29. *Fix $C > 0$ and set $m^+_C = \min_{A,i,j} \mathrm{MIN}(L^+_{A,i,j}(\varphi, C))$. If $T^-_{A,j}(\varphi, C, m^+_C) = \varnothing$ for all A and j, then φ exhibits rational inattention.*

Proof. As explained above, the sets $T^-_{A,j}$ being empty implies that there is no point satisfying $\neg P_C$ and attaining a max-power of $m \leq m^+_C$. At the same time, m^+_C being the minimum over all inattentive LPs means that the minimum of m over points satisfying P_C in any L_A is m^+_C. Therefore, m^+_C is the minimax power, and all solution points of relevant LPs satisfy P_C. Hence, by Theorem 10.2.20, φ exhibits RI. □

Corollary 10.2.30. *We can compute a sufficient condition for the n-variable formula φ to exhibit RI by solving $2^n O(n^2)$ LPs with $O(2^n)$ inequalities each, namely the $O(n^2)$ inattentive LPs and the $O(n)$ attentive LPs associated with each of the 2^n assignments.*

10.3 Rational inattention is widespread

Using this approach, we were able to exhaustively test all formulae that involve at most 4 variables to see whether, as the number of tests in the game increases, optimal strategies were testing a more relevant variable a negligible number of times relative to a less relevant variable. Since the criterion that we use is only a sufficient condition, not a necessary one, we can give only a lower bound on the true number of formulae that exhibit RI.

In the following table, we summarise our results. The first column lists the number of formulae that we are certain exhibit RI; the second column lists the remaining formulae, whose status is unknown. (Since RI is a semantic condition, when we say "formula", we really mean "equivalence class of logically equivalent formulae". There are 2^{2^n} equivalence classes of formulae with n variables, so the sum of the two columns in the row labeled n is 2^{2^n}.) As the results show, at least 15% of formulae exhibit RI.

n	exhibit RI	unknown
1	0	4
2	8	8
3	40	216
4	9952	55584

Table 1: Exhaustive counts of formulae in few variables exhibiting RI

Given the numbers involved, we could not exhaustively check what happens for $n \geq 5$. However, we did randomly sample 4000 formulae that involved n variables for $n = 5, \ldots, 9$. This is good enough for statistical reliability: we can model the process as a simple random sample of a binomially distributed parameter (the presence of RI), and in the worst case (if its probability in the population of formulae is exactly $\frac{1}{2}$), the 95% confidence interval still has width $\leq z\sqrt{\frac{1}{4000}\frac{1}{2}\left(1-\frac{1}{2}\right)} \approx 0.015$, which is well below the fractions of formulae exhibiting RI that we observe (all above 0.048). As the following table shows, RI continued to be quite common. Indeed, even for formulae with 9 variables, about 5% of the formulae we sampled exhibited RI.

n	exhibit RI	unknown
5	585	3415
6	506	3494
7	293	3707
8	234	3766
9	194	3806

Table 2: Counts of formulae in more vars exhibiting RI in a random sample

The numbers suggest that the fraction of formulae exhibiting RI decreases as the number of variables increases. However, since the formulae that characterise situations of real-life interest are likely to involve relatively few variables (or have a structure like disjunction or conjunction that we know exhibits RI), this suggests that RI is a widespread phenomenon. Indeed, if we weaken the notion of RI slightly (in what we believe is quite a natural way!), then RI is even more widespread. As noted in Example 10.1.5, formulae of the form

$v_1 \vee (\neg v_1 \wedge v_2 \wedge \ldots \wedge v_n)$ do not exhibit RI in the sense of our definition. However, for these formulae, if we choose the payoffs b and g appropriately, an optimal strategy may start by testing v_1, but if sufficiently many test outcomes are $v_1 \approx F$, it will then try to establish that the formula is false by focussing on one variable of the conjunction $(v_2 \wedge \ldots \wedge v_n)$, and ignoring the rest. Thus, for all optimal strategies, we would have RI, not for all test-outcome sequences (i.e., not in all histories of the game), but on a set of test-outcome sequences that occur with positive probability.

We found it hard to find formulae that do not exhibit RI in this weaker sense. In fact, we conjecture that the only family of formulae that do not exhibit RI in this weaker sense are equivalent to XORs in zero or more variables $(v_1 \oplus \ldots \oplus v_n)$ and their negations. (Note that this family of formulae includes v_i and $\neg v_i$.) If this conjecture is true, we would expect to quite often see rational agents (and decision-making computer programs) ignoring relevant variables in practice.

CHAPTER 11

TESTING AS A MEASURE OF COMPLEXITY

The notion of associating some "intrinsic difficulty" with concepts (typically characterised using Boolean formulae) has been a topic of continued interest in the cognitive science community [41, 11, 27, 36]. We can use our formalism to define a notion of difficulty for concepts. Our notion of difficulty is based on the number of tests that are needed to guarantee a positive expected payoff for the game $G(\varphi, D, k, \vec{\alpha}, g, b)$. This will, in general, depend on D, $\vec{\alpha}$, g, and b. Actually, by Lemma 9.2.3, what matters is not g and b, but $q(b, g)$ (the threshold determined by g and b). Thus, our complexity measure takes D, $\vec{\alpha}$, and q as parameters.

Definition 11.0.1. Given a formula φ, accuracy vector $\vec{\alpha}$, distribution D, and threshold $0 < q \leq \frac{1}{2}$, the *$(D, q, \vec{\alpha})$-test complexity* $\mathrm{cpl}_{\mathrm{D,q},\vec{\alpha}}(\varphi)$ of φ is the least k such that there exists a strategy with positive payoff for $G(\varphi, D, k, \vec{\alpha}, g, b)$, where g and b are chosen such that $q(b, g) = q$. □

To get a sense of how this definition works, consider what happens if we consider all formulae that use two variables, v_1 and v_2, with the same settings as in Example 9.1.1: $\vec{\alpha} = (1/4, 1/4)$, D is the uniform distribution on assignments, $g = 1$, and $b = -16$:

1. If φ is simply T or F, any strategy that guesses the appropriate truth value, regardless of test outcomes, is optimal and gets a positive expected payoff, even when $k = 0$. So $\mathrm{cpl}_{\mathrm{D,q},\vec{\alpha}}(\varphi) = 0$.
2. If φ is a single-variable formula of the form v_1 or $\neg v_1$, then the greatest certainty $|\Pr_{D,\vec{\alpha}}(\varphi \mid S) - 1/2|$ that is attainable with any sequence of two tests

is 2/5, when $S = (v_1 \approx T, v_1 \approx T)$ or the same with F. This is smaller than $q(b, g)$, and so it is always optimal to make no guess; that is, all strategies for the game with $k = 2$ have expected payoff at most 0. If $k = 3$ and $S = (v_1 \approx T, v_1 \approx T, v_1 \approx T)$, then $(\Pr_{D,\vec{\alpha}}(\varphi \mid S) - 1/2) = 13/28 > q(b, g)$. Thus, if $k = 3$, the strategy that tests v_1 three times and guesses the appropriate truth value iff all three tests agree has positive expected payoff. It follows that $\mathrm{cpl}_{\mathrm{D,q},\vec{\alpha}}(\varphi) = 3$.

3. If φ is $v_1 \oplus v_2$, then the shortest test-outcome sequences S for which $\Pr_{D,\vec{\alpha}}(\varphi \mid S) - 1/2$ is greater than $q(b, g)$ have length 7, and involve both variables being tested. Hence, the smallest value of k for which strategies with payoff above 0 exist is 7, and $\mathrm{cpl}_{\mathrm{D,q},\vec{\alpha}}(\varphi) = 7$.

4. As Example 9.1.1 shows, $\mathrm{cpl}_{\mathrm{D,q},\vec{\alpha}}(v_1 \vee v_2) = 2$, and likewise for all other two-variable conjunctions and disjunctions by symmetry.

It is not hard to see that T and F always have complexity 0, while disjunctions and conjunctions have low complexity. Peraps somewhat counterintuitively, the disjunction $v_1 \vee v_2$ has *lower* complexity than v_1; moreover, the larger k is, the lower the complexity of $v_1 \vee \ldots \vee v_k$. This can be intuitively justified by noting that this measure of complexity captures how much work it takes to attain *certainty* about the truth value of a formula. Longer disjunctions are progressively more likely to be true *a priori*; moreover, any evidence (in the form of test outcomes) that a given disjunction of k variables is true is at least as compelling evidence that an extension of this disjunction to $k + 1$ variables is. Therefore, even though the formula looks more complex, the case for it being true actually becomes easier to make.

We can also characterise the most difficult concepts, according to our com-

plexity measure, at least in the case of a uniform distribution D_u on truth assignments (which is the one most commonly considered in practice).

Theorem 11.0.2. *Among all Boolean formulae in n variables, for all $0 < q \leq \frac{1}{2}$ and accuracy vectors $\vec{\alpha}$, the $(D_u, q, \vec{\alpha})$-test complexity is maximised by formulae equivalent to the n-variable XOR $v_1 \oplus \ldots \oplus v_n$ or its negation.*

Proof sketch. Call a formula φ *antisymmetric* in variable v if $\varphi(A) = \neg\varphi(A')$ for all pairs of assignments A, A' that only differ in the truth value of v. It is easy to check that a formula is antisymmetric in all variables iff it is equivalent to an XOR or a negation of one. Given a formula φ, the *antisymmetrisation φ_v of φ along v* is is the formula

$$\varphi_v = (v \wedge \varphi|_{v=\mathrm{T}}) \vee (\neg v \wedge \neg\varphi|_{v=\mathrm{T}}),$$

where $\varphi|_{v=x}$ denotes the formula that results from replacing all occurrences of v in φ by x. It is easy to chek that φ_v is indeed antisymmetric in v. We can show that the $(D_u, q, \vec{\alpha})$-test complexity of φ_v is at least as high as that of φ, and that if $v' \neq v$, then φ_v is antisymmetric in v' iff φ is antisymmetric in v'. So, starting with an arbitrary formula φ, we antisymmetrise every variable in turn. We then end up with an XOR or the negation of one. Moreover, each antisymmetrisation step in the process gives a formula whose test complexity is at least as high as that of the formula in the previous step. The desired result follows. A detailed proof can be found in the next section. □

The following example illustrates how the antisymmetrisation process affects test complexity.

Example 11.0.3. Consider the formula $\varphi = (a \wedge c) \vee (\neg a \wedge b)$. In order to better separate the complexity values, we increase the threshold q slightly, from 15/34

to 31/68, which can be achieved by setting *bad* and *good* payoffs of −65 and 3, respectively. We then consider the $(D_u, 31/68, \vec{\alpha})$-test complexity of this formula, as well as different successive antisymmetrisations.

Formula	Equivalent Formula	$\mathrm{cpl}_{\mathrm{D,q},\vec{\alpha}}$
φ	$(a \wedge c) \vee (\neg a \wedge b)$	6
φ_a	$(a \wedge c) \vee (\neg a \wedge \neg c)$	8
$(\varphi_a)_c$	$(a \wedge c) \vee (\neg a \wedge \neg c)$	8
$((\varphi_a)_c)_b$	$a \oplus b \oplus c$	12
φ_b	$(b \wedge \neg a) \vee (b \wedge c) \vee (\neg b \wedge a \wedge \neg c)$	7
$(\varphi_b)_a$	$a \oplus b \oplus c$	12

Table 3: Example values of test complexity

□

Theorem 11.0.2 does not rule out the possibility that there are formulae other than those equivalent to the n-variable XOR or its negation that maximise test complexity. We conjecture that this is not the case except when $q = 0$; this conjecture is supported by experiments we've done with formulas that have fewer than 8 variables.

It is of interest to compare our notion of "intrinsic difficulty" with those considered in the cognitive science literature. That literature can broadly be divided up into purely experimental approaches, typically focused on comparing the performance of human subjects in dealing with different categories, and more theoretical ones that posit some structural hypothesis regarding which categories are easy or difficult.

The work of Shepard, Hovland, and Jenkins ([36]) is a good example of the former type; they compare concepts that can be defined using three variables in terms of how many examples (pairs of assignments and corresponding truth values of the formula) it takes human subjects to understand and remember a formula φ, as defined by a subject's ability to predict the truth value of φ correctly for a given truth assignment. We can think of this work as measuring how hard it is to work with a formula; our formalism is measuring how hard it is to learn the truth value of a formula. The difficulty ranking found experimentally by Shepard et al. mostly agrees with our ranking, except that they find two- and three-variable XORs to be easier that some other formulae, whereas we have shown that these are the hardest formulae. This suggests that there might be differences between how hard it is to work with a concept and how hard it is to learn it.

Feldman ([11]) provides a good example of the latter approach. He proposes the notion of the *power spectrum* of a formula φ. Roughly speaking, this counts the number of antecedents in the conjuncts of a formula when it is written as a conjunction of implications where the antecedent is a conjunction of literals and the conclusion is a single literal. For example, the formula $\varphi = (v_1 \wedge (v_2 \vee v_3)) \vee (\neg v_1 \wedge (\neg v_2 \wedge \neg v_3))$ can be written as the conjunction of three such implications: $(v_2 \rightarrow v_1) \wedge (v_3 \rightarrow v_1) \wedge (\neg v_2 \wedge v_1 \rightarrow v_3)$. Since there are no conjuncts with 0 antecedents, 2 conjuncts with 1 antecedent, and 1 conjunct with 2 antecedents, the power spectrum of φ is $(0, 1, 2)$. Having more antecedents in an implication is viewed as making concepts more complicated, so a formula with a power spectrum of $(0, 1, 1)$ is considered more complicated than one with a power spectrum of $(0, 3, 0)$, and less complicated than one with a power spectrum of $(0, 0, 3)$.

A formula with a power spectrum of the form $(i, j, 0, \ldots, 0)$ (i.e., a formula that can be written as the conjunction of literals and formulae of the form $x \to y$, where x and y are literals) is called a *linear category*. Experimental evidence suggests that human subjects generally find linear categories easier to learn than nonlinear ones [11, 27]. (This may be related to the fact that such formulae are linearly separable, and hence learnable by support vector machines [40].) Although our complexity measure does not completely agree with the notion of a power spectrum, both notions classify XORs and their negations as the most complex; these formulae can be shown to have a power spectrum of the form $(0, \ldots, 0, 2^{n-1})$.

Another notion of formula complexity is the notion of *subjective structural complexity* introduced by Vigo ([41]), where the subjective structural complexity of a formula φ is $|Sat(\varphi)|e^{-\|\vec{f}\|_2}$, where $Sat(\varphi)$ is the set of truth assignments that satisfy φ, $f = (f_1, \ldots, f_n)$, f_i is the fraction of truth assignments that satisfy φ such that changing the truth value of v_i results in a truth assignment that does not satisfy φ, and $\|\vec{f}\|_2 = \sqrt{(f_1)^2 + \cdots + (f_n)^2}$ represents the ℓ^2 norm. Unlike ours, with this notion of complexity, φ and $\neg\varphi$ may have different complexity (because of the $|Sat(\varphi)|$ factor). However, as with our notion, XORs and their negation have maximal complexity.

In computer science and electrical engineering, *binary decision diagrams* (BDDs) [26] are used as a

compact representation of Boolean functions. BDDs resemble our notion of a testing strategy, although they do not usually come with a notion of testing error or acceptable error margins on the output (guess). Conversely, we could view testing strategies as a generalisation of BDDs, in which we could "accidentally"

take the wrong branch (testing noise), a given variable can occur multiple times, leaf nodes can also be labelled "no guess", and the notion of correctness of a BDD for a formula is relaxed to require only that the output be correct with a certain probability. The *expected decision depth* problem of Ron, Rosenfeld, and Vadhan [32] asks how many nodes of a BDD need to be visited in expectation in order to evaluate a Boolean formula; this can also be seen as a measure of complexity. In our setting, an optimal strategy for the "noiseless" information-acquisition game ($\alpha = 1/2$, $-\infty$ payoff for guessing wrong) exactly corresponds to a BDD for the formula; asking about the depth of the BDD amounts to asking about whether the strategy uses more than a given number of tests.

11.1 Proof of Theorem 11.0.2

We previously took the XOR $v_1 \oplus \ldots \oplus v_n$ of n variables (often denoted $\bigoplus_{i=1}^{n} v_i$) to be true iff an odd number of the variables are true. This characterisation is actually a consequence of the following standard definition in terms of basic Boolean connectives, of which we also note some useful properties (whose proof is left to the reader).

Definition 11.1.1. The *exclusive OR* (XOR) $\varphi_1 \oplus \varphi_2$ is equivalent to the formula $(\varphi_1 \wedge \neg\varphi_2) \vee (\neg\varphi_1 \wedge \varphi_2)$. □

Proposition 11.1.2. *(Properties of XOR)*

(a) XOR is commutative: $\varphi_1 \oplus \varphi_2 \equiv \varphi_2 \oplus \varphi_1$*;*

(b) XOR is associative: $(\varphi_1 \oplus \varphi_2) \oplus \varphi_3 \equiv \varphi_1 \oplus (\varphi_2 \oplus \varphi_3)$*;*

(c) $v_1 \oplus \ldots \oplus v_n$ *is true iff an odd number of the variables* v_i *is;*

(d) $\neg\varphi \equiv T \oplus \varphi$, *so* $\varphi_1 \oplus \neg\varphi_2 \equiv \neg\varphi_1 \oplus \varphi_2 \equiv \neg(\varphi_1 \oplus \varphi_2)$.

As we said in the proof sketch in the main text, our proof uses the idea of antisymmetry.

The notion of antisymmetry has the useful property that φ_v, the antisymmetrisation of φ along v (recall that φ_v was defined as $(v \wedge \varphi|_{v=\mathrm{T}}) \vee (\neg v \wedge \neg\varphi|_{v=\mathrm{T}})$) is antisymmetric in v and, as we now show, also antisymmetric in all other variables v' that φ was antisymmetric in.

Lemma 11.1.3. *If φ is antisymmetric in a variable $v' \neq v$, then so is φ_v.*

Proof. Suppose that φ is antisymmetric in $v' \neq v$. Then for all truth assignments A, we have

- $\varphi(A[v \mapsto \mathrm{T}]) = \neg\varphi(A[v' \mapsto \mathrm{F}])$ and
- $\varphi_v(A) = \begin{cases} \varphi(A[v \mapsto \mathrm{T}]) & \text{if } A(v) = \mathrm{T} \\ \neg\varphi(A[v \mapsto \mathrm{T}]) & \text{if } A(v) = \mathrm{F}. \end{cases}$

Thus, if $A(v) = \mathrm{T}$, then

$$\begin{aligned} \varphi_v(A[v' \mapsto \mathrm{T}]) &= \varphi(A[v \mapsto \mathrm{T}, v' \mapsto \mathrm{T}]) \\ &= \neg\varphi(A[v \mapsto \mathrm{T}, v' \mapsto \mathrm{F}]) \\ &= \neg\varphi_v(v' \mapsto \mathrm{F}]), \end{aligned}$$

and if $A(v) = \mathrm{F}$, then

$$\begin{aligned} \varphi_v(A[v' \mapsto \mathrm{T}]) &= \neg\varphi(A[v \mapsto \mathrm{T}, v' \mapsto \mathrm{T}]) \\ &= \varphi(A[v \mapsto \mathrm{T}, v' \mapsto \mathrm{F}]) \\ &= \neg\varphi_v(A[v' \mapsto \mathrm{F}]). \end{aligned}$$

Thus, no matter what $A(v)$ is, we have $\varphi_v(A[v' \mapsto \mathrm{T}]) = \neg\varphi_v(A[v' \mapsto \mathrm{F}])$, as required. □

Define $V(\varphi)$, the number of variables a formula φ is *not* antisymmetric in, as

$$V(\varphi) = |\{v : \varphi \not\equiv (v \wedge \varphi|_{v=\mathrm{T}}) \vee (\neg v \wedge \neg\varphi|_{v=\mathrm{T}})\}|.$$

Lemma 11.1.4. *The only formulae φ in the n variables $v_1, \ldots, v_n$ for which $V(\varphi) = 0$ are equivalent to either $\bigoplus_{i=1}^{n} v_i$ or $\neg \bigoplus_{i=1}^{n} v$.*

Proof. By induction on n. If $n = 1$, then it is easy to check that both v_1 and $\neg v_1$ are antisymmetric. Suppose that $n > 1$ and φ is antisymmetric in $v_1, \ldots, v_n$. Since $\varphi \equiv (v_n \wedge \varphi|_{v_n=\mathrm{T}}) \vee (\neg v_n \wedge \varphi|_{v_n=\mathrm{F}})$ and φ is antisymmetric in v_n, by Definition 11.1.1 we have that

$$\varphi \equiv (v_n \wedge \varphi|_{v_n=\mathrm{T}}) \vee (\neg v_n \wedge \neg\varphi|_{v_n=\mathrm{T}}) \equiv v_n \oplus \varphi|_{v_n=\mathrm{T}}. \tag{11.1}$$

It is easy to see that $\varphi|_{v_n=\mathrm{T}}$ mentions only the variables $v_1, \ldots, v_{n-1}$ and, by Lemma 11.1.3, is antisymmetric in each of them. So by the induction hypothesis, $\varphi|_{v_n=\mathrm{T}}$ is equivalent to either $\bigoplus_{i=1}^{n-1} v_i$ or $\neg(\bigoplus_{i=1}^{n-1} v_i)$, and hence by Proposition 11.1.2(d) and (11.1), φ is equivalent to either $\bigoplus_{i=1}^{n} v_i$ or $\neg(\bigoplus_{i=1}^{n} v_i)$. □

To complete the proof of Theorem 11.0.2, we make use of the following two technical lemmas. For the remainder of the proof, we use $v = \mathrm{T}$ and $v = \mathrm{F}$ to denote the events (i.e., the set of histories) where the variable v is true (resp., false). (We earlier denoted these events v and $\neg v$, respectively, but for this proof the $v = b$ notation is more convenient.)

Lemma 11.1.5. *If D is a product distribution and S is a test-outcome sequence, then the projection of a formula $\varphi|_{v_i=b}$ has the same conditional probability on S as φ additionally conditioned on $v_i = b$, that is,*

$$\Pr_{D,\vec{\alpha}}(\varphi \mid S, v_i = b) = \Pr_{D,\vec{\alpha}}(\varphi|_{v_i=b} \mid S).$$

Proof. Given a truth assignment A on $v_1, \ldots, v_n$, let A_i be A restricted to all the variables other than v_i. Since D is a product distribution, $\Pr_{D,\vec{\alpha}}(A) = \Pr_{D,\vec{\alpha}}(A_i) \times \Pr_{D,\vec{\alpha}}(v_i = A(v_i))$.

Note that the truth of $\varphi|_{v_i=b}$ does not depend on the truth value of v_i. Thus, we can pair the truth assignments that make $\varphi|_{v_i=b}$ true into groups of two, that differ only in the truth assignment to v_i. Suppose that the test $v_i \approx \mathrm{T}$ appears in S k_T times and the test $v_i \approx \mathrm{F}$ appears in S k_F times. Using Lemma 10.2.6, we have that

$$\begin{aligned}
\Pr_{D,\vec{\alpha}}(\varphi|_{v_i=b} \mid S) &= \textstyle\sum_{\{A:\, \varphi|_{v_i=b}(A)=T\}} \Pr_{D,\vec{\alpha}}(A \mid S) \\
&= \frac{\sum_{\{A:\, \varphi|_{v_i=b}(A)=T\}} r_{D,\vec{\alpha}}(A,S)}{\sum_{\text{truth assignments } A'} r_{D,\vec{\alpha}}(A',S)} \\
&= \frac{\sum_{\{A:\, \varphi|_{v_i=b}(A)=T\}} r_{D,\vec{\alpha}}(A_i,S)(\Pr_{D,\vec{\alpha}}(v_i=T)o^{k_T} + \Pr_{D,\vec{\alpha}}(v_i=F)o^{k_F})}{\sum_{\text{truth assignments } A'} r_{D,\vec{\alpha}}(A'_i,S)(\Pr_{D,\vec{\alpha}}(v_i=T)o^{k_T} + \Pr_{D,\vec{\alpha}}(v_i=F)o^{k_F})} \\
&= \frac{\sum_{\{A:\, \varphi|_{v_i=b}(A)=T\}} r_{D,\vec{\alpha}}(A_i,S)}{\sum_{\text{truth assignments } A'} r_{D,\vec{\alpha}}(A'_i,S).}
\end{aligned} \tag{11.2}$$

Using the same arguments as in (11.2), we get that

$$\Pr_{D,\vec{\alpha}}(\varphi \wedge v_i = b \mid S) = \frac{\sum_{\{A:\, (\varphi \wedge v_i=b)(A)=T\}} r_{D,\vec{\alpha}}(A_i, S) \Pr_{D,\vec{\alpha}}(v_i = b) o^{k_T}}{\sum_{\text{truth assignments } A'} r_{D,\vec{\alpha}}(A', S)}$$

and

$$\Pr_{D,\vec{\alpha}}(v_i = b \mid S) = \frac{\sum_{\{A:\, A(v_i)=b\}} r_{D,\vec{\alpha}}(A_i, S) \Pr_{D,\vec{\alpha}}(v_i = b) o^{k_T}}{\sum_{\text{truth assignments } A'} r_{D,\vec{\alpha}}(A', S).}$$

Let $C = \Pr_{D,\vec{\alpha}}(v_i = b \mid S) = \frac{\Pr_{D,\vec{\alpha}}(v_i=b)k_b}{\Pr_{D,\vec{\alpha}}(v_i=T)k_T + \Pr_{D,\vec{\alpha}}(v_i=F)k_F}$ be the probability that $v_i = b$ *after* observing the sequence. Note that

$$\sum_{\{A:\, (\varphi \wedge v_i=b)(A)=T\}} r_{D,\vec{\alpha}}(A_i, S) \quad = \sum_{\{A:\, (\varphi|_{v_i=b} \wedge v_i=b)(A)=T\}} r_{D,\vec{\alpha}}(A_i, S) \quad = \quad C \cdot \sum_{\{A:\varphi|_{v_i=b}(A)=T\}} r_{D,\vec{\alpha}}(A_i, S)$$

and

$$\sum_{\{A:\, A(v_i)=b\}} r_{D,\vec{\alpha}}(A_i, S) \quad = \quad C \cdot \sum_{\text{truth assignments } A} r_{D,\vec{\alpha}}(A_i, S).$$

Since, by Bayes' Rule,

$$\Pr_{D,\vec{\alpha}}(\varphi \mid S, v_i = b) = \frac{\Pr_{D,\vec{\alpha}}(\varphi \wedge v_i = b \mid S)}{\Pr_{D,\vec{\alpha}}(v_i = b \mid S)},$$

simple algebra shows that $\Pr_{D,\vec{\alpha}}(\varphi \mid S, v_i = b) = \Pr_{D,\vec{\alpha}}(\varphi|_{v_i=b} \mid S)$, as desired. □

Lemma 11.1.6. *If, for all test-outcome sequences S, there exists a test-outcome sequence S' such that $|S'| = |S|$ and $|\Pr_{D,\vec{\alpha}}(\varphi \mid S') - 1/2| \geq |\Pr_{D,\vec{\alpha}}(\psi \mid S) - 1/2|$, then $\mathrm{cpl}_{\mathrm{D,q},\vec{\alpha}}(\varphi) \leq \mathrm{cpl}_{\mathrm{D,q},\vec{\alpha}}(\psi)$.*

Proof. Suppose that $\mathrm{cpl}_{\mathrm{D,q},\vec{\alpha}}(\psi) = k$. Then there must be some strategy σ for $G(\psi, D, k, \vec{\alpha}, g, b)$ that has positive expected payoff. There must therefore be some test-outcome sequence S of length k that is observed with positive probability when using σ such that the expected payoff of making the appropriate guess is positive. By Lemma 9.2.3, $|\Pr_{D,\vec{\alpha}}(\psi \mid S) - 1/2| > q$.

Since $|\Pr_{D,\vec{\alpha}}(\varphi \mid S') - 1/2| \geq |\Pr_{D,\vec{\alpha}}(\psi \mid S) - 1/2|$ by assumption, there must exist a test-outcome sequence S' such $|\Pr_{D,\vec{\alpha}}(\varphi \mid S') - 1/2| > q$. Let σ' be the strategy for the game $G(\varphi, D, k, \vec{\alpha}, g, b)$ that tests the same variables that are tested in S', and makes the appropriate guess iff S' is in fact observed. By Lemma 9.2.3, a guess with positive expected payoff can be made if S' is observed, which it is with positive probability. So σ' has positive expected payoff, and hence $\mathrm{cpl}_{\mathrm{D,q},\vec{\alpha}}(\varphi)$ is at most k. □

We can now finally prove Theorem 11.0.2. Note that this is the only part of the derivation that actually depends on the assumption that we are working with the uniform distribution D_u.

Proof of Theorem 11.0.2. We show by induction on $V(\varphi)$ that for all formulae φ, there exists a formula φ_0 with $V(\varphi_0) = 0$ such that $\mathrm{cpl}_{\mathrm{D,q},\vec{\alpha}}\varphi \leq \mathrm{cpl}_{\mathrm{D,q},\vec{\alpha}}\varphi_0$. By Lemma 11.1.4, φ_0 must be equivalent to either $\oplus_{i=1}^{n-1} v_i$ or $\neg(\oplus_{i=1}^{n-1} v_i)$.

If $V(\varphi) = 0$, then we can just take $\varphi_0 = \varphi$. Now suppose that $V(\varphi) > 0$. There there must exist some variable v such that $\varphi|_{v=\mathrm{T}} \neq \neg(\varphi|_{v=\mathrm{F}})$. (Here and below we are viewing formulas as functions on truth assignments, justifying the use of "=" rather than "≡".) Note for future reference that, by construction,

$$\varphi_v|_{v=\mathrm{T}} = \varphi|_{v=\mathrm{T}} \text{ and } \varphi_v|_{v=\mathrm{F}} = \neg\varphi|_{v=\mathrm{T}}. \tag{11.3}$$

By Lemma 11.1.3, if φ is antisymmetric in a variable $v' \neq v$, then so is φ_v. In addition, φ_v is antisymmetric in v. Thus, $V(\varphi_v) < V(\varphi)$. If we can show $\mathrm{cpl}_{\mathrm{D,q},\vec{\alpha}}(\varphi) \leq \mathrm{cpl}_{\mathrm{D,q},\vec{\alpha}}(\varphi_v)$, then the result follows from the induction hypothesis. By Lemma 11.1.6, it suffices to show that for all test-outcome sequences S_1, there exists a sequence S of the same length as S_1 such that $|\Pr_{D,\vec{\alpha}}(\varphi \mid S) - 1/2| \geq |\Pr_{D,\vec{\alpha}}(\varphi_v \mid S_1) - 1/2|$.

Given an arbitrary test-outcome sequence S_1, let $p = \Pr_{D,\vec{\alpha}}(v = \mathrm{T} \mid S_1)$. Thus,

$$\begin{aligned}
&\Pr_{D,\vec{\alpha}}(\varphi_v \mid S_1) = p \Pr_{D,\vec{\alpha}}(\varphi_v \mid S_1, v = \mathrm{T}) + (1-p) \Pr_{D,\vec{\alpha}}(\varphi_v \mid S_1, v = \mathrm{F}) \\
&= p \Pr_{D,\vec{\alpha}}(\varphi_v|_{v=\mathrm{T}} \mid S_1) + (1-p) \Pr_{D,\vec{\alpha}}(\varphi_v|_{v=\mathrm{F}} \mid S_1) \qquad \text{[by Lemma 11.1.5]} \\
&= p \Pr_{D,\vec{\alpha}}(\varphi|_{v=\mathrm{T}} \mid S_1) + (1-p) \Pr_{D,\vec{\alpha}}(\neg\varphi|_{v=\mathrm{T}} \mid S_1) \qquad \text{[by (11.3)]} \\
&= p \Pr_{D,\vec{\alpha}}(\varphi|_{v=\mathrm{T}} \mid S_1) + (1-p)(1 - \Pr_{D,\vec{\alpha}}(\varphi|_{v=\mathrm{T}} \mid S_1).
\end{aligned} \tag{11.4}$$

Set $S_2 = S_1[v \approx \mathrm{F} \leftrightarrow v \approx \mathrm{T}]$, that is, the sequence that is the same as S_1 except that all test outcomes of v are flipped in value. Since $\varphi|_{v=\mathrm{T}}$ does not mention v, $\Pr_{D,\vec{\alpha}}(\varphi|_{v=\mathrm{T}} \mid S_1) = \Pr_{D,\vec{\alpha}}(\varphi|_{v=\mathrm{T}} \mid S_2)$ and likewise for $\varphi|_{v=\mathrm{F}}$. Since $\varphi \equiv (v \wedge \varphi|_{v=\mathrm{T}}) \vee (\neg v \wedge \varphi|_{v=\mathrm{F}})$, we have (using an argument similar to that above)

$$\Pr_{D,\vec{\alpha}}(\varphi \mid S_1) = p \Pr_{D,\vec{\alpha}}(\varphi|_{v=\mathrm{T}} \mid S_1) + (1-p) \Pr_{D,\vec{\alpha}}(\varphi|_{v=\mathrm{F}} \mid S_1) \tag{11.5}$$

and, taking $p' = \Pr_{D,\vec{\alpha}}(v = T \mid S_2)$,

$$\begin{aligned}
\Pr_{D,\vec{\alpha}}(\varphi \mid S_2) &= p' \Pr_{D,\vec{\alpha}}(\varphi|_{v=\mathrm{T}} \mid S_2) + (1-p') \Pr_{D,\vec{\alpha}}(\varphi|_{v=\mathrm{F}} \mid S_2) \\
&= p' \Pr_{D,\vec{\alpha}}(\varphi|_{v=\mathrm{T}} \mid S_1) + (1-p') \Pr_{D,\vec{\alpha}}(\varphi|_{v=\mathrm{F}} \mid S_1).
\end{aligned} \tag{11.6}$$

We claim that $p = 1 - p'$. Suppose that the test $v \approx \mathrm{T}$ appears in S_1 k_T times and the test $v \approx \mathrm{F}$ appears in S_1 k_F times. Thus, the test $v \approx \mathrm{T}$ appears in S_2 k_F times and the test $v \approx \mathrm{F}$ appears in S_1 k_T times. All other tests appear the same number of times in both sequences. By Lemma 10.2.6, since the uniform distribution D_u we are using is in particular a product distribution, for $j = 1, 2$, we have that

$$\Pr_{D,\vec{\alpha}}(v = T \mid S_j) = \sum_{\{A:\, A(v)=\mathrm{T}\}} \Pr_{D,\vec{\alpha}}(A \mid S_j) = \frac{\sum_{\{A:\, A(v)=\mathrm{T}\}} r_{D,\vec{\alpha}}(A, S_j)}{\sum_{A'} r_{D,\vec{\alpha}}(A', S_j)}.$$

Suppose that v is the ith variable v_i. Let $r_1 = o_i^{k_T}$, let $r_2 = o_i^{k_F}$, let $R_1 = \sum_{\{A:\, A(v_i)=T\}} \prod_{j=1, j\neq i}^{n} o_j^{n^+_{S_1,A,j}}$, and let $R_2 = \sum_{\{A:\, A(v_i)=F\}} \prod_{j=1, j\neq i}^{n} o_j^{n^+_{S_1,A,j}}$. For $j = 1, 2$ we have that

$$\sum_{\{A:\, A(v)=\mathrm{T}\}} \Pr_{D,\vec{\alpha}}(A \mid S_j) = \frac{\sum_{\{A:\, A(v)=\mathrm{T}\}} r_{D,\vec{\alpha}}(A, S_j)}{\sum_{A'} r_{D,\vec{\alpha}}(A', S_j)} = \frac{r_j R_j}{r_1 R_1 + r_2 R_2}$$

We claim that $R_1 = R_2$. Indeed, for any assignment A such that $A(v_i) = T$, let A' be the unique assignment such that $A'(v_i) = F$ and $A'(v_j) = A(v_j)$ for all $j \neq i$. Then each choice of A occurs once in the sum R_1 and never in the sum R_2, the corresponding A' occurs once in R_2 but not R_1. Since we are working with the uniform distribution D_u, the summands for A and A' are equal. So we can conclude that $p = 1 - p'$. Combining this with (11.6), we get that

$$\Pr_{D,\vec{\alpha}}(\varphi \mid S_2) \;=\; (1-p)\Pr_{D,\vec{\alpha}}(\varphi|_{v=\mathrm{T}} \mid S_1) + p \Pr_{D,\vec{\alpha}}(\varphi|_{v=\mathrm{F}} \mid S_1). \tag{11.7}$$

Let $Q(E) = \Pr_{D,\vec{\alpha}}(E) - \frac{1}{2}$. By adding $-1/2$ on both sides, equations (11.5) and (11.7) hold with $\Pr_{D,\vec{\alpha}}$ replaced by Q, while (11.4) becomes

$$Q(\varphi_v \mid S_1) = pQ(\varphi|_{v=\mathrm{T}} \mid S_1) - (1-p)Q(\varphi|_{v=\mathrm{T}} \mid S_1).$$

We now show that either $|\mathrm{Q}(\varphi \mid S_1)| \geq |\mathrm{Q}(\varphi_v \mid S_1)|$ or $|\mathrm{Q}(\varphi \mid S_2)| \geq |\mathrm{Q}(\varphi_v \mid S_1)|$. This suffices to complete the proof.

To simplify notation, let $x = \mathrm{Q}(\varphi|_{v=\mathrm{T}} \mid S_1)$ and let $y = \mathrm{Q}(\varphi|_{v=\mathrm{F}} \mid S_1)$. By (11.4), (11.5), and (11.7), we want to show that either $|px + (1-p)y| \geq |px - (1-p)x|$ or $|(1-p)x + py| \geq |px - (1-p)x|$. So suppose that $|px + (1-p)y| < |px - (1-p)x|$. We need to consider four cases: (1) $p \geq 1/2$, $x \geq 0$; (2) $p \geq 1/2$, $x < 0$; (3) $p < 1/2$, $x \geq 0$; and (4) $p < 1/2$, $x < 0$. For (1) , note that if $p \geq 1/2$ and $x \geq 0$, then $0 \leq px - (1-p)x \leq px$. We must have $y < -x$, for otherwise $px + (1-p)y \geq px - (1-p)x$. But then $py + (1-p)x < -(px - (1-p)x)$, so $|py + (1-p)x| > |px - (1-p)x|$. For (2), note that if $p \geq 1/2$ and $x < 0$, then $px - (1-p)x < 0$. We must have $y > -x$, for otherwise $px + (1-p)y \leq px - (1-p)x$, and $|px + (1-p)y| \geq |px - (1-p)x|$. But then $py + (1-p)x > -px + (1-p)x$, so $|py + (1-p)x| > |px - (1-p)x|$. The arguments in cases (3) and (4) are the same as for (1) and (2), since we can simply replace p by $1-q$. This gives us identical inequalities (using q instead of p), but now $q > 1/2$. □

CHAPTER 12

CONCLUSIONS

We have presented the information-acquisition game, a game-theoretic model of gathering information to inform a decision whose outcome depends on the truth of a Boolean formula. We argued that it is hard to find optimal strategies for this model by brute force, and presented the random-test heuristic, a simple strategy that has only weak guarantees but is computationally tractable. It is an open question whether better guarantees can be proven for the random-test heuristic, and whether better approaches to testing that are still more computationally efficient than brute force exist. We used our techniques to show that RI is a widespread phenomenon, at least, for formulae that use at most 9 variables. We argue that this certainly covers most concepts that naturally arise in human discourse. Though it is certainly the case that many propositions (e.g., the outcome of elections) depend on many more variables, human speech and reasoning, for reasons of utterance economy if nothing else, usually involves reducing these to simpler compound propositions (such as the preferences of particular blocks of voters). We hope in future work to get a natural structural criterion for when formulae exhibit RI that can be applied to arbitrary formulae.

Finally, we discussed how the existence of good strategies in our game can be used as a measure of the complexity of a Boolean formula. It would be useful to get a better understanding of whether test complexity captures natural structural properties of concepts.

Although we have viewed the information-acquisition game as a single-agent game, there are natural extensions of it to multi-agent games, where agents are collaborating to learn about a formula. We could then examine dif-

ferent degrees of coordination for these agents. For example, they could share information at all times, or share information only at the end (before making a guess). The goal would be to understand whether there is some structure in formulae that makes them particularly amenable to division of labour, and to what extent it can be related to phenomena such as rational inattention (which may require the agents to coordinate on deciding which variable to ignore).

In our model, we allowed agents to choose to make no guess for a payoff of 0. We could have removed this option, and instead required them to make a guess. We found this setting to be less amenable to analysis, although there seem to be analogues to our results. For instance, as in our introductory example, it is still rational to keep testing the same variable in a disjunction with a probability that is bounded away from zero, no matter how many tests are allowed. However, since giving up is no longer an option, there is also a probability, bounded away from both 0 and 1, that all variables have to be tested (namely when the formula appears to be false, and hence it must be ascertained that all variables are). The definition of test complexity makes sense in the alternative setting as well, though the values it takes change; we conjecture that the theorem about XOR being hardest can be adapted with few changes.

BIBLIOGRAPHY

[1] M. Abadi. Logic in access control. In *Proc. 18th IEEE Symposium on Logic in Computer Science*, pages 228–233, 2003.

[2] I. Beer, S. Ben-David, H. Chockler, A. Orni, and R. J. Trefler. Explaining counterexamples using causality. *Formal Methods in System Design*, 40(1):20–40, 2012.

[3] R. A. Botha and J. H. P. Eloff. Separation of duties for access control enforcement in workflow environments. *IBM Systems Journal*, 40(3):666–682, 2001.

[4] E. Cecchetti, A. C. Myers, and O. Arden. Nonmalleable information flow control. In *Proc. 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017.

[5] Y. R. Chen and M. N. Katehakis. Linear programming for finite state multiarmed bandit problems. *Mathematics of Operations Research*, 11(1):180–183, 1986.

[6] H. Chockler and J. Y. Halpern. Responsibility and blame: A structural-model approach. *Journal of A.I. Research*, 20:93–115, 2004.

[7] M. R. Clarkson and F. B. Schneider. Hyperproperties. *Journal of Computer Security*, 18(6):1157–1210, 2010.

[8] E. Cohen. Information transmission in computational systems. In *Proc. 6th ACM Symposium on Operating Systems Principles*, pages 133–139, 1977.

[9] Ellis S Cohen. Information transmission in sequential programs. *Foundations of secure computation*, pages 297–335, 1978.

[10] M. J. Druzdzel and H. J. Suermondt. Relevance in probabilistic models: "Backyards" in a "small world". In *Working notes of the AAAI–1994 Fall Symposium Series: Relevance*, pages 60–63, 1994.

[11] J. Feldman. An algebra of human concept learning. *Journal of Mathematical Psychology*, 50(4):339 – 368, 2006.

[12] R. Focardi and R. Gorrieri. Classification of security properties (Part I: Information flow). In *Foundations of Security Analysis and Design*, pages 331–396. Springer, 2001.

[13] Riccardo Focardi, Sabina Rossi, and Andrei Sabelfeld. Bridging language-based and process calculi security. In *Proceedings of the 8th International Conference on Foundations of Software Science and Computation Structures*, FOSSACS'05, page 299–315, Berlin, Heidelberg, 2005. Springer-Verlag.

[14] J. A. Goguen and J. Meseguer. Security policies and security models. In *Proc. IEEE Symposium on Security and Privacy*, pages 11–20. 1982.

[15] J. T. Haigh and W. D. Young. Extending the noninterference version of MLS for SAT. *IEEE Transactions on Software Engineering*, SE-13(2):141–150, 1987.

[16] J. Y. Halpern. A modification of the Halpern-Pearl definition of causality. In *Proc. 24th International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pages 3022–3033, 2015.

[17] J. Y. Halpern. *Actual Causality*. MIT Press, Cambridge, MA, 2016.

[18] J. Y. Halpern and J. Pearl. Causes and explanations: A structural-model approach. Part I: Causes. In *Proc. Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI 2001)*, pages 194–202, 2001.

[19] J. Y. Halpern and J. Pearl. Causes and explanations: a structural-model approach. Part I: Causes. *British Journal for Philosophy of Science*, 56(4):843–887, 2005.

[20] A. K. Hirsch and M. R. Clarkson. Belief semantics of authorization logic. In *Proc 2013 ACM SIGSAC Conference on Computer & Communications Security*, 2013.

[21] A. Ibrahim, S. Kacianka, A. Pretschner, C. Hartsell, and G. Karsai. Practical causal models for cyber-physical systems. In J. M. Badger and K. Y. Rozier, editors, *NASA Formal Methods*, page 211–227, 2019.

[22] A. Ibrahim, S. Rehwald, A. Scemama, F. Andres, and A. Pretschner. Caual model extration from attack trees to attribute malicious insider attacks. In *International Workshop on Graphical Models for Security*, 2020.

[23] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proc. 16th ACM Symposium on Theory of Computing*, pages 302–311, 1984.

[24] J. Koppel and D. Jackson. Demystifying dependence. In *Proc. 2020 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, page 48–64, 2020.

[25] J. Lang, P. Liberatore, and P. Marquis. Propositional independence – formula-variable independence and forgetting. *Journal of Artificial Intelligence Research*, 18:391–443, 2003.

[26] C. Y. Lee. Representation of switching circuits by binary-decision programs. *The Bell System Technical Journal*, 38:985–999, 1959.

[27] B. C. Love, D. L. Medin, and T. M. Gureckis. Sustain: A network model of category learning. *Psychological Review*, 111(2):309–332, 4 2004.

[28] H. Mantel. Possibilistic definitions of security—an assembly kit. In *Proc. IEEE Computer Security Foundations Workshop*, pages 185–199, 2000.

[29] Benoît Montagu, B. Pierce, and R. Pollack. A theory of information-flow labels. *2013 IEEE 26th Computer Security Foundations Symposium*, pages 3–17, 2013.

[30] A. C. Myers, A. Sabelfeld, and S. Zdancewic. Enforcing robust declassification. In *Proc. 17th IEEE Computer Security Foundations Workshop*, pages 172–186, 2004.

[31] A. C. Myers, A. Sabelfeld, and S. Zdancewic. Enforcing robust declassification and qualified robustness. *Journal of Computer Security*, 14(2):157–196, 2006.

[32] D. Ron, A. Rosenfeld, and S. Vadhan. The hardness of the expected decision depth problem. *Information Processing Letters*, 101(3):112–118, 2007.

[33] John Rushby. Noninterference, transitivity, and channel-control security policies. Technical report, dec 1992.

[34] A. Sabelfeld and A.C. Myers. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications*, 21(1):5–19, 2003.

[35] Andrei Sabelfeld and David Sands. Declassification: Dimensions and principles. *J. Comput. Secur.*, 17(5):517–548, October 2009.

[36] R. N. Shepard, C. I. Hovland, and H. M. Jenkins. Learning and memorization of classifications. *Psychological Monographs: General and Applied*, 75(3):1–42, 1961.

[37] C. A Sims. Implications of rational inattention. *Journal of Monetary Economics*, 50(3):665–690, 2003.

[38] C. Umans. On the complexity and inapproximability of shortest implicant problems. In *Proc. of Automata, Languages and Programming: 26th International Colloquium (ICALP '99)*, pages 687–696, Berlin, Heidelberg, 1999. Springer.

[39] R. van der Meyden. What, indeed, is intransitive noninterference? In J. Biskup and J. López, editors, *Computer Security – ESORICS 2007*, pages 235–250, 2007.

[40] V. N. Vapnik and A. Y. Lerner. Recognition of patterns using generalized portraits. *Avtomat. i Telemekh.*, 24:774–780, 1963.

[41] R. Vigo. Representational information: a new general notion and measure of information. *Information Sciences*, 181:4847–4859, 2011.

[42] Dennis Volpano, Cynthia Irvine, and Geoffrey Smith. A sound type system for secure flow analysis. *J. Comput. Secur.*, 4(2–3):167–187, January 1996.

[43] M. Wiederholt. Rational inattention. In L. E. Blume and S. Durlauf, editors, *The New Palgrave Dictionary of Economics (online edition)*. Palgrave Macmillan, New York, 2010.

[44] S. Zdancewic and A. C. Myers. Robust declassification. In *Proc. 14th IEEE Computer Security Foundations Workshop*, pages 15–23, 2001.