City University London

Department of Computing

# Quantifying Information Flow with Constraints

Ping Zhu

A thesis submitted for the degree of

Doctor of Philosophy of City University London

October 2010

## Abstract

Quantifying flow of information in a program involves calculating how much information (*e.g.* about secret inputs) can be leaked by observing the program's public outputs. Recently this field has attracted a lot of research interest, most of which makes use of Shannon's information theory, *e.g.* mutual information, conditional entropy, *etc.*

Computability entails that any automated analysis of information is necessarily incomplete. Thus quantitative flow of analyses aim to compute upper bounds on the sizes of the flows in a program. Virtually all the current quantitative analyses treat program variables independently, which significantly limits the potential for deriving tight upper bounds.

Our work is motivated by the intuition that knowledge of the dependencies between program variables should allow the derivation of more precise upper bounds on the size of flows, and that classical abstract interpretation provides an effective mechanism for determining such dependencies in the form of linear constraints. Our approach is then to view the problem as one of constrained optimization (maximum entropy), allowing us to apply the standard technique of Lagrange multiplier method. Application of this technique turns out to require development of some novel methods due to the essential use of non-linear (entropy) constraints, in conjunction with the linear dependency constraints.

Using these methods we obtain more precise upper bounds on the size of information flows than is possible with existing analysis techniques.

# Contents

# List of Figures

# List of Tables

# Acknowledgements

I have been very lucky to have Dr. Sebastian Hunt as my supervisor throughout my whole Ph.D. His enthusiasm and attitude towards research, his rigorous methodology, his kind advice, encouragement and support all made a great impression on me. Moreover he also helped me a lot in other aspects which I really appreciate.

I want to thank Dr. David Clark and Dr. Pasquale Malacaria for all their input and suggestions in every project meeting; and thank Dr. Darrell Conklin and Prof. George Spanoudakis for useful feedback for my transfer.

I am also grateful to my friends: Marcus Andrews, Yan Chen, Gilberto Cysneiros Filho, Tao Gong, Sheng Jing, Gang Li, George Lekeas, MD. Hongfen Li, Tshiamo Motshegwa, Lu Ren, Jens Wissmann, Beibei Zhang, Shi Zhuo and others, especially to Han Chen, Chunyan Mu, Zhi Quan who have encouraged me and shared happy time in London.

Finally and specially, I would like to thank my parents for their great love, support, encouragement and patience through my up and downs during all these years, without which I could never finished my studies.

# Chapter 1

# Introduction

With the ever faster development of computer systems and internet (both in size and complexity) and society's increasing dependence on them, the question of how to protect information from being improperly leaked is of greater and greater importance. This is because governments, military, corporations, financial institutions, hospitals and private businesses amass a great deal of confidential, sensitive and private information about their employees, customers, products, research, financial status *etc.* Besides these large organizations, for individuals, how to protect privacy is also of vital importance. This is the topic of information security which is the broader area our work belongs to.

There are three core principles to information security: confidentiality, integrity and availability (known as the CIA triad). Basically speaking, confidentiality prevents information flowing to inappropriate destinations, while integrity requires that information is prevented from inappropriate sources such as a person sneakily changing their own salary in the payroll database.

Integrity can be breached without any violation of confidentiality and also strong enforcement of integrity usually requires proving program correctness. Finally availability means that information must be available when it is needed. We are mainly concerned with confidentiality here (in the following, the uses of information flow policy, if without any special denotation, mean confidentiality policy). Neither access control nor encryption provide complete solutions for protecting confidentiality as they cannot control the propagation once the secret information is released.

A complementary approach is to track and regulate the *information flow* of the system to prevent confidential data from leaking to unauthorized parties. This can be implemented statically by establishing some predefined policy with respect to the data. Information flow policy is an example of such a policy which enforces limits on how the information contained in the confidential data might be revealed in subsequent computations.

The most well known policy is the *non-interference* [30], which requires that any variation of the secret high input(s) to the system cannot result in any variation of the observable low output(s). Non-interference is widely pointed out to be overly strict as it requires an absolute independence of low variable(s) on high variable(s) (if this holds, then the high variables may as well be removed completely from the system) which is impossible in practical systems. Thus far too many programs would be rejected as insecure by this policy, such as the password checking program which is widely used in our daily life for example an ATM machine, with high (secure) input *password* and low (public) output *output*:

```
if guess==password
    output:=pass;
else
    output:=deny
fi
```

The value of output will vary according to the value of *password* therefore it violates *non-interference* yet it is commonly assumed to be secure; moreover, with the increasingly complexity of modern computer systems, not only do high variables interfere with low variables all the time but high inputs are interacting with each other among themselves, and the ways these variables interact are becoming increasingly complicated. Therefore, non-interference, although fairly rigorous and ideal, has a big gap in terms of feasible application.

Over the years, there has been a lot of effort put into relaxing non-interference, such as robust declassification [59, 58, 71, 42], delimited information release [65], flow-sensitive type systems [37] and abstract non-interference [29].

Our work is to try to bridge this gap as well, however, through a different way—via quantitative information flow analysis.

## 1.1 Quantitative Information Flow Analysis

Non-interference and the later work mentioned above that aims to make it more practical are all qualitative, which means that it can only judge a system as secure or insecure according to their own specifications. However, sometimes only knowing this fact is not as informative as to what extent the system is not secure. If the extent to which the system is considered to be insecure is very low, then the system should still be considered safe to use. As a result, researchers switched away from *secure or insecure* to *how secure*. Thus, quantitative information flow analysis came into being and is attracting more and more interest. It actually calculates how much information is leaked by observing the low output(s) which may interfere with the secret high input(s) and this makes more sense compared to qualitative information analysis. Quantitative information flow analysis is what this thesis is focused on.

Given any deterministic program, once the input is fixed, the output can be determined. Thus the deterministic program can be viewed as a function transforming its input into its output. Since the input of a program can take many values according to a probability distribution, similarly, the program can also be viewed as a function transforming the distribution of input into the distribution of its output. In the context of quantitative information flow, probability distributions are used by Shannon's entropy definition to represent information. As the public input of the program is usually known, it is considered to contain no information (with probability mass one). Therefore, once the information contained in the secret input of the program and that in the output of the program are accounted for respectively, the difference between them is the information leaked by the program and this is the basic idea of quantitative information flow analysis.

### 1.1.1 Information Leakage Definition

The first step towards quantitative information flow analysis is the definition of information leakage. There have been quite a few versions of definitions [25, 11, 51, 57, 43, 60, 4]. Of them, some use Shannon's information theory to define the leakage [25, 11], some use learning theory [57], some use process algebra [4] and some use differences in processes [60].

The idea of applying Shannon's information theory to quantify information leakage is not new, as it was pioneered by Denning in the 1970's [24, 25, 27, 26]. Later, Millen [51] built the first formal correspondence between non-interference and mutual information, while in [50], McLean introduced *time* into the analysis. Clark, Hunt and Malacaria (CHM) [15, 16, 14, 12, 11] use conditional entropy to correct Denning's measure of information leakage (please refer to Page 50 for the deficiencies of Denning"s definition). Approaching from a different angle, recently, Clarkson, Myers and Schneider have proposed to quantify information flow using beliefs [57]. Lowe's definition of information flow [43] uses the process algebra CSP, and is based upon counting the number of different behaviors of a high level user that can be distinguished by a low level user.

A simple example (direct flow) to show how to actually calculate the leakage using CHM's definition is as follows:

**Example 1.1.1.** $L := H$

Suppose $H$ is an uniformly distributed 4 bits variable, before the execution of this program, the low variable $L$ contains 0 bits of information, after the assignment it contains all 4 bits of $H$, hence there is an information leakage of 4 bits.

The reader should note that unlike non-interference, there isn't a *standard* definition of information leakage in the information flow community, therefore all the work mentioned above argues the validity and suitability for their own definition of information leakage.

The next step is to develop a framework in which quantitative information analysis can be readily automated. Such a framework should be practical, sound, as accurate as possible and hopefully widely applicable. However, not much effort has been put into the development of such a systematic framework. The only work on such a framework, to the best of our knowledge, is Clark, Hunt and Malacaria's work [12, 11]. Their framework is based on a set of inference rules defined for different commands and operations which can be used to deduce an upper bound of information leakage after the command's execution. Details and examples are discussed in Chapter 3.

We must point out that (quantitative) flow analysis is *not* complete, and this can be illustrated by a straightforward example:

**Example 1.1.2.** $P; Y := X$

where $P$ is any program not involving $X$, and $X$ is a high variable and $Y$ is a low variable.

The amount of information leaked into $Y$ is larger than 0 iff $P$ terminates. Thus a complete flow-analyzer would allow us to solve the halting problem.

Hence our work is *not* trying to make (quantitative) information flow analysis complete. Instead, our work takes CHM's work as a starting point and tries to improve the accuracy of their framework while making sure it is still sound.

The deficiency of CHM's framework is that they analyze high variables independently no matter what relationship may exist between them. The basic idea of their analysis when it tries to determine an upper bound for the total information leakage is just to add all the source information together, no matter whether there is any duplicated information or not. A simple example can demonstrate the problem:

**Example 1.1.3.**

$$X := Y;$$

$$Z := X + Y$$

After $X := Y$ , $X$ and $Y$ are actually equivalent to each other; they contain exactly the same information as each other. Hence $X + Y$ should just contain as much information as either $X$ or $Y$. However, CHM's work analyzes that $Z$ contains double information of $X$ ( or $Y$) by simply adding the information of $X$ and $Y$ together which are exactly the same as each other, hence the problem is given the name of "double counting". Examples and discussion of the "double counting" problem are in Chapter 3.

In our work, the maximum information of each individual high variable is known, and the problem of CHM's framework is improved by taking linear constraints (captured by abstract interpretation) among high program variables into account (in the simple example above, it is $X == Y$), and transforms them into a constraint in our problem. Our idea is that the total amount of information that can be leaked is the *joint entropy* of all the high program variables, instead of the sum of the information of all the high variables, this can be worked out as a constrained optimization problem by applying the Lagrange multiplier method. In the simple example above, the joint entropy $\mathcal{H}(X, Y)$ of $X$ and $Y$ is the same as either $\mathcal{H}(X)$ or $\mathcal{H}(Y)$ given that $X == Y$. More details are in Chapter 5. This piece of construction is the effort to make the original framework of CHM's more accurate as the relationships between program variables are taken into account.

The result of our work is a quantity which is the maximum joint entropy of linearly related program high variables. Although the aim of this thesis is *not* to develop a specific information flow policy, our result can be useful to help guide the design of such policy. For example, if when executing a program, it always leaks the maximum amount of information, then the policy can certainly rule it as insecure.

## 1.2 Abstract Interpretation for Linear Constraints Detection

As we mentioned before, abstract interpretation is used for automatic linear constraints detection in our work.

The theory of abstract interpretation was developed by Patrick and Radhia Cousot in the late 1970s [17]. An abstract interpretation is defined as a non-standard (approximated) program semantics obtained from the concrete one by replacing the concrete domain of computations and its concrete semantic operations with an abstract domain and corresponding abstract semantic operations respectively. It is applied to program source code to infer an approximation of the program's run-time behavior by carrying out a static analysis. The most important issue about abstract interpretation is its soundness, which means that if a property holds in the concrete program, it should also gets hold by the abstract interpretation of the program.

Let's take the example of parity to see how abstract interpretation works. In this case, the concrete domain is the set of integers and the abstract domain is the set of $\{0, 1\}$.

All the even numbers are abstracted as 0 and all the odd numbers as 1, the abstract semantics would be

$$0 \pm^\sharp 0 = 0;\ 1 \pm^\sharp 1 = 0;\ 1 \pm^\sharp 0 = 0 \pm^\sharp 1 = 1$$
$$0 \times^\sharp 0 = 0;\ 1 \times^\sharp 1 = 1;\ 0 \times^\sharp 1 = 1 \times^\sharp 0 = 0$$

where $\pm^\sharp$ denotes the abstract semantic operations of either $+$ or $-$. For example, if the program contains just one single assignment: $X := X * 2$, then the abstract analysis should give us the final result of $X$ to be 0 because the abstract semantics gives us both $0 \times^\sharp 0 = 0$ and $1 \times^\sharp 0 = 0$ .

The application of abstract interpretation for automatically capturing linear constraints was pioneered by Cousot and Halbwachs [22]. In this case, the concrete domain is the set of stores (mapping from program variables to integers) and the abstract domain is the set of linear constraints over the program variables. Each kind of command,*i.e.* assignment, **if** statement, **while** loop, has a particular abstract specification of how to derive the linear constraint(s) after the command's execution given the input linear constraints. For example, suppose the input assertion to an assignment is defined by

$$\begin{cases} X_2 \geq 1, \\ X_1 + X_2 \geq 5, \\ X_1 - X_2 \geq -1 \end{cases}$$

If the assignment is $X_2 := X_1 + 1$, the abstract interpretation of deriving linear constraints for this type of assignment is to eliminates $X_2$ in the input constraints and adjoin the assignment itself in the resulting constraints, in more detail, if we substitute $X_2$ with $X_1 + 1$ in each of the above three inequalities, we get $X_1 \geq 2$, together with the reformulated original assignment $X_2 - X_1 = 1$, we get the following constraints after the assignment:

$$\begin{cases} X_1 \geq 2, \\ X_2 - X_1 = 1 \end{cases}$$

More details about it and other examples are presented in Chapter 3.

## 1.3 Scope and Contributions

In this section, the contribution of our work is highlighted, and the scope and limitation of our work is discussed.

### 1.3.1 Scope

The core of this thesis is an improvement of the quantitative information analysis framework originally proposed by Clark, Hunt and Malacaria [12, 11]. By comparing our work with theirs we show that our method can make the analysis more precise. Also it can be built into the original analysis to reason about programs written in the simple while language, and it should also be able to apply to real languages such as Java or C++. However, it has *not* been applied to large pre-existing programs (thousands of lines).

Our analysis is static, as opposed to [48, 49], which is dynamic and gives an upper bound of information leakage of one particular execution under a specified input provided by the user on pre-existing programs. Their upper bound is only valid for that particular execution, and it is possible that another run of the program will leak more information than this. By contrast, our result is an universal upper bound, which is suitable for any input and round of execution of the program.

Our work is useful for systems that satisfy some information flow policy, *i.e.* allow certain portion of secret data to flow within the system.

However, please note that information leakage from other aspects are not our concern, *i.e.* memory allocation, termination, running time and etc.

Also the type of linear constraints that we can make use of is a linear inequality constraint (*e.g.* $X \leq Y$), non-linear constraints between program variables cannot be dealt with in [22], hence out of the scope of our work as well.

### 1.3.2 Contribution

The main contributions of this thesis are as follows:

- We integrate abstract interpretation into quantitative information flow analysis. More precisely, the linear constraints between program variables are explored and made advantage of to improve the precision of the quantitative information flow analysis.

- We propose to use joint entropy of high variables to represent the maximum amount of information that a program can leak.

- We are able to provide an analytical formula of how to construct the input probabilistic distribution that corresponds to the maximum amount of the information leakage, given linear constraints and non-linear constraints. Although our work can be put into the area of convex optimization, however, except least-squares problems and linear programs, it is widely accepted that there is in general no analytical formula for the solution of convex optimization. Thus, our result is both encouraging and promising.

- We extend the application of the Lagrange multiplier method to the field of quantitative information flow, especially to the derivation of an input probability distribution that has the largest possible information leakage under given marginal entropy constraints. Although the idea of the application of the Lagrange multiplier method has been introduced in [45, 10, 9, 8], the only constraint they consider is a simplex constraint: *i.e.* all the probabilities of an input sum up to one, thus the input probability which can give rise to the maximum information leakage is the uniform distribution, which is a very basic (and very old) result in information theory. In contrast, we use the Lagrange multiplier method to solve more complicated non-linear constraints (marginal entropy).

## 1.4 Organization of the Thesis

The reminder of this thesis is structured as follows:

- Chapters 2 to 4 constitute a reference to the underlying mathematical background and the inspiration of our research.
  - Chapter 2 is the introduction of Shannon's information theory with simple examples. The definitions of different entropies, mutual information and chain rules are introduced, the relationship among them is explored. These are the foundation of quantitative information analysis as the information leakage calculation is defined using one of the entropies (*e.g.* conditional entropy). Moreover, as the joint entropy represents the sum of the information that high variables can have, this is the total amount of information that can ever be leaked, so in our work we try to place an upper bound on this quantity.
  - Chapter 3 reviews the developments of quantitative information flow and abstract interpretation respectively. In particular, the disadvantage of the current framework of CHM's is illustrated by a straightforward example and a possible solution, by taking into

account linear constraints among programs, is proposed. Since linear constraints can be nicely captured by abstract interpretation, hence our research idea is to make use of it.

- Chapter 4 contains two parts: the first part is the main technique we used in our work that is the Lagrange multiplier method, and the theory and examples showing its application; the second part is the introduction of convex optimization, since mathematically the problem we try to solve belongs to this category. However, the speciality of our problem is pointed out and discussed.

- Chapters 5 and 6 constitute the main body of this thesis.

  - Chapter 5 discusses the maximization of joint entropy under one single entropy constraint, together with linear constraint(s) between program variables. We show how the partition version of entropy definition can be used to facilitate the derivation of the Lagrange multiplier method; and finally a rigorous mathematical formula is derived which constructs the probability distribution that maximizes the joint entropy under those constraints. Moreover, we show that the problem can be boiled down to search for a suitable $\alpha$, which is a parameter taking positive real numbers.

  - Chapter 6 extends Chapter 5 by considering two marginal entropy constraints. The Lagrange multiplier method is still used to deduce an analytical form for constructing the joint probability distribution which gives the maximal joint entropy. The problem is further broken down to search for feasible marginal probability distributions which satisfy marginal entropy constraints, which can be further broken down to search for a pair of parameters $\alpha$ and $\beta$. In addition, we identify situations where there is no corresponding $\alpha$ or $\beta$, and these are usually when one of the marginal entropies (or both) hits its (their) maximum. We use examples to compare our result with that of the original CHM's framework.

- Chapter 7 concludes and provides some suggestions for future investigations.

# Chapter 2

# Background of Information Theory

This chapter covers some basic notions of probability distribution, information theory and the Lagrange multiplier method. All of them are very broad topics, especially information theory, so we only review the most relevant concepts to this thesis. We introduce definitions of different entropies and their relationship between each other, in more detail.

## 2.1 Probability

A discrete random variable takes values from a countable set of specific values, each with some probability greater than zero; a continuous random variable takes values from an uncountable set, and the probability of any subset of values is positive.

The classical definition of the probability of an event occurring is defined as the number of favorable cases for the event, over the number of total outcomes possible in an equiprobable sample space. The formal definition of discrete probability is:

**Definition 2.1.1.** *Suppose a random variable $X$ which can take all its possible values in a sample space $\Omega = \{x_1, x_2, \dots, x_n\}$ in the classical sense, then for each $x \in \Omega$, a probability value $p(x)$ is attached, which satisfies the following properties [23]:*

- $p(x) \in [0, 1]$ *for all* $x \in \Omega$
- $\sum_{x \in \Omega} p(x) = 1$

Thus, probability is a value between 0 and 1, and $p(x)$ sums up to 1 over all the values in the sample space. An event is defined as any subset $S$ of the sample space $\Omega$. The probability of the event $S$ is defined as

$$P(S) = \sum_{x \in S} p(x)$$

The function $p(x)$ mapping a point in the sample space to the probability value is called a probability mass function.

When the sample space is continuous (this is just for completeness, we do not consider continuous probability in this thesis), the probability is defined as:

**Definition 2.1.2.** *If the sample space of a random variable $X$ is the set of real numbers $R$ or a subset, then a function called the cumulative distribution function $f$ exists, defined as $f(x) = p(X \leq x)$. That is, $f(x)$ returns the* *probability that $X$ will be less than or equal to $x$. And it also satisfies the following [23]:*

- *$f$ is a monotonically non-decreasing, right-continuous function;*
- $\lim_{x \to -\inf} f(x) = 0$
- $\lim_{x \to \inf} f(x) = 1$

Thus, the probability that $X$ is between two points $a$ and $b$ is:

$$f[a \leq x \leq b] = \int_a^b f(x)dx$$

and

$$\int_{-\inf}^{\inf} f(x) = 1$$

A probability distribution identifies either the probability of each value of an unidentified random variable (when the random variable is discrete), or the probability of the value falling within a particular interval (when the variable is continuous).

There are many well known discrete probability distributions, *e.g.* Bernoulli distributions, binomial distribution, uniform distribution, geometric distribution; continuous distributions are logarithmic distribution, Gaussian distribution, continuous uniform distribution, etc. As these distributions are not the focus of this thesis, here we just introduce the discrete uniform distribution which is most commonly used:

Uniform distribution assigns equal probability to each outcome of the sample space $\Omega = \{x_1, x_2, \ldots, x_n\}$, namely:

$$\forall x \in \Omega, p(x) = \frac{1}{n}$$

## 2.2 Function

In mathematics a function is a relation between a given set of elements (the domain) and another set of elements (the codomain), which associates each element in the domain with exactly one element in the codomain.

A function $f : X \to Y$ is surjective (onto) if and only if for every $y$ in the codomain Y there is at least one $x$ in the domain $X$ such that $f(x) = y$.

A function is one-to-one if every element of its codomain is mapped to by at most one element of its domain.

There are two types of functions that are of most interest to us, namely convex and concave functions:

### 2.2.1 Convex Function

**Definition 2.2.1.** *A real-valued function $f$ defined on an interval is called convex, if for any two points $x$ and $y$ in its domain $D$ and any $t$ in $[0, 1]$, the following condition holds [3]:*

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y)$$

Pictorially, a function is called convex if the function lies below the straight line segment connecting two points, for any two points in the interval, refer to Figure 2.1:

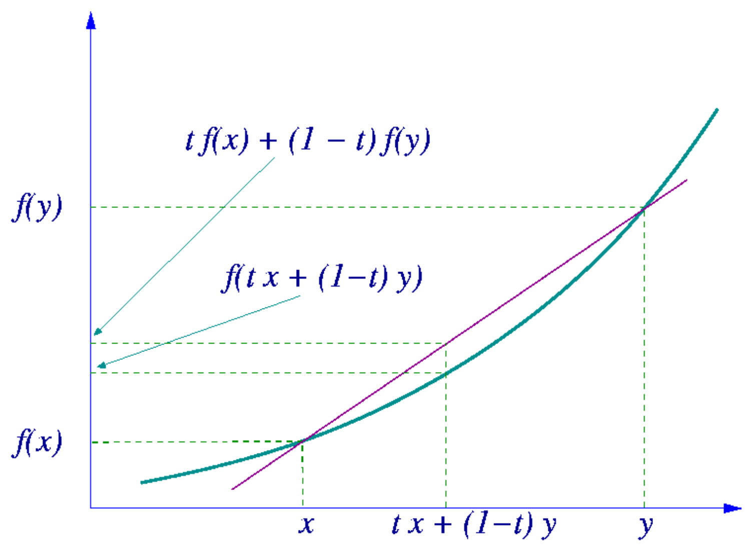

Figure 2.1: Example of Convex Function

A function is called strictly convex if

$$f(tx + (1-t)y) < tf(x) + (1-t)f(y)$$

for any $t \in (0,1)$ and $x \neq y$.

A function is midpoint convex on an interval $D$ if

$$f\left(\frac{x+y}{2}\right) \leq \frac{f(x)+f(y)}{2}$$

for all $x$ and $y$ in $D$.

This condition is only slightly weaker than convexity. However, a continuous function that is midpoint convex will be convex.

A differentiable function of one variable is convex on an interval if and only if its derivative is monotonically non-decreasing on that interval.

A continuously differentiable function of one variable is convex on an interval if and only if the function lies above all of its tangents: $f(y) \geq f(x) + f'(x)(y - x)$ for all $x$ and $y$ in the interval.

A twice differentiable function of one variable is convex on an interval if and only if its second derivative is non-negative; this gives a practical test for convexity. If the second derivative is positive then it is strictly convex, but the converse does not hold.

Examples of convex functions include $x^2$, $|x|$, $e^x$, $x \log x$(for $x \geq 0$), *etc.*

### 2.2.2 Concave Function

Basically, a function $f$ is concave if $-f$ is convex. Formally,

**Definition 2.2.2.** *A real-valued function $f$ defined on an interval is called concave, if for any two points $x$ and $y$ in its domain $D$ and any $t$ in $[0, 1]$, the following holds [3]:*

$$f(tx + (1 - t)y) \geq tf(x) + (1 - t)f(y)$$

Pictorially, a function is called concave if the function lies above the straight line segment connecting two points, for any two points in the interval, refer to Figure 2.2:

Similarly, a function is called strictly concave if

$$f(tx + (1 - t)y) > tf(x) + (1 - t)f(y)$$

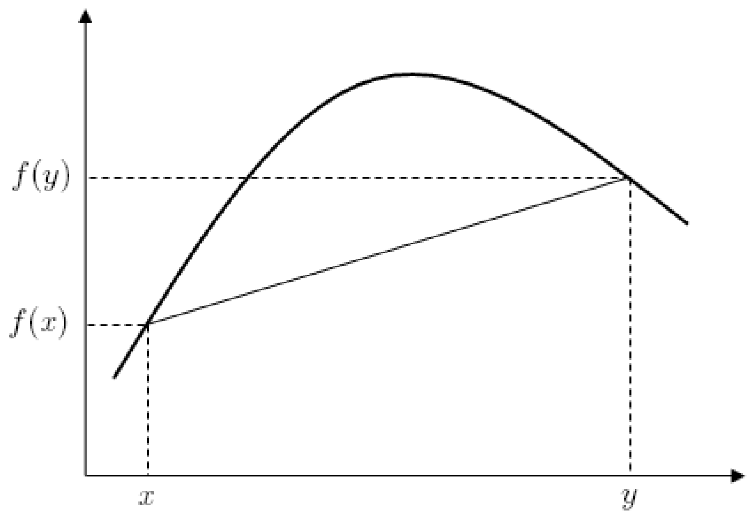

Figure 2.2: Example of Concave Function

for any $t$ in $(0,1)$ and $x \neq y$.

A continuous function on an interval $D$ is concave if and only if

$$f\left(\frac{x+y}{2}\right) \geq \frac{f(x)+f(y)}{2}$$

for any $x$ and $y$ in $D$.

A differentiable function $f$ is concave on an interval if its derivative is monotonically decreasing on that interval.

A twice differentiable function of one variable is concave on an interval if and only if its second derivative is negative.

Examples of concave functions include $\log x$ and $\sqrt{x}$ for $x \geq 0$.

## 2.3 Partition

The mathematical definition of partition is as follows:

**Definition 2.3.1.** *Let $X$ be any set. $\mathcal{P} \subseteq \wp(X)$ (power set of $X$) is a partition of $X$ if*

- $\bigcup_{P \in \mathcal{P}} P = X$
- *for all* $P_1,\ P_2 \in \mathcal{P}$ *either* $P_1 = P_2$ *or* $P_1 \cap P_2 = \emptyset$

## 2.4 Information Theory

One of the most important feature of Shannon's theory is the concept of entropy, which lays the foundation for information theory [67]. In this section, we will start from the very fundamental concept of entropy which measures the uncertainty of a random variable. Then the definition of joint entropy and conditional entropy which are extended notions of entropy are introduced. Joint entropy measures the uncertainty of two or more variables in combination while conditional entropy measures the remaining uncertainty of one variable when knowing the other. Moreover, mutual information and relative entropy are introduced based on joint entropy and conditional entropy. Mutual information measures the amount of information one random variable contains about the other. Relative entropy represents the difference in the probability distributions between two random variables. All these concepts are closely related, and we show how they can be deduced from each other.

### 2.4.1 Entropy

In information theory, entropy is used to measure the uncertainty associated with a random variable.

**Definition 2.4.1.** *Given a discrete random variable $X$ whose values are in the set $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$, entropy is defined as [67]*

$$\mathcal{H}(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x)$$

where $p(x)$ is the probability mass function of random variable $X$ taking value $x$. The log is to the base 2 and entropy is expressed in bits. The convention is that $0 \log 0 = 0$ and $1 \log 1 = 0$ which can be justified as neither 'zero' probability (something guaranteed not to happen) or 'one' probability (something guaranteed to happen) has any uncertainty. Also, it is very clear from the definition that entropy doesn't depend on the actual value that random variable $X$ may take, it only depends on the probability distributions of $X$. Furthermore, entropy is a permutation invariant of probability distribution, in other words, once the probability distribution is fixed, the entropy is fixed, no matter which possible value of $X$ takes which probability. Also entropy is always non-negative.

Note that the expectation of $X$ is $EX = \sum_{x \in \mathcal{X}} xp(x)$, if interpreted in this way, the entropy of $X$ can also be viewed as the expected value of the random variable $-\log p(X)$. Thus,

$$\mathcal{H}(X) = EP[-\log p(X)]$$

**Example 2.4.2.** *Let*

$$X = \begin{cases} a & \text{with probability } \frac{1}{2} \\ b & \text{with probability } \frac{1}{4} \\ c & \text{with probability } \frac{1}{8} \\ d & \text{with probability } \frac{1}{8} \end{cases}$$

*The entropy of $X$ is*

$$\mathcal{H}(X) = -\frac{1}{2}\log\frac{1}{2} - \frac{1}{4}\log\frac{1}{4} - \frac{1}{8}\log\frac{1}{8} - -\frac{1}{8}\log\frac{1}{8} = 1.75 \text{ bits}$$

**Example 2.4.3.** *The entropy of a discrete random variable $X$ which can take $n$ possible values achieves its maximum when its probabilistic distribution is uniform:*

$$\mathcal{H}(X) = \log n$$

We will prove the above conclusion using Lagrange multiplier method in Chapter 4.

The entropy function is concave, Figure 2.3 demonstrates this property using a binary random variable.

### 2.4.2 Joint Entropy

Joint entropy can be viewed as an extension of entropy of a single random variable, as a pair of random variables $(X, Y)$ (or more) can be considered to be a single vector-valued random variable.

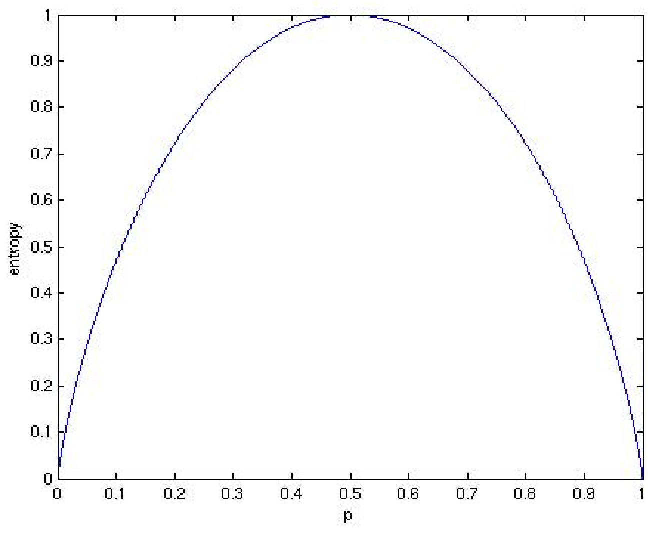

Figure 2.3: Concaveness of Entropy Function

**Definition 2.4.4.** *The joint entropy $\mathcal{H}(X, Y)$ of a pair of discrete random variables $(X, Y)$ with a joint distribution $p(x, y)$ where $X$ taking values in the set $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ and $Y$ taking values in the set $\mathcal{Y} = \{y_1, y_2, \ldots, y_n\}$ respectively, is defined as [23]*

$$\mathcal{H}(X, Y) = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log p(x, y)$$

The expectation version of joint entropy can also be expressed as

$$\mathcal{H}(X, Y) = E[-\log p(X, Y)]$$

**Example 2.4.5.** *Suppose a pair of discrete random variables $(X, Y)$ have the following joint distribution as shown in Table 2.1, then the joint entropy*

| Y \ X | $x_0$ | $x_1$ | $x_2$ |
|---|---|---|---|
| $y_0$ | 1/6 | 1/9 | 1/18 |
| $y_1$ | 1/9 | 1/9 | 1/9 |
| $y_2$ | 1/3 | 0 | 0 |

Table 2.1: Joint Probabilistic Distribution

*$\mathcal{H}(X,Y)$ is:*

$$\begin{aligned}\mathcal{H}(X,Y) &= -\textstyle\sum_x \sum_y p(x,y)\log p(x,y)\\ &= -\tfrac{1}{6}\log\tfrac{1}{6} - \tfrac{4}{9}\log\tfrac{1}{9} - \tfrac{1}{18}\log\tfrac{1}{18} - \tfrac{1}{3}\log\tfrac{1}{3}\\ &= 1.9641 \text{ bits}\end{aligned}$$

### 2.4.3 Conditional Entropy

Conditional entropy measures the remaining uncertainty of a random variable given that another related variable $Y$ is known.

**Definition 2.4.6.** *Suppose two random variables $X$ and $Y$ with $X$ taking values in the set $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ and $Y$ taking values in the set $\mathcal{Y} = \{y_1, y_2, \ldots, y_n\}$ respectively, if the conditional probability is denoted as $p(x|y)$, then the conditional entropy $\mathcal{H}(X|Y)$ is defined as [23]*

$$\begin{aligned}\mathcal{H}(X|Y) &= \textstyle\sum_{y\in\mathcal{Y}} p(y)\mathcal{H}(X|Y=y)\\ &= -\textstyle\sum_{y\in\mathcal{Y}} p(y)\sum_{x\in\mathcal{X}} p(x|y)\log p(x|y)\\ &= -\textstyle\sum_{x\in\mathcal{X},y\in\mathcal{Y}} p(x,y)\log p(x|y)\end{aligned}$$

**Example 2.4.7.** *Suppose the joint probabilistic distribution is still the one shown in Table 2.1, then the marginal probabilities for $X$ and $Y$ are $\{\frac{11}{18}, \frac{2}{9}, \frac{1}{6}\}$*

*and* $\{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\}$*, the conditional entropy of* $\mathcal{H}(X|Y)$ *is:*

$$\begin{aligned}\mathcal{H}(X|Y) &= -\textstyle\sum_{x,\,y} p(x,y)\log p(x|y)\\ &= -\tfrac{1}{6}\log\tfrac{1}{2} - \tfrac{4}{9}\log\tfrac{1}{3} - \tfrac{1}{18}\log\tfrac{1}{6} - \tfrac{1}{3}\log 1\\ &= 1.0147 \text{ bits}\end{aligned}$$

*similarly the conditional entropy of* $\mathcal{H}(Y|X)$ *is:*

$$\begin{aligned}\mathcal{H}(Y|X) &= -\textstyle\sum_{x,\,y} p(x,y)\log p(y|x)\\ &= -\tfrac{1}{6}\log\tfrac{3}{11} - \tfrac{1}{9}\log\tfrac{2}{11} - \tfrac{1}{3}\log\tfrac{6}{11} - \tfrac{2}{9}\log\tfrac{1}{2} - \tfrac{1}{18}\log\tfrac{1}{3} - \tfrac{1}{9}\log\tfrac{2}{3}\\ &= 1.2524 \text{ bits}\end{aligned}$$

It is very clear from this example that

$$\mathcal{H}(X|Y) \neq \mathcal{H}(Y|X)$$

which means in general the uncertainty left for $X$ knowing $Y$ is not the same as the other way around.

Both joint entropy and conditional entropy measures some kind of uncertainty between two random variables, naturally one may wonder if there is any relationship between them, and this is proved by the following theorem:

**Theorem 2.4.8** (Chain Rule)**.**

$$\mathcal{H}(X,Y) = \mathcal{H}(X) + \mathcal{H}(Y|X)$$

$$\mathcal{H}(X,Y) = \mathcal{H}(Y) + \mathcal{H}(X|Y)$$

*Proof.* We prove $\mathcal{H}(X,Y) = \mathcal{H}(Y) + \mathcal{H}(X|Y)$, and the other can be proved in the same way.

$$\begin{aligned}
\mathcal{H}(X,Y) &= -\textstyle\sum_{x\in X}\sum_{y\in Y} p(x,y)\log p(x,y) \\
&= -\textstyle\sum_{x\in X}\sum_{y\in Y} p(x,y)\log p(y)p(x|y) \\
&= -\textstyle\sum_{x\in X}\sum_{y\in Y} p(x,y)\log p(y) - \sum_{x\in X}\sum_{y\in Y} p(x,y)\log p(x|y) \\
&= -\textstyle\sum_{y\in Y} p(y)\log p(y) - \sum_{x\in X}\sum_{y\in Y} p(x,y)\log p(x|y) \\
&= \mathcal{H}(Y) + \mathcal{H}(X|Y)
\end{aligned}$$

$\square$

### 2.4.4 Relative Entropy

The relative entropy is a measure of the distance between assuming probabilistic distribution $q$ while the true distribution is $p$.

**Definition 2.4.9.** *The relative entropy between two probability mass functions $p(x)$ and $q(x)$ is defined as [23]*

$$D(p||q) = \sum_{x\in\mathcal{X}} p(x)\log\frac{p(x)}{q(x)}$$

In the above definition, we use the convention that $0\log\frac{0}{0} = 0$, $0\log\frac{0}{q} = 0$ and $p\log\frac{p}{0} = \infty$. Thus, if there is any symbol $x \in \mathcal{X}$ such that $p(x) > 0$ and $q(x) = 0$, then $D(p||q) = \infty$. Relative entropy is always non-negative, and is zero if and only if $p = q$. However, it is not symmetric and does not satisfy the triangle inequality. Nonetheless, it is often useful to think of relative entropy as a "distance" between distributions.

**Example 2.4.10.** *Let X={0, 1} and consider two distributions p and q on $\mathcal{X}$. Let $p(0) = 1 - r$, $p(1) = r$ and let $q(0) = 1 - s$, $q(1) = s$. Then*

$$D(p||q) = (1-r)\log\frac{1-r}{1-s} + r\log\frac{r}{s}$$

*and*

$$D(q||p) = (1-s)\log\frac{1-s}{1-r} + s\log\frac{s}{r}$$

*If $r = \frac{1}{2}, s = \frac{1}{4}$, we can calculate*

$$D(p||q) = \frac{1}{2}\log\frac{\frac{1}{2}}{\frac{3}{4}} + \frac{1}{2}\log\frac{\frac{1}{2}}{\frac{1}{4}} = 0.2075 \text{ bits}$$

*whereas*

$$D(p||q) = \frac{3}{4}\log\frac{\frac{3}{4}}{\frac{1}{2}} + \frac{1}{4}\log\frac{\frac{1}{4}}{\frac{1}{2}} = 0.1887 \text{ bits}$$

## 2.4.5 Mutual Information

Mutual information is another important concept in information theory. It measures the uncertainty shared between two random variables. It can also be viewed as the relative entropy between the joint distribution $p(x, y)$ and the product distribution $p(x)p(y)$.

**Definition 2.4.11.** *Suppose two random variables $X$ and $Y$ which taking values in sets $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ and $\mathcal{Y} = \{y_1, y_2, \ldots, y_n\}$ respectively and associated with a joint probability mass function $p(x, y)$ and marginal probability mass functions $p(x)$ and $p(y)$. The mutual information $\mathcal{I}(X; Y)$*

*is defined as [23]:*

$$\begin{aligned}\mathcal{I}(X;Y) &= \textstyle\sum_{x\in X}\sum_{y\in Y} p(x,y)\log\frac{p(x,y)}{p(x)p(y)}\\ &= D(p(x,y)||p(x)p(y))\end{aligned}$$

Note that if $X$ and $Y$ are independent from each other, then

$$p(x,y) = p(x)p(y)$$

In this case $\log\frac{p(x,y)}{p(x)p(y)} = \log 1 = 0$, therefore, mutual information $\mathcal{I}(X;Y) = 0$. This makes sense as independent random variables don't share any information between each other.

**Theorem 2.4.12.** *The relationship between mutual information and entropy is:*

$$\mathcal{I}(X;Y) = \mathcal{H}(X) - \mathcal{H}(X|Y)$$

*Proof.*

$$\begin{aligned}\mathcal{I}(X;Y) &= \textstyle\sum_{x,\,y} p(x,y)\log\frac{p(x,y)}{p(x)p(y)}\\ &= \textstyle\sum_{x,\,y} p(x,y)\log\frac{p(x|y)}{p(x)}\\ &= -\textstyle\sum_{x,\,y} p(x,y)\log p(x) + \sum_{x,\,y} p(x,y)\log p(x|y)\\ &= -\textstyle\sum_{x} p(x)\log p(x) - \Big(-\sum_{x,\,y} p(x,y)\log p(x|y)\Big)\\ &= \mathcal{H}(X) - \mathcal{H}(X|Y)\end{aligned}$$

□

Thus, the mutual information $\mathcal{I}(X;Y)$ is the reduction in the uncertainty of $X$ due to the knowledge of $Y$. By symmetry, it also follows that

$$\mathcal{I}(X;Y) = \mathcal{H}(Y) - \mathcal{H}(Y|X)$$

Thus, $X$ contains as much uncertainty about $Y$ as $Y$ contains about $X$, in other words,

$$\mathcal{I}(X;Y) = \mathcal{I}(Y;X)$$

*Proof.*

$$\begin{aligned}
\mathcal{I}(X;Y) &= \mathcal{H}(Y) - \mathcal{H}(Y|X) \\
&= \mathcal{H}(Y) - \mathcal{H}(X,Y) + \mathcal{H}(X) \\
&= \mathcal{H}(X) + \mathcal{H}(Y) - \mathcal{H}(X,Y) \\
&= \mathcal{H}(X) - \mathcal{H}(X|Y) \\
&= \mathcal{I}(Y;X)
\end{aligned}$$

□

Since $\mathcal{H}(X,Y) = \mathcal{H}(X) + \mathcal{H}(Y|X)$, as shown in Theorem 2.4.8, we have

$$\mathcal{I}(X;Y) = \mathcal{H}(X) + \mathcal{H}(Y) - \mathcal{H}(X,Y)$$

Collecting the above results, the following theorem holds:

**Theorem 2.4.13** (Mutual Information and Entropy)**.**

$$\begin{aligned}
\mathcal{I}(X;Y) &= \mathcal{H}(X) - \mathcal{H}(X|Y) \\
\mathcal{I}(X;Y) &= \mathcal{H}(Y) - \mathcal{H}(Y|X) \\
\mathcal{I}(X;Y) &= \mathcal{H}(X) + \mathcal{H}(Y) - \mathcal{H}(X,Y) \\
\mathcal{I}(X;Y) &= \mathcal{I}(Y;X) \\
\mathcal{I}(X;X) &= \mathcal{H}(X)
\end{aligned}$$

Figure 2.4 shows the relationship between entropy and mutual information more clearly.

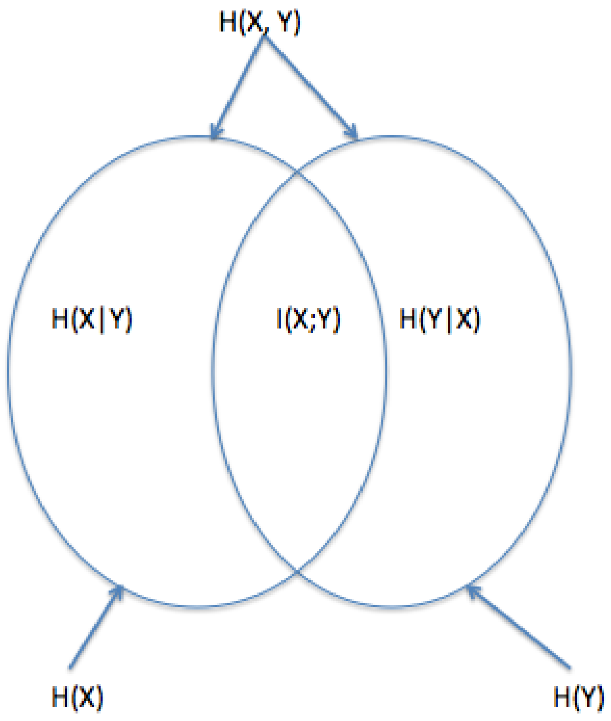

Figure 2.4: Relationship between entropy and mutual information

**Example 2.4.14.** *Suppose the joint probabilistic distribution is still the one shown in* Table *2.1, from the computation of conditional entropy, we know that* $\mathcal{H}(X|Y) = 1.0147\ \text{ bits}$, *and* $\mathcal{H}(Y|X) = 1.2524\ \text{ bits}$. *The marginal entropy for* $X$ *and* $Y$ *are* $\{\frac{11}{18},\ \frac{2}{9},\ \frac{1}{6}\}$ *and* $\{\frac{1}{3},\ \frac{1}{3},\ \frac{1}{3}\}$. *Then*

$$\mathcal{H}(X) = \mathcal{H}(\tfrac{11}{18},\ \tfrac{2}{9},\ \tfrac{1}{6}) = 1.3472 \text{ bits}$$
$$\mathcal{H}(Y) = \mathcal{H}(\tfrac{1}{3},\ \tfrac{1}{3},\ \tfrac{1}{3}) = 1.5847 \text{ bits}$$

*Thus,*

$$\begin{aligned}\mathcal{I}(X;Y) &= \mathcal{H}(X) - \mathcal{H}(X|Y)\\ &= 1.347223 - 1.014703\\ &= \mathcal{H}(Y) - \mathcal{H}(Y|X)\\ &= 1.584963 - 1.252443\\ &= 0.3325 \text{ bits}\end{aligned}$$

Like conditional entropy, there is conditional mutual information whose definition is as follows:

**Definition 2.4.15.** *The conditional mutual information of random variables* $X$ *and* $Y$ *given* $Z$ *is defined as [23]*

$$\begin{aligned}\mathcal{I}(X;Y|Z) &= \mathcal{H}(X|Z) - \mathcal{H}(X|Y,Z)\\ &= \textstyle\sum_{x,\ y,\ z} p(x,y,z) \log \frac{p(x,y|z)}{p(x|z)p(y|z)}\end{aligned}$$

### 2.4.6 Chain Rules for Joint Entropy, Relative Entropy and Mutual Information

It is interesting to see that the joint entropy of a collection of random variables is the sum of the conditional entropies.

**Theorem 2.4.16** (Chain Rule for Entropy). *Let $X_1, X_2, \ldots, X_n$ be drawn according to $p(x_1, x_2, \ldots, x_n)$. Then*

$$\mathcal{H}(X_1, X_2, \ldots, X_n) = \sum_{i=1}^{n} \mathcal{H}(X_i|X_{i-1}, \ldots, X_1)$$

*Proof.* Applying the mathematical induction:

Base case - when $n = 2$:

$$\mathcal{H}(X_1, X_2) = \mathcal{H}(X_1) + \mathcal{H}(X_2|X_1) \text{ holds}$$

Suppose that when $n = k$ it also holds, namely:

$$\mathcal{H}(X_1, X_2, \ldots, X_k) = \sum_{i=1}^{k} \mathcal{H}(X_i|X_{i-1}, \ldots, X_1)$$

When n=k+1:

$$\begin{aligned}
\mathcal{H}(X_1, X_2, \ldots, X_n) &= \mathcal{H}(X_1, X_2, \ldots, X_k, X_{k+1}) \\
&= \mathcal{H}(X_1, X_2, \ldots, X_k) + \mathcal{H}(X_{k+1}|X_1, X_2, \ldots, X_k) \\
&= \textstyle\sum_{i=1}^{k} \mathcal{H}(X_i|X_{i-1}, \ldots, X_1) + \mathcal{H}(X_{k+1}|X_1, X_2, \ldots, X_k) \\
&= \textstyle\sum_{i=1}^{k+1} \mathcal{H}(X_i|X_{i-1}, \ldots, X_1)
\end{aligned}$$

It also holds when $n = k + 1$, thus, it holds for all $n$. □

There is also a chain rule for mutual information.

**Theorem 2.4.17** (Chain Rule for Information)**.** *Let $X_1, X_2, \ldots, X_n$ and $Y$ all be random variables, then*

$$\mathcal{I}(X_1, X_2, \ldots, X_n; Y) = \sum_{i=1}^{n} \mathcal{I}(X_i; Y | X_{i-1}, X_{i-2}, \ldots, X_1)$$

*Proof.*

$$\begin{aligned}
\mathcal{I}(X_1, X_2, \ldots, X_n; Y) &= \mathcal{H}(X_1, X_2, \ldots, X_n) - \mathcal{H}(X_1, X_2, \ldots, X_n | Y) \\
&= \textstyle\sum_{i=1}^{n} \mathcal{H}(X_i | X_{i-1}, \ldots, X_1) - \sum_{i=1}^{n} \mathcal{H}(X_i | X_{i-1}, \ldots, X_1, Y) \\
&= \textstyle\sum_{i=1}^{n} \mathcal{I}(X_i; Y | X_1, X_2, \ldots, X_{i-1})
\end{aligned}$$

□

Finally we show the chain rule for relative entropy.

**Theorem 2.4.18** (Chain Rule for Relative Entropy)**.** *Suppose two random variables $X$ and $Y$ and two different probability mass functions $p$ and $q$:*

$$D(p(x, y) || q(x, y)) = D(p(x) || q(x)) + D(p(y|x) || q(y|x))$$

*Proof.*

$$\begin{aligned}
D(p(x,y)||q(x,y)) &= \textstyle\sum_x \sum_y p(x,y) \log \frac{p(x,y)}{q(x,y)} \\
&= \textstyle\sum_x \sum_y p(x,y) \log \frac{p(x)p(y|x)}{q(x)q(y|x)} \\
&= \textstyle\sum_x \sum_y p(x,y) \log \frac{p(x)}{q(x)} + \sum_x \sum_y p(x,y) \log \frac{p(y|x)}{q(y|x)} \\
&= D(p(x)||q(x)) + D(p(y|x)||q(y|x))
\end{aligned}$$

□

# Chapter 3

# Literature Review: Motivation of Our Research

This chapter explains the main motivation of our research, namely one of the key weaknesses of the existing CHM's framework—the "double counting" problem, which will be explained in Section 3.1.4 and demonstrated by some examples; then a feasible solution addressing the problem is proposed, that is by adopting an abstract interpretation-based analysis to improve the precision of quantitative information flow analysis.

Quantitative information flow analysis and abstract interpretation, by themselves, are both very broad and active research areas although quantitative information flow analysis is relatively new compared to abstract interpretation. For the sake of completeness and to familiarize the reader with the necessary basis and development of these two areas, brief introductions of them are included before we discuss the weakness of the current quantitative information flow analysis framework and explain how it can be improved using one specific kind of abstract interpretation analysis. However, we do not want to make this chapter self-contained, so for more detailed information about them, readers are advised to refer to wider literature on these two topics. For quantitative information flow analysis, just naming a few, look at the work of Clark, Hunt and Malacaria [11, 13, 12, 16, 14, 15, 37, 44]; while for abstract interpretation, please start with the work of Cousot [18, 19, 22, 20], Jones and Neilson [39].

The rest of this chapter is divided into two main parts: quantitative information flow analysis and abstract interpretation. They are introduced in turn.

## 3.1 Quantitative Information Flow Analysis

The traditional mechanism to protect secure information is access control, however, it is well established that access control *cannot* control the propagation of the secure information once it is released from the source when access is granted in the first place.

Moreover, more and more information security breaches have the form of transferring secrecy to unauthorized parties. A popular example is "spyware": you have received a program from an untrusted source, say *M*. *M* promises to help you to optimize your personal financial investments, which is the private information you have stored in a database on your home computer. The software is free (for a limited time) under the condition that you permit a log-file containing a summary of your usage of the program to be automatically emailed back to the developers of the program (who claim they wish to determine the most commonly used features of their tool). Is such program safe to use? Can you ensure that $M$ is not obtaining your sensitive private financial information by cunningly encoding it in the contents of the innocent-looking log-file [66, 40]? Note that the traditional access control method fails here since the program $M$ has legitimate access to the private database. Actually, these are the situations which require a much stronger policy, namely information flow control [64]. This policy should enable us to track how information flows within the program; access control, while useful, cannot identify these processes and hence isn't suitable for information flow control.

Thus researchers turn to information flow security to reason about how information actually flows within the system.

Quantitative information flow analysis, as its name suggests, not only confirms whether there is any information leakage at all, but also, if there is, it calculates the amount of the information that can be or is leaked.

### 3.1.1 The Origination of Quantitative Information Flow Analysis

Before quantitative information flow analysis, the analyses of information leakage of computer program (system) were all qualitative, which means it simply tells whether there is any information leaked or not. This is, most of the time, not as useful as knowing how much information has been leaked. Meanwhile, in order to enforce the absolute security of computer program (system), non-interference policy was proposed by Goguen and Messeguer in

1982 [30] which states that confidential data may *not* interfere with (affect) any public data. In other words, the public output cannot reflect any change of the confidential data, they are totally independent from each other. This policy is problematic, as it judges far too many programs to be "insecure", for example a password checking program does leak a small amount of information:

```
if H==L then
    access;
else
    deny
fi
```

if the attacker observes deny, they will know his (her) guess of $l$ is wrong. The defect of non-interference policy is elegantly put in [62]:

*In most non-interference models, a single bit of compromised information is flagged as a security violation, even if one bit is all that is lost. To be taken seriously, a non-interference violation should imply a more significant loss. Even … where timing is not available, and a bit per millisecond is not distinguishable from a bit per fortnight … a channel that compromises an unbounded amount of information is substantially different from one that cannot.*

It is this inherent defect of qualitative information flow analysis — overly strict, that leads to the origination of quantitative information flow analysis.

### 3.1.2 The Development of Quantitative Information Flow Analysis

Quantitative information flow analysis can be dated back to Denning's work in the 1970's [24, 25, 27, 28, 26], later other definitions continuously come up [50, 31, 51, 11, 57, 43]. In [24], Denning pioneered the application of information theory to quantify the information leakage of programs, in which she demonstrated the analysis of calculating leakage of a few particular assignments and **if** statements by using Shannon's entropy. However, later her definition of information leakage was demonstrated to be flawed and since then quite a lot of work in the field of quantitative information flow analysis focused on developing the "correct" definition for information leakage calculation, in such a way that either the new definition satisfies the requirement of some particular situation, *e.g.* covert channel [43], **while** loop [44] *etc*; or the new definition is claimed to be capable of sitting in more situations than the existing ones [57]. Next, some typical definitions of quantitative information leakage are introduced. We start from Denning's definition.

#### Denning's Definition

Denning's work [24] defined the information flow as follows: given two program variables $x$, $y$ in a program $P$ and two states $s$, $s'$ in the execution of $P$, there is a flow of information from $x$ at state $s$ to $y$ at state $s'$ if uncertainty about the value of $x$ at $s$ given knowledge of $y$ at $s'$ is less than uncertainty about the value of $x$ at $s$ given knowledge of $y$ at $s$, mathematically $\mathcal{H}(x_s|y_s) - \mathcal{H}(x_s|y_{s'}) > 0$. However, it is established that her measure, although seeming quite natural, is arguably flawed, because the observation of $y$ at $s'$ does not necessarily contain more information of $x$ than that at $s$ as the second term $\mathcal{H}(x_s|y'_s)$ accounts for the final observation of $x$ but effectively assumes that the initial observation of $y$ has been forgotten. This is a safe assumption only if we know that the observer has no memory. However, we must assume that the observer knows both the initial and final values of $y$ in general [12].

**McLean's Definition**

In [50], McLean introduced *time* into the analysis of the notion of secure information flow. His model is highly abstract, and it states that a system is secure if $p(L_t|(H_s, L_s)) = p(L_t|L_s)$, where $L_t$ describes the value taken by the low system objects at time $t$ while $L_s$ and $H_s$ are the sequences of values taken by the low and high objects, respectively, at times preceding $t$. McLean's flow model provides the right security model for a system with memory. However, his work is actually qualitative and there is not enough machinery to implement an analysis based on it [12] since his work is intended to provide a way for evaluating security models rather than a means of evaluating real systems.

**Gray's Definition**

The use of conditional mutual information in the context of information leakage was proposed by Gray [31], however, his work is not aimed at measuring information leakage but to define it for a flow model. Such work also include Volpano and Smith's [70] which showed that well-typed programs cannot leak confidential data in polynomial time.

### Clark, Hunt and Malacaria's (CHM's) Definition

Clark, Hunt and Malacaria have done a lot of work quantifying the actual information leakage [15, 14, 16, 12, 11] using information theory explicitly. Their system model is essentially McLean's [50], and the quantity they estimate was first defined by Gray [31] in the context of a channel capacity theorem. For example, their analysis will give the information leakage of the following programs as the maximum $k$ bits and 1 bit respectively (suppose $H$ is a $k$ ($k$ is a positive integer) bit variable):

- $L = H$
- **if** $H == 0$ **then** $L = 1$; **else** $L = -1$; **fi**

More details of their definition and analysis will be discussed in the next section.

Later Malacaria [44] gave a more precise quantitative analysis of loop constructs. He defined an information theoretical formula for leakage of the command **while** $e$ $M$ in which both the amount and the rate of leakage are calculated. Following this work, Chen and Malacaria [7] recently presented a mechanism for quantitative leakage analysis for multi-threaded programs. The basic idea was based on program transformation: the multi-threaded program with a probabilistic scheduler is transformed to a single-threaded program with probabilistic operators, and then the leakage of the transformed program can be computed using the formula of [44].

### Other Definitions

Recently, some people proposed other definition of information leakage. Different from all the definitions above, Clarkson, Myers and Schneider have proposed to quantify the information flow using beliefs [57]. Their idea is to monitor how the attacker's belief changes upon their observations of outputs of a (probabilistic) program. If before executing the program, the attacker believes the secret is overwhelming to be something, and when the program is terminated and the observer finds out that the secret is something else, then it is claimed that there is a large amount of information leaked to the attacker. CMS' definition of information as belief is objective which depends on the initial guess of the attacker, so it seems that the more wrong the attacker's pre-belief is, the more information they would get from the post-belief.

### Other Related Work

Di Pierro, Hankin, and Wiklicky [61] used the notion of *approximate noninterference* to conceptualize an "up to $\epsilon$" noninterference. They introduced a quantified measure of the similarity between two processes in a process algebra. It is measured using the supremum norm of the difference matrix of probabilistic state transition matrices the processes create. This quantity, as they demonstrated, is linked with an attacker's ability to distinguish two processes. Finally, the paper showed an abstract interpretation that allows approximation of the confinement of a process. Their more recent work [60] generalized this to measure the confinement in probabilistic transition systems and gave well-understood examples.

McCamant and Ernst [47] developed an approach to track the use of data through arbitrary calculation in programs to determine how much information about secret inputs was revealed by the public outputs. It was shown that an implementation of such techniques based on dynamic binary translation on C, C++ and object-oriented programs can provide meaningful security checks.

In another recent paper [69], Smith reviewed the existing definitions of quantitative information flow, and refined the security bound given by Fano's inequality. He proposed a new conceptual foundation of "vulnerability" and proposed to use the min-entropy in Renyi entropy instead of Shannon's entropy to measure the information leakage.

### Discussion

While there is not an universally accepted definition of information leakage, the application of entropy from information theory to quantify information flow seems common. And there are still researchers investigate the advantage and disadvantage of each definition mentioned above and propose new definitions, however, in our research we will adopt the definition of CHM's because it is quite natural and objective when it comes to the real world problem of information flow.

### 3.1.3 The Framework of Quantitative Information Flow Analysis

Most of the work in the field of quantitative information flow analysis mentioned above only deals with the definition of information leakage which does not provide a systematic, formal framework to analyze quantitative information flow; however, Clark, Hunt and Malacaria developed a whole analysis framework for quantifying information flow. Their analysis is presented as a set of syntax-directed inference rules, and there is some attempt to automate it [56]. In the beginning, their framework worked for a simple language without loops and there are also some constraints on the equality test ( *e.g.* it can only test against constants) [11]; later, they improved their analysis framework by including while loops and made use of some mathematical properties [12] to make the analysis more precise. This section introduces the main ideas of their analysis framework.

The rules they present are intended to derive bounds on the leakage of a variable at a program point, given only assumptions on the entropy of the confidential variable at the entry point. Such assumptions actually give very limited knowledge of the distribution of input values and this means that a direct calculation of the leakage at a program point is usually impossible.

#### Main Idea

The ultimate aim of quantitative information analysis is to place an upper bound on the amount of information that can be leaked through the program. The information leakage they are concerned with is the information that may be leaked from private variables to public ones. In other words, they are modeling a situation in which the environment delivering high inputs to the program is trusted, even though the program itself is not. This is appropriate for example in the analysis of untrusted code which is to be downloaded and run on a user's computer, where the user is the owner of the confidential data.

Their programs are written in **while** language, which contains just the following control constructs: assignment, **while** statements, **if** statements and sequential composition; the left hand side of assignment are variable identifiers while the right hand side are integer or boolean expressions; **while** loops and **if** statements involve boolean expressions in the standard way; expressions in their language define total functions on stores. The syntax of the while language is standard and hence omitted here.

The program they consider is deterministic, so for any given program $P$, the semantics induces a partial function $[\![P]\!] : \Sigma \rightarrow \Sigma$, where $\Sigma$ is the domain of stores, which is a finite map from variable names to $k$-bit integers (in the range $-2^{k-1} \leq n \leq 2^{k-1}$) and booleans.

The variables of a program $P$ are partitioned into two sets, $H$ (*high*) and $L$ (*low*). High variables may contain confidential information when the program is run, but these variables cannot be examined by an attacker at any point before, during or after the program's execution. Low variables do not contain confidential information before the program is run and can be freely examined by an attacker before and after (but not during) the program's execution.

The main idea of their analysis is: as they are interested in how much of the information carried by the high inputs to a program can be learnt by observation of the low outputs, assuming that the low inputs are known. Since the language is deterministic, any variation in the outputs is a result of variation in the inputs. Once the knowledge of the program's low inputs is accounted for, the only possible source of surprise in an output is interference with high inputs.

Given a program variable (or set of program variables) $X$, let $X^l$ and $X^w$ be, respectively, the corresponding random variables on entry to and exit from the program (termination is assumed), the measure of the amount of leakage into $X$ due to the program is:

$$\mathcal{L}(X) =^{\text{def}} \mathcal{H}(X^w|L^l)$$

In order to systematically analyze the information leakage, at each program point a random variable corresponding to observations of the low variable at that point is defined. When it comes to **while** loop, the *use-definition graph* (UDG), a directed graph whose nodes are program points, is used to identify the source nodes of all possible information which flows into some certain program point of the while loop, the details of how the identification works is omitted here as it is not the scope of this thesis, however, please refer to [12] for detailed explanation.

They also noticed that low program variables are hard to be counted for, as either there will be little or no knowledge available for the low variables or the distribution for low inputs may actually be in the control of attackers. To cope with this problem, they work with an interval which is the bound of the entropy representing the range of information that can be leaked to the observable program variable at each program point, and the final information leakage from the high inputs to the low outputs can be obtained automatically using their inference rules which are introduced in the next sub-section.

### Inference Rules

Here we gave a basic idea as how exactly the analysis framework of CHM's works. The following are some of their inference rules as shown in Table 3.1. Where $\mathcal{B}(q) = -q \log q - (1-q)\log(1-q)$ and $\mathcal{U}_k(q) = -q \log q - (1-q) \log \frac{1-q}{2^k-1}$. The expression analysis rules have the form $\Gamma \vdash E : [a, b]$ where $\Gamma$ is a partial function from Var to real-valued pairs (representing intervals) of the form $[a, b]$ with $a \leq b$. The meaning of a rule $\Gamma \vdash E : [a, b]$ is that the expression $E$ has information leakage in the interval $[a, b]$ assuming that the leakage of each variable $x$ in $E$ lies in the interval $\Gamma(x)$. If $[\_, b]$ is used, it means the upper bound of information leakage of $E$ is $b$ (the lower bound doesn't matter), in other words, it means $E$ leaks at most $b$ bit(s) information and similarly for $[a, \_]$ which means the lower bound of information leakage of $E$ is $a$, namely at least $a$ bit bit(s) information is leaked by $E$.

From the table we can see that no matter what the exact inference rule it is, without any exception, it embodies the concept that the total amount of information that can be leaked cannot exceed the information that has flowed into the system.

We summarize how their analysis works for a whole program: their analysis calculates bounds on the best and worst cases for leakage over the com-

$$\text{DP}\frac{n_1 \vdash [E(n_1)] \leq b_1, \ldots, \quad n_k \vdash [E(n_k)] \leq b_k}{n \vdash [E] \leq \sum_{i=1}^{k} b_k}$$

$$\text{EConj}\frac{\Gamma \vdash E : [a_1, b_1] \quad \Gamma \vdash E : [a_2, b_2]}{\Gamma \vdash E : [\max(a_1, a_2), \min(b_1, b_2)]}$$

$$\text{BConj}\frac{\Gamma \vdash B : [a_1, b_1] \quad \Gamma \vdash B : [a_2, b_2]}{\Gamma \vdash B : [\max(a_1, a_2), \min(b_1, b_2)]}$$

$$k\text{-Bits}\frac{}{\Gamma \vdash E : [0, k]} \qquad \text{1-Bit}\frac{}{\Gamma \vdash B : [0, 1]}$$

$$\text{Const}\frac{}{\Gamma \vdash n : [0, 0]} \qquad \text{Var}\frac{}{\Gamma, x : [a, b] \vdash x : [a, b]}$$

$$\text{And}\frac{\Gamma \vdash B_i : [\_, b_i] \;\; i = 1, 2}{\Gamma \vdash (B_1 \wedge B_2) : [0, b_1 + b_2]} \qquad \text{Neg}\frac{\Gamma \vdash B : [a, b]}{\Gamma \vdash \neg B : [a, b]}$$

$$\text{Plus}\frac{\Gamma \vdash E_i : [\_, b_i]}{\Gamma \vdash (E_1 + E_2) : [0, b_1 + b_2]} \qquad \text{Eq(1)}\frac{\Gamma \vdash E_1 : [\_, b_1] \quad \Gamma \vdash E_2 : [\_, b_2]}{\Gamma \vdash (E_1 == E_2) : [0, b_1 + b_2]}$$

$$\text{Eq(2)}\frac{\Gamma \vdash E_1 : [a, \_] \quad \Gamma \vdash E_2 : [\_, b]}{\Gamma \vdash (E_1 == E_2) : [0, \mathcal{B}(q)]} \quad \frac{1}{2^k} \leq q \leq \frac{1}{2},\ \mathcal{U}_k(q) \leq (a - b)$$

$$\text{If}\frac{\Gamma \vdash e : [\_, b] \quad \Gamma \vdash c_i \downarrow x : [\_, b_i]}{\Gamma \vdash \textbf{if}\ e\ \textbf{then}\ c_1\ \textbf{else}\ c_2\ \downarrow x : [0, b + b_1 + b_2]}$$

Table 3.1: Leakage inference: Expressions

plete set of possible input values for the low variables, thus the leakage of the program is within the range $\mathcal{H}^-(X^w|L^l) \leq \mathcal{H}(X^w|L^l) \leq \mathcal{H}^+(X^w|L^l)$ where $\mathcal{H}^-(X^w|L^l)$ is $\min\mathcal{H}(X_{L=l})$ and $\mathcal{H}^+(X^w|L^l)$ is $\max\mathcal{H}(X_{L=l})$ and then the final information leakage by the program is obtained by applying the corresponding inference rule at each program point.

Let's take a look at an example:

**Example 3.1.1.**

```
if $Y = 0$ then
    $X := 0;$
else
    $X := 1;$
fi
```

with $Y$ high-security variable and $X$ low-security variable to see how the inference rules work. Suppose $Y$ is a 32-bit variable and the input distribution makes $Y$ uniform over its $2^{32}$ possible values and independent of $X$. The analysis can start with $\Gamma_0 = X : [0,0],\ Y : [32,32]$. The rules presented above are easily seen to derive: $\Gamma_0 \vdash Y = 0 : [\epsilon,\ \epsilon]$ (where $\epsilon = \mathcal{B}(1\backslash 2^{32}) \approx 7.8 \times 10^{-7}$). Thus , using **if**, we can derive: $\Gamma_0 \vdash c \downarrow X : [0, \epsilon]$.

In later work [12, 15], more mathematical properties have been explored, and hence more precise inference rules such as [ZeroMult] and [OddMult]:

$$\text{ZeroMult}\frac{}{n \vdash [E_1 * E_2] = 0} E_2 = 0$$

$$\text{OddMult}\frac{n \vdash [E_1] \sim a \; n \vdash [E_2] \sim a}{n \vdash [E_1 * E_2] \sim a}$$

where $\sim$ can be any of $\leq$, $=$, $\geq$. Please note that these rules together with the inference rules above only work outside the while loops. They use UDG to identify all the possible sources of information that flows into the while loop with which their analysis is very conservative, which was later improved further by Malacaria's work [44].

### 3.1.4 The Weakness of Quantitative Information Flow Analysis—"Double Counting"

Although Clark, Hunt and Malacaria's framework is systematic, their analysis is too conservative in some sense. Let's take a contradicting example to demonstrate the problem:

$$X := Y;$$
$$Z := X + Y$$

Suppose that both $X$ and $Y$ are high variables and $Z$ is the low observable variable. Suppose both $\mathcal{H}(X) = a$ and $\mathcal{H}(Y) = b$ hold before entering the program, then after the execution of $X := Y$, using the [DP] inference rule above, we can deduce that $\mathcal{H}(X) = \mathcal{H}(Y) = b$. Then by applying the [Plus] inference rule, we can finally have $\mathcal{H}(Z) = \mathcal{H}(X) + \mathcal{H}(Y) = 2b$.

Although this example is very straightforward and yet a little bit extreme, it demonstrates the problem of "double counting" very clearly which exists in the CHM's analysis framework. Without any way of incorporating the relationship between program variables together, the best that can be said about the total amount of information that can be leaked by a program is the sum of the information of all the possible sources (except in some very special cases *e.g.* [ZeroMult]), even though some parts of this information are exactly the same as each other, hence the name of "double counting".

If we look at the above simple program again, it is not hard to notice that actually $X$ and $Y$ become equivalent to each other, hence their information is entirely shared. Adding them together doesn't double the amount of information as they contain exactly the same information as each other. Therefore, $Z$ does not contain twice the information as either $X$ or $Y$, actually it contains exactly the same information as either of them.

A more general example would be:

$$X_1 := X_2 := \ldots = X_n;$$
$$X := \sum_i X_i$$

If we apply CHM's analysis framework without taking into account the fact that $X_i$s are all equivalent, it is ended up with $X$ having $n$ times the information as that of $X_i$, when $n$ is sufficiently large, it would deduce that $X$ has a very large amount of information (suppose the memory is sufficiently large and $X$ can contain infinite bits) which is clearly much too imprecise.

Thus, this simple example gives us the hint that relationships between program variables can be used to deduce more precise quantitative information leakage analysis. For example in the above program, if the relationship of $X_1 == X_2 == \ldots == X_n$ can be successfully incorporated into the analysis, then no matter how large $n$ is, the information content of $X$ should be determined to be equal to that of any single $X_i$.

Luckily there is a well developed framework to obtain this kind of relationship (constraints) between program variables, namely abstract interpretation, which will be introduced in the next section.

## 3.2 Abstract Interpretation

There are some specific techniques in the area of abstract interpretation which can nicely approximate the linear relationship between program variables [19, 6, 68]. Before we go into details about it, a simple introduction of the basics of abstract interpretation is presented.

### 3.2.1 Intuition

Generally speaking, abstract interpretation refers to the name that applies to quite a few techniques for reasoning about programs by evaluating them over non-standard domains whose elements denote properties of the standard domain. The intuition behind abstract interpretation is: when it comes to program analysis, it is time-consuming and often impossible trying to analyze programs running on all possible inputs of all possible paths; and also we are only interested in some certain properties of the program which can be abstracted, or these properties already can tell us everything we need to know about the program. For example, instead of handling sets of integers, one might want to over-approximate them using an interval. If all computations are done monotonically, the result interval is necessarily a superset of the exact set of possible values at the end of execution.

There are so many applications of abstract interpretation in real life, for example the most common one of "the rule of sign" in which all the positive numbers are denoted by "+", all the negative as "-" and "?" as unknown meaning it can be either positive or negative. And the abstract semantics defines $(+) +^\sharp (+) = +$, $(-) +^\sharp (-) = -$, $(+) +^\sharp (-) =?$, $(+) -^\sharp (-) = +$, $(-) -^\sharp (+) = -$ and *etc* where $+^\sharp$ means the abstract addition while $-^\sharp$ is the abstract subtraction. Next we use a little bit complicated example than this to illustrate the usefulness of abstract interpretation in practice.

**Example of Abstract Interpretation—the "Casting Out Nines"**

We adopt the classical example of "casting out nines" (mod 9) from [1] to illustrate how abstract interpretation can be used to check whether a complicated calculation is correct or not within seconds of time. Consider the following calculation:

$$123 * 457 + 76543 =? = 132654$$

The abstract interpretation technique does the check in this way: the above is checked by reducing 123 to 6, 457 to 7, and 76543 to 7, and then reducing 6*7 to 42 and so further to 6, and finally 6+7 is reduced to 4. This differs from 3, the sum modulo 9 of the digits of 132654 so the calculation was incorrect. The method abstracts the actual computation by only recording values modulo 9. Even though much information is lost, useful results are still obtained since this implication holds: if the alleged answer modulo 9 differs from the answer got by casting out nines, there is definitely an error. The mathematical proof of this method is not hard, and we omit it for the sake of space here.

More formally, here the abstraction is $\alpha(x) = x \bmod 9$ and the corresponding concretization is $\gamma(a) = \{x | x \bmod 9 = a\}$. The abstract operators are $x +^\sharp y = (x + y) \bmod 9$ and $x *^\sharp y = (x * y) \bmod 9$.

Another thing we can see from the example is that abstract interpretation simulates many computations at once. As an example, the abstract value 6 actually represents a set of integer numbers with the same property that their module 9 is 6.

### 3.2.2 Formal Definition

Formally, abstract interpretation is a theory of sound approximation of the semantics of computer programs, based on monotonic functions over ordered sets. It has been formalized by Cousot and Cousot [17, 18] for flow-chart language. Moreover their work has had a considerable impact on later work in various areas, of which related to information flow analysis includes David Monniaux's work on abstract interpretation of probabilistic programs [54, 55, 53], Hunt and Mastroeni's work on abstract non-interference [38, 46] and etc.

First let's introduce partially ordered set (poset) and monotone function.

#### Poset

A poset is a pair $(P, \leq)$ where $P$ is a set, and $\leq$ is a partial order (a reflexive, transitive and anti-symmetric relation) on P. That is $\forall x,\ y,\ z \in P$:

- $x \leq x$ (reflexive)
- $x \leq y \ \wedge \ y \leq x \ \Rightarrow \ x = y$ (anti-symmetric)

- $x \leq y \land y \leq z \Rightarrow x \leq z$ (transitivity)

**Monotone Function**

Let $A$ and $B$ be posets. A map $f : A \rightarrow B$ is *monotone* if $a \leq a' \Rightarrow f(a) \leq f(a')$.

**Formal Definition of Abstract Interpretation**

Let $A$ and $C$ be two posets, namely abstract domain and concrete domain respectively.

A function $\alpha$ is called an abstract function if it maps an element $x$ in the concrete domain $C$ to an element $\alpha(x)$ in the abstract domain $A$.

A function $\gamma$ is called an concretization function if it maps an element $x'$ in the abstract domain $C$ to an element $\gamma(x')$ in the concrete domain $A$.

**Correctness of Abstract Interpretation**

No matter how abstract the abstract domain is, correctness (soundness) of the analysis is central. In other words we must be able to prove that the properties resulting from the abstract analysis are satisfied by the values that the standard semantics operates on.

For example, if we have an abstraction for the simple arithmetic operation $X := X * 2$ given that $X$ is an integer, and each variable $X$ is abstracted as its parity (abstract domain of (even , odd or not known), then the correctness of the abstract analysis will ensure a final abstract value of $X$ even or not known (either even or odd), but if the analysis give us $X$ odd as the final value, then it is wrong.

More precisely, the values obtained from the standard semantics should be in the set of concretization of the corresponding abstract property. As soundness is not the main topic of this thesis, it is only briefly mentioned here although it is vital.

Let $C_1, C_2, A'_1$ and $A'_2$ be ordered sets. The concrete semantics $f$ is a monotonic function from $C_1$ to $C_2$. A function $f'$ from $A'_1$ to $A'_2$ is said to be a sound abstraction of $f$ if for all $x' \in A'_1, (f \cdot \gamma)(x') \leq (\gamma \cdot f')(x')$.

### 3.2.3 Application of Abstract Interpretation

To make abstract interpretation of any use, first of all an abstract domain has to be decided on which nicely identifies the property of interest. The most common are numerical abstract domains, such as the simplest interval [17], more complex convex polyhedron (linear (in)equality) [19, 35, 36, 32, 33] and other domains [52].

The application of abstract interpretation usually consists of computing a fixed point for the program on the abstract domain. This usually involves an iterative process, generating a sequence of approximations to the post fixed point of the program and checking for convergence on each iteration. Hence, consideration must be given to the operations required to compute the fixed point, or at least some useful approximation of the fixed point. Monotonicity is a requirement of fixed point equations, therefore the structure of abstract domains ranges from posets to complete lattices. When analyses are defined over abstract domains with infinite or very long chains, they may require a technique that either forces or accelerates convergence. The notion of widening and narrowing were introduced to manage such situations and are reviewed in [21].

## Widening and Narrowing

A widening is an operator on two successive iterations in a fixed point calculation over increasing chains that approximate a post fixed point. The loss in precision is compensated for by operational tractability, as in practice, post fixed point can be easier to compute than fixed points. The following definitions are due to [21].

**Definition 3.2.1.** *A widening denoted $\bigtriangledown$, on the posets $\langle P, \leq \rangle$ is defined by $\bigtriangledown : P \times P \rightarrow P$ such that $\forall x, y \in P.\ x \leq x \bigtriangledown y$ and $\forall x, y \in P.\ y \leq x \bigtriangledown y$ and for all increasing chains $x_0 \leq x_1 \leq \ldots$, the increasing chain defined by $y_0 = x_0, \ \ldots, y_{i+1} = y_i \bigtriangledown x_{i+1}$ is not strictly increasing, that is $y_{l+1} \leq y_l$ for some $l$.*

For example, the widening operates on the abstract domain of integer intervals works as follows:

$$[a_1, b_1] \bigtriangledown [a_2, b_2] =$$

$$[\textbf{if } a_2 < a_1 \textbf{ then } -\infty \textbf{ else } a_1,$$

$$\textbf{if } b_2 > b_1 \textbf{ then } +\infty \textbf{ else } b_1]$$

**Example 3.2.2.**

$$[1, 10] \bigtriangledown [1, 11] = [1, +\infty]$$

$$[1, +\infty] \bigtriangledown [0, 12] = [-\infty, +\infty]$$

Narrowing, denoted $\bigtriangleup$, is a similar technique that is applied to decreasing chain.

**Definition 3.2.3.** *A narrowing denoted $\bigtriangleup$, on the posets $\langle P, \leq \rangle$ is defined by $\bigtriangleup : P \times P \rightarrow P$ such that $\forall x, y \in P$. $(y \leq x) \Rightarrow y \leq (x \bigtriangleup y) \leq x$ and for all decreasing chains $x_0 \geq x_1 \geq \ldots$, the decreasing chain defined by $y_0 = x_0, \ldots, y_{i+1} = y_i \bigtriangleup x_{i+1}$ is not strictly decreasing, that is $y_{l+1} \geq y_l$ for some $l$.*

Similarly, narrowing operates on abstract domain of intervals:

$$[a_1, b_1] \bigtriangleup [a_2, b_2] =$$

$$[\textbf{if } a_1 == -\infty \textbf{ then } a_2 \textbf{ else } \text{MIN}(a_1, a_2),$$

$$\textbf{if } b_1 == +\infty \textbf{ then } b_2 \textbf{ else } \text{MIN}(b_1, b_2)]$$

**Example 3.2.4.**

$$[-\infty, +\infty] \bigtriangleup [-\infty, 101] = [-\infty, 101]$$

$$[-\infty, 101] \bigtriangleup [0, 100] = [0, 100]$$

$$[0, 100] \bigtriangleup [0, 99] = [0, 99]$$

In the next section, how abstract interpretation can be used to easily derive linear constraints between program variables is introduced. The correctness proof is omitted for the sake of space, however, it is guaranteed by the soundness of general abstract interpretation.

### 3.2.4 Abstract Interpretation for Linear Constraints Detection

The abstract domain of our analysis is integer polyhedra, the advantage of this domain is that it contains the relationship between variables.

Now we introduce how abstract interpretation can be used to automatically deduce linear constraints (convex polyhedron) between program variables. The work was first proposed by Cousot and Halbwachs [22]. Later, a refined widening for polyhedra proposed in Cousot and Halbwachs' paper is reported in [32, 33]. They demonstrated that it is possible to lose less information by reformulating the first polyhedron so that the number of common constraints is maximized. More recent work on polyhedrons is that of Howe and King [35, 36]. They proposed an abstract interpretation on finite domain of constraint logic programs. By allowing the propagation of linear constraints at compile time as a program specialization that preserves the semantics of the original program, the search space is reduced and problem solving is expedited. Some of the analyzed programs exhibited a significant execution time improvement. Other works directly on integer polyhedra are [68, 6] of which the latest is [6].

In [68], set of finite sets of linear inequalities as abstract domain is proposed where each inequality contains at most two variables with unrestricted coefficients (*e.g.* coefficients $\in \mathcal{R}$). This is the domain which is richer than intervals as it is relational. They exploit a way to decompose the original set into a series of projections, one for each two dimensional plane, by doing so, all the operations can be expressed in terms of two dimensional case which proves to be efficient. The set of linear inequality constraints our algorithm can deal with don't necessarily have to have this neat form, namely it can contain more than two variables.

In [6], how to derive the most precise integer polyhedra of a system of inequalities is discussed. The integer polyhedron is grown step by step, first an integer solution satisfying the set of linear constraints are calculated, then a distinct solution which has the maximal distance from the previous solution is calculated. Then a convex hull of this point and the previous space is taken and the process is iterated until it reaches the dimension of the final solution. To avoid the problem of ending up with an unmanageably large number of inequalities, a Monte Carlo approximation of the number of integer points that a constraint bars from a polyhedron is calculated. The least contributing constraints are relaxed.

In our analysis, we need to derive a new set of linear constraints from the set of linear constraints before the assignment. [68, 6] don't deal with the transformation of the linear constraints directly. However, we can directly apply [22] to get the linear constraints. [6] can be used on the linear constraint set after the assignment to get the most precise integer polyhedron. In the next subsection, how it works is detailed.

### 3.2.5 Algorithm for deriving linear constraints of assignment

In this section we specify how to derive linear constraints for assignments based on the type of the assignment.

Basically, there are two types of assignments: invertible assignment and non-invertible assignment. For example $X_2 := X_1 + X_2/2 + 1$ $(X_1,\ X_2 \in \mathcal{N})$ is invertible as $X_2$ can be recovered by subtracting $X_1 + 1$ and then times the result by 2 while $X_2 := X_1 + 1$ is non-invertible as it can not be recovered in the same way.

Suppose we have a system of linear constraints before the assignment:

$$\begin{cases} c_{11}x_1 + \ldots + c_{1n}x_n \geq b_1 \\ \ldots \\ c_{m1}x_1 + \ldots + c_{mn}x_n \geq b_m \end{cases}$$

Where $b_i,\ c_{i,j} \in \mathcal{R}$.

If the assignment is non-invertible, $x_i = c'_1x_1 + c'_2x_2 + \ldots + c'_{i-1}x_{i-1} + c'_{i+1}x_{i+1} + \ldots + c'_nx_n$ $(c'_i \in \mathcal{R}\ )$.

The derivation of the linear constraints after this assignment would be: to substitute each $x_i$ in the original constraints with $c'_1x_1 + \ldots + c'_{i-1}x_{i-1} + c'_{i+1}x_{i+1} + \ldots + c'_nx_n$, then put the assignment itself as one of the constraint as well:

$$\begin{cases} c_{11}x_1 + \ldots + c_{1i}(c'_1x_1 + \ldots + c'_{i-1}x_{i-1} + c'_{i+1}x_{i+1} + \ldots + c'_nx_n) + c_{1(i+1)}x_{i+1} + \ldots + c_{1n}x_n \geq b_1 \\ \ldots \\ c_{m1}x_1 + \ldots + c'_{mi}(c'_1x_1 + \ldots + c'_{i-1}x_{i-1} + c'_{i+1}x_{i+1} + \ldots + c'_nx_n) + c_{1(i+1)}x_{i+1} + \ldots + c_{mn}x_n \geq b \\ x_i = c'_1x_1 + c'_2x_2 + \ldots + c'_{i-1}x_{i-1} + c'_{i+1}x_{i+1} + \ldots + c'_nx_n \end{cases}$$

And then simplify the above system to give the final result.

If the assignment is invertible, $x_i = c'_1x_1 + c'_2x_2 + \ldots + c'_ix_i + \ldots + c'_nx_n$ The derivation of the linear constraints after this assignment would be: first, rename each of the variable at the right hand side of the assignment, *e.g.* $x_1 = x'_1,\ x_n = x'_n$, then replace each variable in the original constraints with its renamed one, thus the constraints after the assignment are:

$$\begin{cases} c_{11}x'_1 + \ldots + c_{1i}(c'_1x'_1 + \ldots + c'_ix'_i + \ldots + c'_nx_n) + c_{1(i+1)}x'_{i+1} + \ldots + c_{1n}x'_n \geq b_1 \\ \ldots \\ c_{m1}x'_1 + \ldots + c_{mi}(c'_1x'_1 + \ldots + c'_ix'_i + \ldots + c'_nx'_n) + c_{1(i+1)}x'_{i+1} + \ldots + c_{mn}x'_n \geq b_m \end{cases}$$

Simplification may be needed. Please note that the work of Chapter 5 and 6 are all based on this.

Here we adopt the example from [22] to demonstrate how to derive linear constraints for assignment.

**Example 3.2.5.** *suppose the input assertion to an assignment is defined by*

$$\begin{cases} X_2 \geq 1, \\ X_1 + X_2 \geq 5, \\ X_1 - X_2 \geq -1 \end{cases}$$

*if the assignment is non-invertible such as $X_2 := X_1 + 1$, the output assertion would be just getting rid of $X_2$ from the original set of inequalities, add the assignment itself to the new set as well:*

$$\begin{cases} X_1 \geq 2, \\ X_2 - X_1 = 1 \end{cases}$$

*if the assignment is invertible such as $R_2 := R_1 + R_2/2 + 1$, then the output assertion is obtained by reformulating $R_1$ as $R_1'$ and $R_2$ as $2R_2' - 2R_1' - 2$ then substitute back to the original input assertion:*

$$\begin{cases} 2X_2' - 2X_1' \geq 3, \\ 2X_2' - X_1' \geq 7, \\ -2X_2' + 3X_1' \geq -3 \end{cases}$$

Thus, for our simple example of:

$$X := Y;$$
$$Z := X + Y;$$

either representation will easily give the result of:

- Before entering the program, no information is obtainable about the linear constraint of either $X$ or $Y$;
- After the first assignment, it is $\{X - Y = 0\}$;
- After the second assignment, it is $\{Z - X - Y = 0,\ X - Y = 0\}$.

This example is too simple to actually need to use abstract interpretation to work out the linear constraints between program variables. However, abstract interpretation is very powerful, as it can identify those linear constraints among the variables that never appear explicitly in the program context, and often escape the notice of anyone who is studying it. The following is such an example [22]:

```
{$P_0$}          $I := 2; J := 0;$
{$P_1$}     $L$ :
{$P_2$}          **if** ... **then**
{$P_3$}          $I := I + 4$
{$P_4$}          **else**
{$P_5$}          $J := J + 1;\ I := I + 2;$
{$P_6$}          **fi**
{$P_7$}          **go to L**;
```

The inequality representation will give the approximation of linear restraints as:

- (0): no information

- (1): I=2, J=0
- (2), (3), (5): 2J+2≤I, J≥0
- (4): 2J+6≤I, J≥0
- (6): 2J+2≤I, J≥1
- (7): 2J+2≤I, 6≤I+2J, J≥0

## 3.3 Case study-Comparison With CHM's Analysis

In this section we use some examples to show the usefulness of abstract interpretation where CHM's analysis falls short.

### 3.3.1 Program with Single Linear Constraint

**Example 3.3.1.**

$$X := Y;$$
$$Z := X + Y;$$

already demonstrates the point that abstract interpretation can provide useful information which CHM's analysis is not capable of. Basically the CHM's analysis can claim that the information leaked into $Z$ is the sum of that of $X$ and $Y$, that is $\mathcal{H}(Z) = \mathcal{H}(X) + \mathcal{H}(Y)$ if both $X$ and $Y$ are high program variables; while the abstract interpretation can infer the constraint $X = Y$ which can then be used to establish $\mathcal{H}(X) = \mathcal{H}(Y)$, and the maximum joint entropy of $X$ and $Y$ is $\mathcal{H}_{\max}(X, Y) = \mathcal{H}(X) = \mathcal{H}(Y)$, finally the maximum amount of information that can be leaked into $Z$ is $\mathcal{H}(Z) = \mathcal{H}(X) = \mathcal{H}(Y)$.

### 3.3.2 Program with Multiple Linear Constraints

The following program (Example 3.3.2) has more than one linear constraint between high program variables $X$ and $Y$ (as shown in Figure 3.1). The abstract interpretation is expected to derive the following linear constraints: suppose there isn't any other constraint of $X$ and $Y$, then after the **if** statement, abstract interpretation is expected to derive $X + Y \geq 5, X - Y \geq -1$; after the assignment of $Y := X + \frac{Y}{2} + 1$, it is $2Y - 2X \geq 3,\ 2Y - X \geq 7,\ -2Y + 3X \geq -3$. Suppose $\mathcal{H}(X) = a$, we would expect the maximum joint entropy of $X$ and $Y$ to be between $a$ and the result of CHM's simply because neither are $X$ and $Y$ independent from each other (as treated in CHM's framework) nor they have such a close relationship that they are equivalent to each other as in Example 3.3.1; our analysis demonstrated in details in Chapter 5 Section 5.7 confirms this while the best of CHM's analysis can do is still the same as claiming that the sum of information of $X$ and $Y$ will be leaked into $Z$ which is $a + \log n$ (suppose $Y$ can have $n$ values) .

**Example 3.3.2.**

```
if $X + Y \geq 5$ && $X - Y \geq -1$ then
    $Z := 0;$
else
        $X = 3;$
        $Y = 4;$
fi
$Y := X + \frac{Y}{2} + 1;$
$Z := f(X, Y);$
```

Where $f(X, Y)$ is some linear function of $X$ and $Y$. This program is discussed in detail in Chapter 5 Section 5.7.

### 3.3.3 A While Loop

Now let's use a simple While loop program to show the point again, and the linear constraint between $X$ and $Y$ that can be obtained by abstract interpretation is demonstrated in Figure 3.2.

**Example 3.3.3.**

```
while $X \leq Y$
    $Z := 0;$
    $Z := X + Y;$
    $X := X + 1;$
end while
```

with $X$ and $Y$, each can take integer values in the range of $[1, \ldots, n]$, and

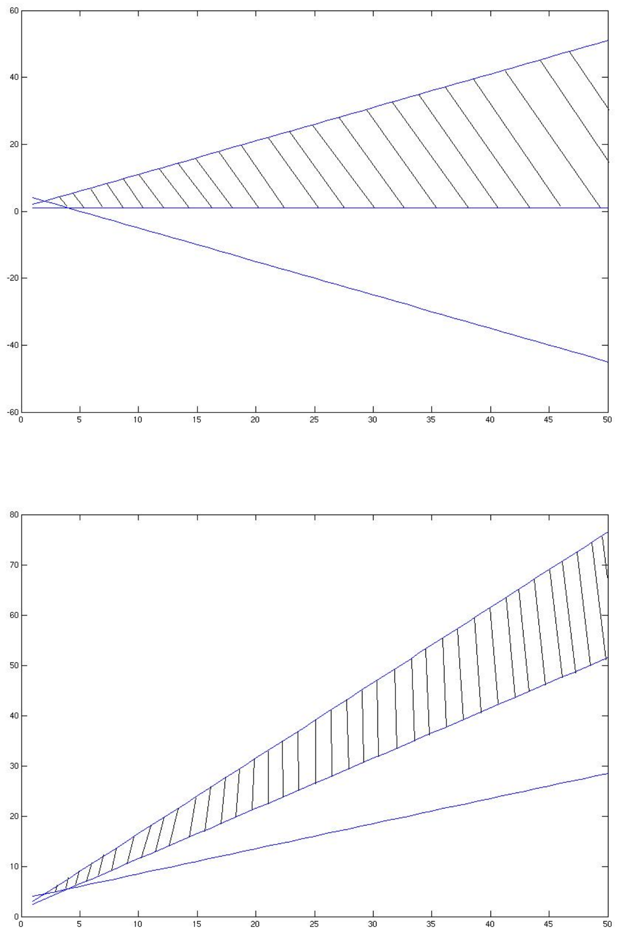

Figure 3.1: Polyhedron for Example 3.3.2 with horizontal axis being $X$ and vertical axis being $Y$

the marginal constraint $\mathcal{H}(Y) = a$.

CHM's analysis will first conduct a dependence analysis which is very conservative, and just tells that $Z$ depends on $X$ and $Y$, hence the maximum information that can be leaked into $Z$ is the sum of the largest amount of information of $X$ and $Y$; our analysis, on the other hand, can use the result of abstract interpretation of $X \leq Y$, although it is not precise, to derive a better upper bound of information leakage which is smaller than CHM's. Our analysis can be improved if the abstract interpretation is able to derive tighter linear constraints. Detailed discussion is also in Chapter 5 Section 5.7.

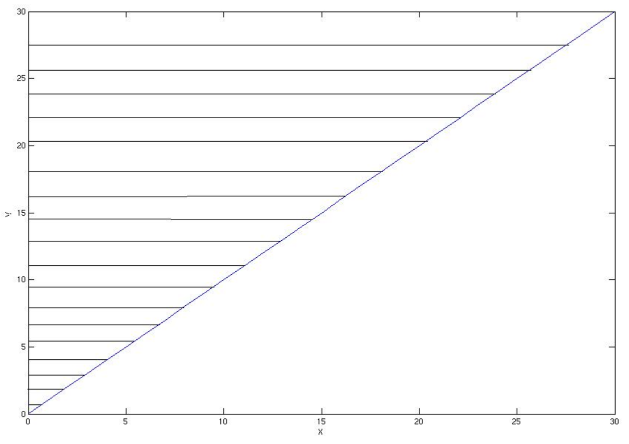

Figure 3.2: Polyhedron for Example 3.3.3 with horizontal axis being $X$ and vertical axis being $Y$

## Discussion

From above, it is clear that abstract interpretation is a very powerful method to obtain linear constraints between program variables; moreover, it can also be automated to detect such properties. As has been demonstrated in the last section, a more precise information leakage may be obtained if linear constraints between program variables is considered during the quantitative information analysis. Therefore, our idea is to use abstract interpretation to get approximate linear constraints between program variables and then use this information to improve the analysis framework of quantitative information flow proposed by Clark, Hunt and Malacaria which starts from Chapter 5.

# Chapter 4

# Lagrange Multiplier Method and Convex Optimization

This chapter introduces the key technique that is used to derive the maximum joint entropy in our work, the Lagrange multiplier method, which is a very powerful mathematical tool to solve constrained optimization problem. Then convex optimization is discussed, since the problem we are trying to solve belongs to this special category of constrained optimization, however, the speciality of our problem is pointed out.

## 4.1 Lagrange Multiplier Method

Lagrange multiplier method is a very popular and useful mechanism for constrained mathematical optimization. It provides a strategy for finding the maximum (minimum) of a function subject to constraints. The basic idea behind it can be seen from geometric point of view: the contour of the objective function we want to maximize(minimize) should touch the boundary of the constraint function in such a way that they are tangent to each other at one point which is exactly the maximum (minimum). This is demonstrated in Figure 4.1 [2].

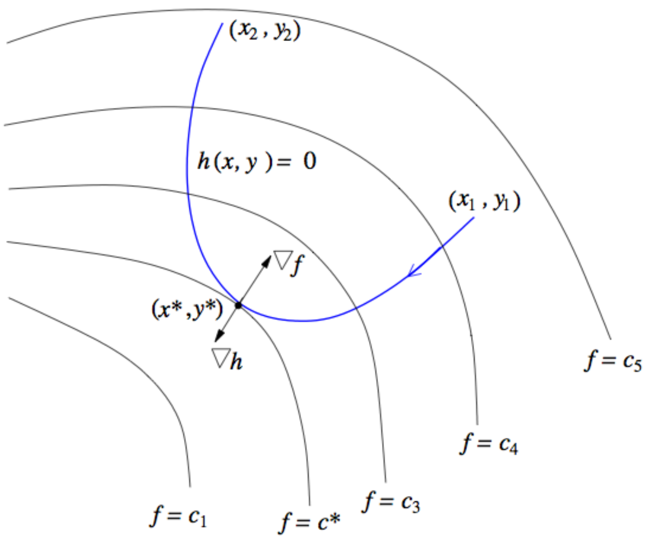

Figure 4.1: Lagrange multiplier Method

Mathematically, suppose we want to maximize the function $f(x, y)$ subject to the constraint $h(x, y) = b$. The method of Lagrange multiplier works by first introducing an auxiliary function $\Lambda(x, y, \lambda)$ to incorporate the constraint into one equation:

$$\Lambda(x, y, \lambda) = f(x, y) - \lambda\,(h(x, y) - b)$$

and solve the partial derivatives of $\Lambda(x, y, \lambda)$ in terms of $x,\ y,\ \lambda$ respectively:

$$\nabla_{x,y,\lambda}\Lambda(x, y, \lambda) = 0$$

Formally, let $\Lambda(x^*, \lambda)$ ($x^*$ is a vector of variables) be the *Lagrangian* of a function $f$ subject to a family of constraints $C_{1\leq i\leq m}$ (where $C_i \equiv h_i(x) = b_i$), *i.e.*

$$\Lambda(x^*, \lambda) = f(x^*) - \sum_{1\leq i\leq m} \lambda_i(h_i(x^*) - b_i)$$

The Lagrange multiplier method is justified by the following theorem:

**Theorem 4.1.1.** *Assume the vector $x^* = (x_1, \ldots, x_n)$ maximizes (or minimizes) the function $f(x)$ subject to the constraints $(h_i(x) = b_i)_{1\leq i\leq m}$. Then either*

1. *the vectors $(\nabla h_i(x^*))_{1\leq i\leq m}$ are linearly dependent, or*

2. *there exists a vector $\lambda^* = (\lambda_1, \ldots, \lambda_m)$ such that* $\nabla\Lambda(x^*, \lambda^*) = 0$, *i.e.*

$$\left(\frac{\partial\Lambda}{\partial x_i}(x^*, \lambda^*) = 0\right)_{1\leq i\leq n}, \quad \left(\frac{\partial\Lambda}{\partial\lambda_i}(x^*, \lambda^*) = 0\right)_{1\leq i\leq m}$$

*where $\nabla$ is the gradient.*

The theorem also applies to functions with more than one variable.

For the technique of Lagrange multiplier method to work, the domain of $f$ should be an open set containing all points satisfying the constraints. Furthermore, $f$ and the $g_i$ must have continuous first partial derivatives and the gradients of the $g_i$ must not be zero on the domain.

### 4.1.1 Simple Example of Lagrange Multiplier Method

Here is an example of how exactly Lagrange multiplier method works, which can be found in mathematics textbooks involving the subject, such as [2]:

Suppose we want to maximize

$$f(x,y) = 10 - (x-5)^2 - (y-3)^2$$

under the constraint $g(x,\ y)$

$$x + y = 1$$

First we rewrite the constraint as

$$g(x,\ y) - c = x + y - 1 = 0$$

Then we construct the Lagrangian:

$$\Lambda(x,\ y,\ \lambda) = f(x,\ y) - \lambda(g(x,\ y) - c) = 10 - (x-5)^2 - (y-3)^2 - \lambda(x+y-1)$$

Note that $\lambda$ may be either added or subtracted although we choose to use subtraction in this thesis.

Partial derivatives of $\Lambda$ of $x,\ y,\ \lambda$ respectively are taken which gives us the following set of equations:

$$\frac{\partial \Lambda}{\partial x} = -2x + 10 - \lambda = 0 \tag{4.1}$$

$$\frac{\partial \Lambda}{\partial y} = -2y + 6 - \lambda = 0 \tag{4.2}$$

$$\frac{\partial\Lambda}{\partial\lambda} = x + y - 1 = 0 \tag{4.3}$$

solving above equations: $\max(f(x, y)) = f(x = \frac{3}{2}, y = -\frac{1}{2}) = -14.5$.

### 4.1.2 Entropy Example

It is well known that uniform distribution gives the maximum entropy and this can also be easily derived by applying Lagrange multiplier method:

The objective function is entropy:

$$f(p_1, p_2, \ldots p_n) = -\sum_{i=1}^{n} p_i \log p_i$$

And there is one simple constraint that

$$g(p_1, p_2, \ldots p_n) - c = \sum_{i=1}^{n} p_i - 1 = 0$$

The Lagrangian is constructed as follows:

$$\Lambda(p_1,\ p_2, \ldots,\ p_n,\ \lambda) = f(p_1,\ p_2, \ldots,\ p_n) - \lambda(g(x,\ y) - c) = -\sum_{i=1}^{n} p_i \log p_i - \lambda(\sum_{i=1}^{n} p_i - 1)$$

For each $p_i$, partial derivative of the Lagrangian gives:

$$(\frac{1}{\ln 2} + \log p_i) + \lambda = 0$$

Thus all the $p_i$ are equal to $\frac{1}{n}$.

## 4.2 Convex Optimization

Convex optimization is a special kind of non-linear optimization. In this section we introduce the definitions of convex optimization followed by a discussion of why our problem cannot be solved using traditional convex optimization algorithms or techniques although it is in the form of convex optimization.

### 4.2.1 Basic Definition

A set $C$ is convex if the line segment between any two points in $C$ lies in $C$, *i.e.,* if for any $x_1,\ x_2 \in C$ and any $0 \leq \lambda \leq 1$, we have

$$\lambda x_1 + (1-\lambda)x_2 \in C.$$

A convex optimization problem is one of the form

$$\text{minimize } f_0(x)$$

$$\text{subject to } f_i(x) \leq b_i,\ i = 1, \ldots, m;\ h_i(x) = 0,\ i = 1, \ldots, p$$

where the functions $f_0, \ldots, f_m :\ R^n \to R$ are convex.

The set of points for which the objective and all constraint functions are defined,

$$\mathcal{D} = \bigcap_{i=0}^{m} \mathrm{dom} f_i \cap \bigcap_{i=1}^{p} \mathrm{dom} h_i$$

is called the domain of the optimization problem. A point $x \in \mathcal{D}$ is *feasible* if it satisfies the constraints $f_i(x) \leq 0,\ i = 1, \ldots, m$, and $h_i(x) = 0, i = 1, \ldots, p$.

The optimization problem is said to be feasible if there exists at least one feasible point, and *infeasible* otherwise.

The optimal value $p^*$ of the problem is defined as

$$p^* = \inf\{f_0(x)|f_i(x) \leq 0,\ i = 1, \ldots, m, h_i(x) = 0,\ i = 1, \ldots, p\}.$$

$x^*$ is an optimal point or solves the problem, if $x^*$ is feasible and $f_0(x^*) = p^*$. The set of all optimal points is the optimal set, denoted

$$X_{\text{opt}} = \{x|f_i(x) \leq 0,\ i = 1, \ldots, m,\ h_i(x) = 0,\ i = 1, \ldots, p,\ f_0(x) = p^*\}$$

Note that the optimal set can be infinite, however in practice, it is enough to just find one such optimal point.

A feasible point $x$ is locally optimal if there is an $R > 0$ such that

$$f(x) = inf\{f_0(z)|f_i(z) \leq 0, i = 1, \ldots, m, h_i(z) = 0, i = 1, \ldots, p, ||z - x||_2 \leq R\},$$

or in other words, $x$ solves the optimization problem

$$\text{minimize } f_0(z)$$

$$\text{subject to } f_i(z) \leq 0, i = 1, \ldots, m\ h_i(z) = 0,\ i = 1, \ldots, p\ ||z - x||_2 \leq R$$

### 4.2.2 Least-squares Problem and Linear Programming

Of convex optimization, the most well known and mature techniques are the least-square problem and linear programming. Unfortunately, our problem doesn't fall into any of the two categories. Here, we basically introduce the definition of each of them.

**Least-square Problem**

A least-square problem is a special subclass of convex optimization with no constraints and an objective which is a sum of squares of terms of the form $a_i^T x - b_i$:

$$\text{minimize } f_0(x) = ||Ax - b||_2^2 = \sum_{i=1}^{k} (a_i^T x - b_i)^2$$

Here $A \in R^{k \times n}$(with $k \geq n$), $a_i^T$ are the rows of $A$, and the vector $x \in R^n$ is the optimization variable.

**Linear Programming**

Linear programming is another important class of convex optimization problems, in which the objective and all constraint functions are linear:

$$\text{minimize } c^T x$$

$$\text{subject to } a_i^T x \leq b_i, \; i = 1, \ldots, m.$$

Here the vectors $c, a_1, \ldots, a_m \in R^n$ and scalars $b_1, \ldots, b_m \in R$ are problem parameters that specify the objective and constraint functions.

### 4.2.3 Numeric Algorithms for Convex Optimization

In this section, we introduce some of the most popular numeric algorithms for solving convex optimization. Let's first start with the most popular one.

**Newton's Method**

Newton's algorithm is the most well-known and popular method for solving unconstrained or equality constraint convex optimization. Here we introduce how the algorithm works:

Given a starting point $x \in \mathrm{dom} f$, tolerance $\epsilon > 0$.

repeat:

1. Compute the Newton step decrement.

$$\triangle x_{nt} := f''(x)^{-1} f'(x); \quad \lambda^2 := f'(x)^T f''(x)^{-1} f'(x).$$

2. Stopping criterion. quit if $\lambda^2/2 \leq \epsilon$.

3. Line search. Choose step size $t$ by backtracking line search.

4. Update. $x := x + t\triangle x_{nt}$.

When working with equality constraint(s), it is necessary that the initial point also satisfies the constraint(s) in addition to belonging to the domain of the objective function $f$.

**Interior-point Method**

Interior-point method is a relatively new algorithm to solve the convex optimization problem with inequality constraints. It also makes use of Newton's method in its centering step. Here we introduce one particular kind of interior-point method, namely the barrier method, whose convergency property is proved in [5]. The algorithm works by incorporating inequality constraints implicitly into the objective function, and then working on a sequence of equality constraint problem.

Suppose we want to minimize $f_0(x)$ subject to $f_i(x) \leq 0,\ i = 1, \ldots, m$ and $Ax = b$, where $f_0, \ldots, f_m : R^n \rightarrow R$ are convex and twice continuously differentiable, and $A \in R^{p \times n}$ with rank $A = p < n$.

The barrier method works by first introducing the logarithmic barrier function $\hat{I}_u = -(1/t)\log(-u)$, then the original problem is approximated as:

minimize $f_0(x) + \sum_{i=1}^{m} -(1/t)\log(-f_i(x))$ subject to $Ax = b$.

Then the algorithm works as follows:

Given strictly feasible $x,\ t := t^{(0)} > 0, \mu > 1,$ tolerance $\epsilon > 0,\ \phi > 0$.

repeat

1. Centering step.

   Compute $x^*(t)$ by minimizing $tf_0 + \phi$, subject to $Ax = b$, starting at $x$.

2. Update. $x = x^*(t)$.

3. Stopping criterion. $q$uit if $m/t < \epsilon$.

4. Increase $t$. $t := \mu t$.

where strictly feasible means that $x^*(t)$ satisfies the constraints $Ax^*(t) = b$ and $f_i(x^*(t)) < 0,\ i = 0, \ldots, m$, and $f_0(x^*(t)) - p^* \leq m/t$. When the initial strictly feasible point is unknown, the algorithm is extended with *phase 1* method in order to find such point to start the algorithm with. For more details please refer to [5].

### 4.2.4 Discussion

Most of the work in convex optimization is concentrated on how to transfer the original problem to convex optimization, hence it can be solved using the corresponding algorithms. These transformations work by introducing new variables or reformulating the original optimization problem, eliminating constraints and *etc.*

Our problem doesn't need to be transformed as it is already in the form of convex optimization. However, in convex optimization when dealing with multi-variables, usually the constraints treat each variable as independent from each other, which is not the case in our problem. Thus, our problem cannot be solved using traditional convex optimization, or in other words, as far as we know in convex optimization, there hasn't been much research into the type of our problem.

With equality constrained convex optimization, the most popular numeric algorithm is Newton's method. However, it depends on providing a feasible start point which for our problem is quite difficult, sometimes even impossible. Although there is an algorithm coping with infeasible starting points, it is only for linear constraints such as $Ax = b$ where $A$ is a $m \times n$ matrix, $x$ is a vector of length $n$, $b$ is a value. Hence our problem is outside the scope of Newton's method.

For the barrier method to work, in addition to the problem of providing feasible initial point, the type of inequality constraint must be of the form $-\mathcal{H} + b \leq 0$, namely we can only work with the lower bound of marginal entropy constraint, which is usually not the case as we either work with equality constraint or upper bound of marginal entropy constraint.

Our algorithm, on the other hand, doesn't require any initial feasible points with which the algorithm iteration starts. Instead, our algorithm searches for a parameter with which the marginal probability distribution satisfies the marginal entropy constraint. Once the parameter is found, the maximal joint entropy is obtained thereafter. From the next chapter we demonstrate how our analytical algorithm works step by step.

# Chapter 5

# Single Constraint Joint Entropy Maximization and Its Generalization

In previous chapters we summarized the current work on quantitative information flow analysis, introduced the background of information theory, abstract interpretation and convex optimization. Most importantly, we demonstrate how Lagrange multiplier method can solve constrained optimization problems using some examples.

This chapter starts an analytical discussion on maximizing information leakage under a single constraint. Here, single constraint means there is only one non-linear constraint (*i.e.* marginal entropy constraint). It doesn't mean there is one and only one constraint, as the precondition that all probabilities sums up to one is always counted as one constraint. Although it is always safe to say that the total amount of information that can be leaked by program variables will not exceed the sum of their self information (entropy), upper bound as precise as possible is always desired. Constrained information leakage poses new challenges to quantitative information flow analysis.

Facing the challenge, we propose a fundamental rigorous algorithm based on Lagrange multiplier method to quantify the maximum possible information that can be leaked by the program. The constraints we consider in this chapter are of four types, linear and non-linear, equality and non-equality. It is a very difficult problem, as it is well known that even if it is a convex optimization, usually there is no analytical form for the maximization problem with non-linear constraint [5]. Investigating the single constraint, we found that by applying the partition version of entropy definition, the derivation can be made much simpler. This enables us to tailor our algorithm to a much easier-to-understand procedure.

## 5.1 Introduction

Our work is based on CHM's framework of quantitative information flow analysis. As discussed in Chapter 3, it suffers from the weakness of "double counting" because it treats programs variables as independent from each other. Recall the simple extreme example from Chapter 3:

$$X := Y;$$
$$Z := X + Y;$$

Without noticing $X$ is just a copy of $Y$, duplicated information is counted twice in $Z$.

Therefore, we are inspired by the fact that relationships between program variables can be used to deduce a more precise upper bound of information leakage. With the increasing complexity of computer systems, the relationships between program variables are very complicated, not only between high and low variables, but between high and high, low and low variables as well. Hence, it is unreasonable to ignore their relationship and treat them as independent from each other.

Hence, we make the first attempt to look into this problem which will be explained in the next few sections.

## 5.2 Quantity to Maximize

Before going into the details of the technique, we first explain the quantity we choose to maximize—the joint entropy.

As demonstrated by Figure 5.1, joint entropy counts for all the information among variables, including the mutual information they share. Therefore, in quantitative information flow analysis, the joint entropy of all the secret inputs stands for the total amount of information that may be leaked. Moreover, the situations we consider here are those when there are more than one high program variables and they interact with each other via various ways. The joint entropy seems to be a quite natural quantity to fit into this situation.

However, there is no conspiracy to choose to maximize joint entropy.

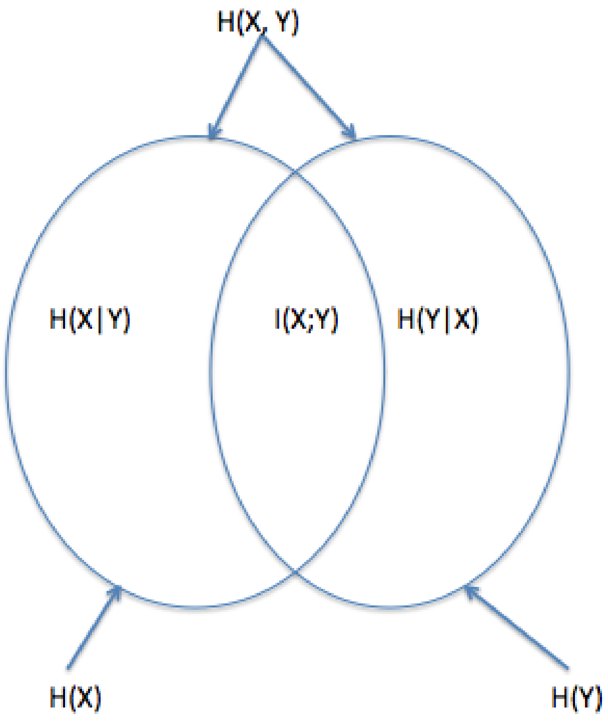

Figure 5.1: Relationship between entropy and mutual information

Others may want to maximize the conditional entropy of observable variable based upon high variables. They are basically two sides of the same coin, as $\mathcal{H}(X,Y) = \mathcal{H}(Y) + \mathcal{H}(X|Y)$ (just take two variables as an example), and the marginal entropy $\mathcal{H}(Y)$ (or $\mathcal{H}(X)$) is fixed in our problem, thus maximizing either of the two terms left will maximize the other one.

In the next section, we first use a concrete example to demonstrate how our algorithm finds out the probability distribution that maximizes the joint entropy under certain constraints before we move on to the more general case.

## 5.3 Simple Problem

Let's consider:

**Example 5.3.1.**

```
**if** $X \leq Y$ **then**
    $Z := X + Y;$
**else**
    $Z := 0;$
**fi**
```

with both $X$ and $Y$ being non-negative integers.

The abstract interpretation analysis is expected to give the following derivation of linear constraints between programs variables:

Suppose before the **if** statement there isn't any constraint on the secret inputs $X$ and $Y$, then in the truth branch after the non-invertible assignment of $Z := X + Y$, it is $\{X \leq Y;\ Z = X + Y\}$, and $\{X \geq Y + 1,\ Z = 0\}$ for the false branch as the assignment is also non-invertible.

The total amount of information leakage would be the sum of three parts together: the information leakage from the guard **if** $X \leq Y$ , from the **then** branch, together with that from the **else** branch.

Suppose $Z$ is the observable output, what is the possible maximum information leakage given the constraints obtained from abstract interpretation analysis? As the maximum leakage of the guard is 1 bit, and $Z := 0$ has information leakage zero, thus we only need to consider the maximum information that can be leaked through $Z := X + Y$ under constraint $X \leq Y$, and this is the type of information leakage that this thesis concerns.

To make derivations as simple as possible in the beginning, let's assume that each of them can only take integer values in the range of [1, 3]. Now let's add a single marginal entropy constraint $\mathcal{H}(Y) = a$ (it is more or less the same if the constraint is $\mathcal{H}(X) = a$ since $X$ and $Y$ are symmetric in this case). Then the question is, what is the maximum possible value for their joint entropy?

In this chapter we use $P$ to denote probability distribution. Let $p_{i,j} = P(X = i, Y = j)$, $q_i = P(Y = i)$. Note that, since $X \leq Y$, we may restrict attention to $p_{i,j}$ such that $i \leq j$. Therefore, the objective function we want to maximize is:

$$\mathcal{H}(X,Y)_{X \leq Y,\ X,Y \in [1,\ 3]} = -\sum_{i,\ j \in [1,\ 3],\ i \leq j} p_{i,j} \log p_{i,j} \tag{5.1}$$

Before constructing the Lagrangian, let's first introduce the partition version of entropy:

Given a distribution $P$ over a set $S = \{s_{1,1},\ s_{1,2},\ \ldots,\ s_{n,m}\}$ and a partition of $S$ into sets $(S_i)_{1 \leq i \leq n}$:

$$\begin{aligned} \mathcal{H}(P(s_{1,1}),\ P(s_{1,2}),\ \ldots,\ P(s_{n,m})) &= \mathcal{H}(P(S_1),\ P(S_2),\ \ldots, P(S_n)) \\ &+ \textstyle\sum_{i=1}^{n} P(S_i)\mathcal{H}(\frac{P(s_{i,1})}{P(S_i)},\ \frac{P(s_{i,2})}{P(S_i)}\ \ldots\ \frac{P(s_{i,m})}{P(S_i)}) \end{aligned}$$

Note that the marginal probability distribution of $Y$ actually imposes a partition on the original set of $(X, Y)$ according to the values of $Y$, therefore, we can rewrite the objective function to embody this:

$$\mathcal{H}(X,Y)_{X \leq Y,\ X,Y \in [1,\ 3]} = \mathcal{H}(q_1,\ q_2,\ q_3) + \sum_{i=1}^{3} q_i \mathcal{H}(\frac{p_{1,i}}{q_i}, \ldots, \frac{p_{i,i}}{q_i}) \tag{5.2}$$

In the above equation, as the marginal entropy for $Y$ is fixed (which is $a$ in this case), in order to get maximal joint entropy, we only need to maximize $\sum_{i=1}^{3} q_i \mathcal{H}(\frac{p_{1,i}}{q_i}, \ldots, \frac{p_{i,i}}{q_i})$. It is well known that uniform distribution maximizes entropy (refer to the example of applying Lagrange multiplier method to entropy in Section 4.1.2), thus for our problem, uniform distributions in each part of $q_i$ maximizes the joint entropy. As a result the above equation can be further reduced to:

$$\mathcal{H}(X,Y)_{X \leq Y,\, X,Y \in [1,\, 3]} = a + \sum_{i=1}^{3} q_i \log i \tag{5.3}$$

Thus it is suffices to maximize equation 5.3 subject to the following two constraints:

$$\mathcal{H}(Y) - a = 0 \tag{5.4}$$

$$\sum_{i=1}^{3} q_i - 1 = 0 \tag{5.5}$$

Now apply the Lagrange multiplier method and define the Lagrangian $\Lambda$ as

$$\Lambda = a + \sum_{i=1}^{3} q_i \log i - \lambda_1(H(Y) - a) - \lambda_2(\sum_{i=1}^{3} q_i - 1) \tag{5.6}$$

which gives rise to the following family of partial derivative equations,

$1 \leq i \leq 3$:

$$\frac{\partial \Lambda}{\partial q_1} = \log 1 + \frac{\lambda_1}{\ln 2}(\ln q_1 + 1) - \lambda_2 = 0 \tag{5.7}$$

$$\frac{\partial \Lambda}{\partial q_2} = \log 2 + \frac{\lambda_1}{\ln 2}(\ln q_2 + 1) - \lambda_2 = 0 \tag{5.8}$$

$$\frac{\partial \Lambda}{\partial q_3} = \log 3 + \frac{\lambda_1}{\ln 2}(\ln q_3 + 1) - \lambda_2 = 0 \tag{5.9}$$

By subtracting equation 5.7 from equation 5.8 and 5.9 respectively, we can derive the following:

$$\log 2 = \log 1 + \frac{\lambda_1}{\ln 2}(\ln q_1 + 1) - \frac{\lambda_1}{\ln 2}(\ln q_2 + 1) \tag{5.10}$$

$$\log 3 = \log 1 + \frac{\lambda_1}{\ln 2}(\ln q_1 + 1) - \frac{\lambda_1}{\ln 2}(\ln q_3 + 1) \tag{5.11}$$

hence

$$\log 2 = \frac{\lambda_1}{\ln 2}(\ln q_1 + 1 - \ln q_2 - 1) = -\lambda_1 \log \frac{q_2}{q_1} \tag{5.12}$$

$$\log 3 = \frac{\lambda_1}{\ln 2}(\ln q_1 + 1 - \ln q_3 - 1) = -\lambda_1 \log \frac{q_3}{q_1} \tag{5.13}$$

Let $\alpha = -\frac{1}{\lambda_1}$ $(\lambda_1 \neq 0)$. Then we have:

$$\log \frac{q_2}{q_1} = \log(2^\alpha) \tag{5.14}$$

$$\log \frac{q_3}{q_1} = \log(3^\alpha) \tag{5.15}$$

hence

$$q_2 = 2^\alpha q_1 \tag{5.16}$$

$$q_3 = 3^\alpha q_1 \tag{5.17}$$

Then from $\sum_1^3 q_i = 1$ we derive:

$$q_1 = \frac{1}{Z(\alpha)} \tag{5.18}$$

$$q_2 = \frac{2^\alpha}{Z(\alpha)} \tag{5.19}$$

$$q_3 = \frac{3^\alpha}{Z(\alpha)} \tag{5.20}$$

where $Z(\alpha) = 1 + 2^\alpha + 3^\alpha$.

We only need to vary $\alpha$ to find suitable a solution satisfying $\mathcal{H}(Y) = a$ as illustrated in Figure 5.2.

There are two values of $\alpha$ for one given marginal entropy, one positive, the other negative, however, we only need the positive $\alpha$, as shown in Figure 5.3, positive $\alpha$ gives larger joint entropy than the corresponding negative one:

Figure 5.2 also shows that marginal entropy monotonically decreases when positive $\alpha$ increases, which is a very useful observation as it allows a very simple search for a suitable $\alpha$ for a given marginal entropy constraint in this region, *i.e.* a binary search does the job very well.

Now let's prove the monotonically decreasing property with positive $\alpha$ in this simple case where $Y \in [1, 2, 3]$:

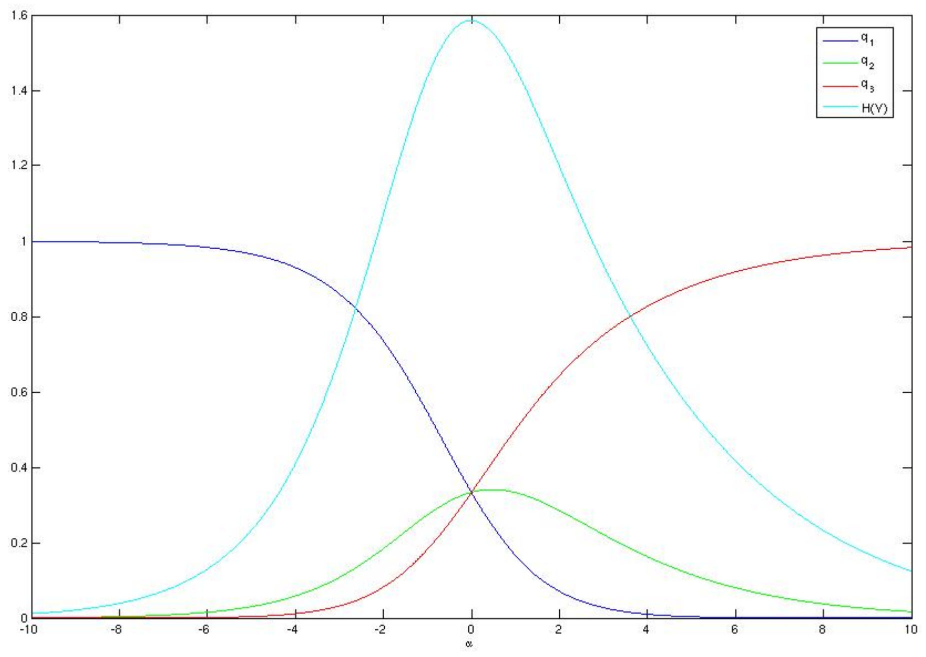

Figure 5.2: Marginal Probability Distribution and Marginal Entropy in terms of $\alpha$

**Proposition 5.3.2.** *Suppose* $Y \in [1, 2, 3]$ *and the probability* $P(Y)$ *of* $Y$ *is* $\{\ \frac{1}{Z(\alpha)},\ \frac{2^{\alpha}}{Z(\alpha)},\ \frac{3^{\alpha}}{Z(\alpha)}\}$, *if* $\ 0 \leq \alpha < \alpha'$, *then* $\mathcal{H}(Y(\alpha')) < \mathcal{H}(Y(\alpha))$ *where* $Z(\alpha) = 1 + 2^{\alpha} + 3^{\alpha}$.

One way of proving that a function monotonically decreases is to show that its first derivative is less than zero, and we take this approach in our proof:

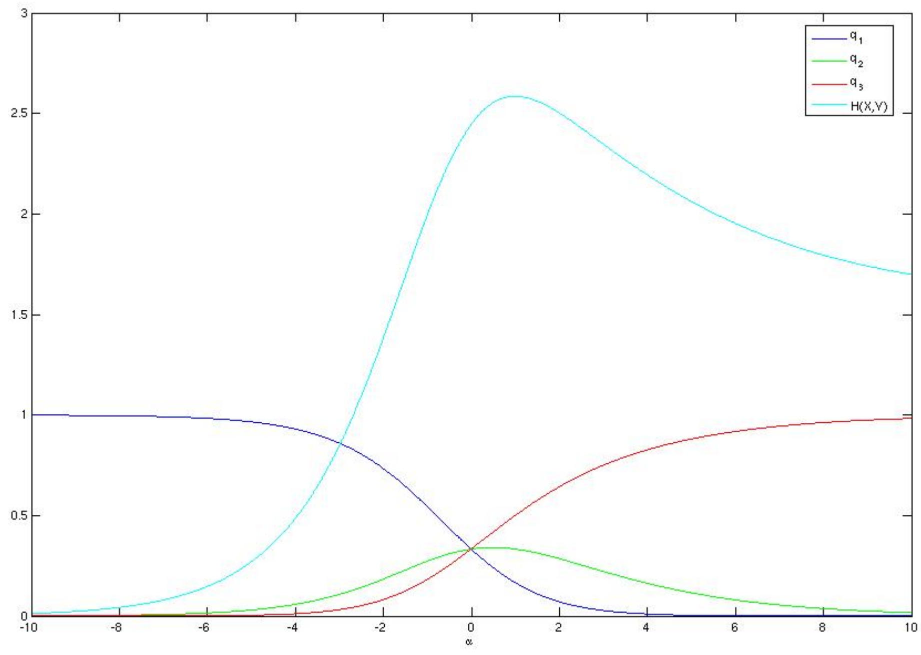

Figure 5.3: Marginal Probability Distribution and Maximal Joint Entropy in terms of $\alpha$

*Proof.*

$$\begin{aligned}
\mathcal{H}(Y(\alpha)) &= -\tfrac{1}{Z(\alpha)}\log(\tfrac{1}{Z(\alpha)}) - \tfrac{2^\alpha}{Z(\alpha)}\log(\tfrac{2^\alpha}{Z(\alpha)}) - \tfrac{3^\alpha}{Z(\alpha)}\log(\tfrac{3^\alpha}{Z(\alpha)}) \\
&= \tfrac{1}{Z(\alpha)}\log(Z(\alpha)) - \tfrac{2^\alpha}{Z(\alpha)}(\log 2^\alpha - \log Z(\alpha)) - \tfrac{3^\alpha}{Z(\alpha)}(\log 3^\alpha - \log Z(\alpha)) \\
&= \tfrac{1}{Z(\alpha)}\log(Z(\alpha)) - \tfrac{2^\alpha}{Z(\alpha)}\log 2^\alpha + \tfrac{2^\alpha}{Z(\alpha)}\log Z(\alpha) - \tfrac{3^\alpha}{Z(\alpha)}\log 3^\alpha + \tfrac{3^\alpha}{Z(\alpha)}\log Z(\alpha) \\
&= \log Z(\alpha) - \tfrac{\alpha 2^\alpha}{Z(\alpha)} - \tfrac{\alpha 3^\alpha \log 3}{Z(\alpha)}
\end{aligned}$$

Its first derivative of $\alpha$ is:

$$\begin{aligned}
\mathcal{H}'(Y(\alpha)) &= \tfrac{Z'(\alpha)}{Z(\alpha)\ln 2} - \tfrac{(\alpha 2^\alpha)' Z(\alpha) - \alpha 2^\alpha Z'(\alpha)}{(Z(\alpha))^2} - \tfrac{(\alpha 3^\alpha \log 3)' Z(\alpha) - \alpha 3^\alpha \log 3 Z'(\alpha)}{(Z(\alpha))^2} \\
&= \tfrac{Z'(\alpha)}{Z(\alpha)\ln 2} - \tfrac{2^\alpha + \alpha 2^\alpha \ln 2 + 3^\alpha \log 3 + \alpha 3^\alpha \ln 3 \log 3}{Z(\alpha)} + \tfrac{(\alpha 2^\alpha + \alpha 3^\alpha \log 3) Z(\alpha)'}{Z(\alpha)^2} \\
&= \tfrac{Z'(\alpha)}{Z(\alpha)\ln 2} - \tfrac{2^\alpha \ln 2 + \alpha 2^\alpha (\ln 2)^2 + 3^\alpha \ln 3 + \alpha 3^\alpha (\ln 3)^2}{Z(\alpha)\ln 2} + \tfrac{\alpha (Z(\alpha)')^2}{(Z(\alpha))^2 \ln 2} \\
&= \tfrac{Z'(\alpha)}{Z(\alpha)\ln 2} - \tfrac{Z'(\alpha) + \alpha 2^\alpha (\ln 2)^2 + \alpha 3^\alpha (\ln 3)^2}{Z(\alpha)\ln 2} + \tfrac{\alpha (Z(\alpha)')^2}{(Z(\alpha))^2 \ln 2} \\
&= \tfrac{\alpha (Z(\alpha)')^2 - \alpha (2^\alpha (\ln 2)^2 + 3^\alpha (\ln 3)^2) Z(\alpha)}{(Z(\alpha))^2 \ln 2}
\end{aligned}$$

As $\alpha > 0$, we only need to compare $\frac{(Z(\alpha)')^2 - (2^\alpha (\ln 2)^2 + 3^\alpha (\ln 3)^2) Z(\alpha)}{(Z(\alpha))^2 \ln 2}$ and zero:

$$\begin{aligned}
&= \tfrac{(2^\alpha \ln 2 + 3^\alpha \ln 3)^2 - (2^\alpha (\ln 2)^2 + 3^\alpha (\ln 3)^2)(1 + 2^\alpha + 3^\alpha)}{(Z(\alpha))^2 \ln 2} \\
&= \tfrac{2^{2\alpha}(\ln 2)^2 + 3^{2\alpha}(\ln 3)^2 + 2 2^\alpha 3^\alpha \ln 2 \ln 3 - 2^\alpha (\ln 2)^2 - 3^\alpha (\ln 3)^2 - 2^{2\alpha}(\ln 2)^2 - 2^\alpha 3^\alpha (\ln 3)^2 - 2^\alpha 3^\alpha (\ln 2)^2 - 3^{2\alpha}(\ln 3)^2}{(Z(\alpha))^2 \ln 2} \\
&= \tfrac{2 2^\alpha 3^\alpha \ln 2 \ln 3 - 2^\alpha (\ln 2)^2 - 3^\alpha (\ln 3)^2 - 2^\alpha 3^\alpha (\ln 3)^2 - 2^\alpha 3^\alpha (\ln 2)^2}{(Z(\alpha))^2 \ln 2} \\
&= \tfrac{-2^\alpha (\ln 2)^2 - 3^\alpha (\ln 3)^2 - 2^\alpha 3^\alpha (\ln 2 - \ln 3)^2}{(Z(\alpha))^2 \ln 2} \\
&< 0
\end{aligned}$$

The first derivative is less than zero, hence the marginal entropy is monotonically decreasing in the range $\alpha \in [0, +\infty)$. □

### 5.3.1 Three High Variables Case of Simple Problem

In last section, there are only two high variables involved, in this section, we show how more than two variables can also be derived using our analysis:

In this section, we use $X_1$, $X_2$ and $X_3$ to represent three high variables such that $X_1 \leq X_2 \leq X_3$ and all of them $\in [1, 3]$, $p_i$, $q_i$, $r_k$ to represent $P(X_1 = i), P(X_2 = j), P(X_3 = k)$ respectively, and with single constraint $\mathcal{H}(X_2) = a$ (it can be either $\mathcal{H}(X_1)$ or $\mathcal{H}(X_3)$).

By using the partition version of the entropy definition, we can get:

$$\mathcal{H}(X_1, X_2, X_3) = \mathcal{H}(X_2) + \sum_{i=1,\, i\leq j,\, j\leq k}^{3} q_j \mathcal{H}(\frac{p_{1,j,j}}{q_j}, \ldots, \frac{p_{j,j,3}}{q_j}) \tag{5.21}$$

Similarly as the two high variable case, the uniform distribution in each part of $q_j$ maximizes the joint entropy. As a result the above equation can be further reduced to:

$$\mathcal{H}(X_1, X_2, X_3) = a + \sum_{j=1}^{3} q_j \log j(4-j) \tag{5.22}$$

Now together with the constraints of $\mathcal{H}(X_2) - a = 0$ and $\sum_{j=1}^{3} q_j - 1 = 0$, the Lagrangian is:

$$\Lambda = a + \sum_{j=1}^{3} q_j \log j(4-j) - \lambda_1(\mathcal{H}(X_2) - a) - \lambda_2(\sum_{j=1}^{3} q_j - 1) \tag{5.23}$$

which gives rise to the following family of partial derivative equations, $1 \leq j \leq 3$:

$$\frac{\partial\Lambda}{\partial q_1} = \log 3 + \frac{\lambda_1}{\ln 2}(\ln q_1 + 1) - \lambda_2 = 0 \tag{5.24}$$

$$\frac{\partial\Lambda}{\partial q_2} = \log 4 + \frac{\lambda_1}{\ln 2}(\ln q_2 + 1) - \lambda_2 = 0 \tag{5.25}$$

$$\frac{\partial\Lambda}{\partial q_3} = \log 3 + \frac{\lambda_1}{\ln 2}(\ln q_3 + 1) - \lambda_2 = 0 \tag{5.26}$$

Then we can derive:

$$q_1 = \frac{3^\alpha}{Z(\alpha)} \tag{5.27}$$

$$q_2 = \frac{4^\alpha}{Z(\alpha)} \tag{5.28}$$

$$q_3 = \frac{3^\alpha}{Z(\alpha)} \tag{5.29}$$

where $Z(\alpha) = 3^\alpha + 4^\alpha + 3^\alpha$.

Therefore, it ends up searching for a suitable $\alpha$, the same as the two high variable case.

The derivation of more high variables is more or less the same as above, hence the result is also similar.

## 5.4 General Case of Simple Problem

Now let's generalize the three-value case in the last section to $n$ values:

More or less the same as above, let $X, Y$ be random variables, each with range $\{1, \ldots, n\}$ and such that $X \leq Y$ and $p_{i,j} = P(X = i, Y = j)$ and let $q_i = P(Y = i)$. Since $X \leq Y$, we may restrict attention to $p_{i,j}$ such that $i \leq j$ only.

Suppose $\mathcal{H}(Y) = \mathcal{H}(\vec{q}) = a$. What is the maximum possible value for $\mathcal{H}(X, Y) = \mathcal{H}(\vec{p})$?

Suppose that a joint distribution satisfies $\mathcal{H}(\vec{q}) = a$ and maximizes $\mathcal{H}(\vec{p})$. The same reasoning as above shows that $p_{i,j} = p_{i',j}$ for all $i, i' \leq j$ and hence that $\mathcal{H}(\vec{p}) = f(\vec{q})$, where:

$$f(\vec{q}) = a + \sum_{i=1}^{n} q_i \log i \tag{5.30}$$

Thus it suffices to maximize $f(\vec{q})$ subject to the constraints:

$$\mathcal{H}(\vec{q}) - a = 0 \tag{5.31}$$

$$\sum_{i=1}^{n} q_i - 1 = 0 \tag{5.32}$$

Define the Lagrangian $\Lambda$ as

$$\Lambda(\vec{q}) = f(\vec{q}) - \lambda_1(\mathcal{H}(\vec{q}) - a) - \lambda_2(\sum_{i=1}^{n} q_i - 1) \tag{5.33}$$

giving rise to the following family of equations, $1 \leq i \leq n$:

$$\frac{\partial \Lambda}{\partial q_i} = \log i + \frac{\lambda_1}{\ln 2}(\ln q_i + 1) - \lambda_2 = 0 \tag{5.34}$$

For $2 \leq i \leq n$ we derive by subtracting partial derivative equation of $q_i$ from that of $q_1$:

$$\log i = \log 1 + \frac{\lambda_1}{\ln 2}(\ln q_1 + 1) - \frac{\lambda_1}{\ln 2}(\ln q_i + 1) \tag{5.35}$$

hence

$$\log i = \frac{\lambda_1}{\ln 2}(\ln q_1 + 1 - \ln q_i - 1) = -\lambda_1 \log \frac{q_i}{q_1} \tag{5.36}$$

Let $\alpha = -\frac{1}{\lambda_1}$ $(\lambda_1 \neq 0)$. Lagrange multiplier method, for $2 \leq i \leq n$

$$\log \frac{q_i}{q_1} = \log(i^{\alpha}) \tag{5.37}$$

hence

$$q_i = i^\alpha q_1 \tag{5.38}$$

Then from $\sum_1^n q_i = 1$ we derive, for $1 \le i \le n$:

$$q_i = \frac{i^\alpha}{Z(\alpha)} \tag{5.39}$$

where $Z(\alpha) = 1 + 2^\alpha + \ldots + n^\alpha$.

Figure 5.4 shows the shape of marginal entropy with different $n$:

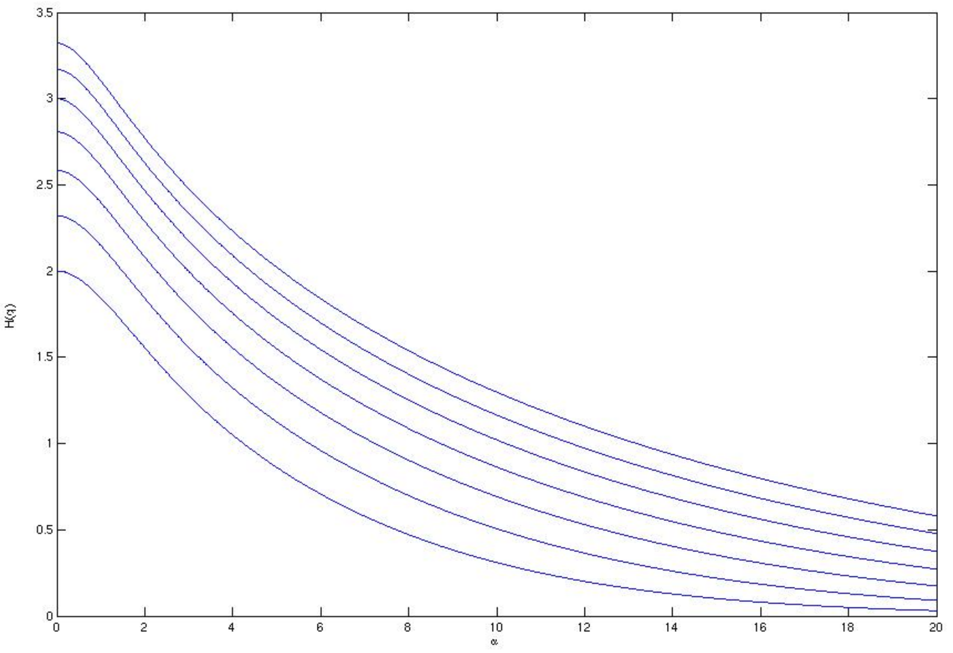

Figure 5.4: Marginal Probability Distributions of Different $n$: the bottom one is $n = 4$, the top one is $n = 10$

Now the same as above, let's prove the property of monotonically decreasing in this general case:

**Proposition 5.4.1.** *Let $Y(\alpha)$ be a distribution for $[1, 2, \ldots, n]$ such that $P(Y(\alpha) = i) = \frac{i^\alpha}{Z(\alpha)}$, where $Z(\alpha) = \sum_i i^\alpha$. If $0 \leq \alpha < \alpha'$, then $\mathcal{H}(Y(\alpha')) < \mathcal{H}(Y(\alpha))$.*

We also show that the first derivative of marginal entropy of $Y$ is less than zero as in the specific three-value case above, before our formal proof, let's prove another proposition which our proof relies on:

**Proposition 5.4.2.** *Suppose $0 < \alpha$, then $(\sum_{i=1}^n i^\alpha \ln i)^2 < \sum_{i=1}^n i^\alpha (\ln i)^2 \sum_{i=1}^n i^\alpha$*

We use mathematical induction to prove this proposition:

*Proof.* When $m = 2$:

$$(\sum_{i=1}^{2} i^\alpha \ln i)^2 = (1 \ln 1 + 2^\alpha \ln 2)^2 = 2^{2\alpha} (\ln 2)^2$$

$$\sum_{i=1}^{2} i^\alpha (\ln i)^2 \sum_{i=1}^{2} i^\alpha = 2^\alpha (\ln 2)^2 (1 + 2^\alpha) = 2^{2\alpha} (\ln 2)^2 + 2^\alpha (\ln 2)^2$$

It is clear that
$$(\sum_{i=1}^{2} i^\alpha \ln i)^2 < \sum_{i=1}^{2} i^\alpha (\ln i)^2 \sum_{i=1}^{2} i^\alpha$$

Thus the conclusion holds when $m = 2$.

Suppose it also holds when $m = k$, that is

$$(\sum_{i=1}^{k} i^\alpha \ln i)^2 < \sum_{i=1}^{k} i^\alpha (\ln i)^2 \sum_{i=1}^{k} i^\alpha$$

When $m = k+1$:

$$\begin{aligned}(\textstyle\sum_{i=1}^{k+1} i^\alpha \ln i)^2 &= \left(\textstyle\sum_{i=1}^{k} i^\alpha \ln i + (k+1)^\alpha \ln(k+1)\right)^2 \\ &= \left(\textstyle\sum_{i=1}^{k} i^\alpha \ln i\right)^2 + 2(k+1)^\alpha \ln(k+1)\textstyle\sum_{i=1}^{k} i^\alpha \ln i + (k+1)^{2\alpha}(\ln(k+1))^2\end{aligned}$$

$$\begin{aligned}\textstyle\sum_{i=1}^{k+1} i^\alpha (\ln i)^2 \sum_{i=1}^{k+1} i^\alpha &= \textstyle\sum_{i=1}^{k+1} i^\alpha (\ln i)^2 \left(\sum_{i=1}^{k} i^\alpha + (k+1)^\alpha\right) \\ &= \textstyle\sum_{i=1}^{k+1} i^\alpha (\ln i)^2 \sum_{i=1}^{k} i^\alpha + (k+1)^\alpha \sum_{i=1}^{k+1} i^\alpha (\ln i)^2 \\ &= \textstyle\sum_{i=1}^{k} i^\alpha (\ln i)^2 \sum_{i=1}^{k} i^\alpha + (k+1)^\alpha (\ln(k+1))^2 \sum_{i=1}^{k} i^\alpha \\ &\quad +(k+1)^\alpha \textstyle\sum_{i=1}^{k+1} i^\alpha (\ln i)^2\end{aligned}$$

Since $(\sum_{i=1}^{k} i^\alpha \ln i)^2 < \sum_{i=1}^{k} i^\alpha (\ln i)^2 \sum_{i=1}^{k} i^\alpha$:

$$(\textstyle\sum_{i=1}^{k+1} i^\alpha \ln i)^2 < \textstyle\sum_{i=1}^{k} i^\alpha (\ln i)^2 \sum_{i=1}^{k} i^\alpha + 2(k+1)^\alpha \ln(k+1) \sum_{i=1}^{k} i^\alpha \ln i + (k+1)^{2\alpha}(\ln(k+1))^2$$

Therefore, we only need to prove that:

$$\begin{aligned}& 2(k+1)^\alpha \ln(k+1) \textstyle\sum_{i=1}^{k} i^\alpha \ln i + (k+1)^{2\alpha}(\ln(k+1))^2 \\ < \ & (k+1)^\alpha (\ln(k+1))^2 \textstyle\sum_{i=1}^{k} i^\alpha + (k+1)^\alpha \sum_{i=1}^{k+1} i^\alpha (\ln i)^2\end{aligned}$$

$$\begin{aligned}& 2(k+1)^\alpha \ln(k+1) \textstyle\sum_{i=1}^{k} i^\alpha \ln i + (k+1)^{2\alpha}(\ln(k+1))^2 \\ = \ & (k+1)^\alpha \left(2\ln(k+1) \textstyle\sum_{i=1}^{k} i^\alpha \ln i + (k+1)^\alpha (\ln(k+1))^2\right)\end{aligned}$$

$$\begin{aligned}& (k+1)^\alpha (\ln(k+1))^2 \textstyle\sum_{i=1}^{k} i^\alpha + (k+1)^\alpha \sum_{i=1}^{k+1} i^\alpha (\ln i)^2 \\ = \ & (k+1)^\alpha \left(\textstyle\sum_{i=1}^{k} i^\alpha (\ln(k+1))^2 + \sum_{i=1}^{k+1} i^\alpha (\ln i)^2\right) \\ = \ & (k+1)^\alpha \left(\textstyle\sum_{i=1}^{k} i^\alpha (\ln(k+1))^2 + \sum_{i=1}^{k} i^\alpha (\ln i)^2 + (k+1)^\alpha (\ln(k+1))^2\right)\end{aligned}$$

Since

$$\begin{aligned}
& \textstyle\sum_{i=1}^{k} i^{\alpha}(\ln(k+1))^2 + \sum_{i=1}^{k} i^{\alpha}(\ln i)^2 - 2\ln(k+1)\sum_{i=1}^{k} i^{\alpha}\ln i \\
= \; & \textstyle\sum_{i=1}^{k} i^{\alpha}\left(\ln(k+1) - \ln i\right)^2 \\
> \; & 0
\end{aligned}$$

Thus

$$\begin{aligned}
& 2(k+1)^{\alpha}\ln(k+1)\textstyle\sum_{i=1}^{k} i^{\alpha}\ln i + (k+1)^{2\alpha}(\ln(k+1))^2 \\
< \; & (k+1)^{\alpha}(\ln(k+1))^2\textstyle\sum_{i=1}^{k} i^{\alpha} + (k+1)^{\alpha}\sum_{i=1}^{k+1} i^{\alpha}(\ln i)^2
\end{aligned}$$

holds.

Therefore, the conclusion holds when $n = k+1$, which is:

$$(\sum_{i=1}^{k+1} i^{\alpha}\ln i)^2 < \sum_{i=1}^{k+1} i^{\alpha}(\ln i)^2 \sum_{i=1}^{k+1} i^{\alpha}$$

As a result, the proposition holds for all $n$. □

Now let's prove Proposition 5.4.1 by showing that the first derivative of marginal entropy is less than zero:

*Proof.*

$$\begin{aligned}
\mathcal{H}(Y(\alpha)) &= -\textstyle\sum_i \frac{i^{\alpha}}{Z(\alpha)}\log\frac{i^{\alpha}}{Z(\alpha)} \\
&= -\textstyle\sum_i \frac{i^{\alpha}}{Z(\alpha)}(\log i^{\alpha} - \log Z(\alpha)) \\
&= -\textstyle\sum_i \frac{i^{\alpha}}{Z(\alpha)}\log i^{\alpha} + \sum_i \frac{i^{\alpha}}{Z(\alpha)}\log Z(\alpha) \\
&= \log Z(\alpha) - \textstyle\sum_i \frac{\alpha i^{\alpha}\log i}{Z(\alpha)}
\end{aligned}$$

Its first derivative of $\alpha$ is:

$$\begin{aligned}
\mathcal{H}(Y(\alpha))' &= \frac{Z(\alpha)'}{Z(\alpha)\ln 2} - \frac{\sum_i (i^\alpha + \alpha i^\alpha \ln i) Z(\alpha)\log i - \alpha i^\alpha Z(\alpha)' \log i}{(Z(\alpha))^2} \\
&= \frac{Z(\alpha)'}{Z(\alpha)\ln 2} - \frac{\sum_i i^\alpha \log i Z(\alpha) + \sum_i \alpha i^\alpha \ln i \log i Z(\alpha) - \sum_i \alpha i^\alpha \log i Z(\alpha)'}{(Z(\alpha))^2} \\
&= \frac{Z(\alpha)'}{Z(\alpha)\ln 2} - \frac{\sum_i i^\alpha \ln i Z(\alpha) + \sum_i \alpha i^\alpha (\ln i)^2 Z(\alpha) - \sum_i \alpha i^\alpha \ln i Z(\alpha)'}{(Z(\alpha))^2 \ln 2} \\
&= \frac{Z(\alpha)' Z(\alpha) - Z(\alpha)' Z(\alpha) + \alpha((Z(\alpha)')^2 - \sum_i i^\alpha (\ln i)^2 Z(\alpha))}{(Z(\alpha))^2 \ln 2} \\
&= \frac{\alpha\Big((\sum_i i^\alpha \ln i)^2 - \sum_i i^\alpha (\ln i)^2 \sum_i i^\alpha\Big)}{(Z(\alpha))^2 \ln 2} \\
&< 0 (as\ \alpha > 0)
\end{aligned}$$

which is just Proposition 5.4.2, hence the proof completes. □

### 5.4.1 General Case of Three High Variables Case of Simple Problem

Now let's generalize the three high variable case in the last section to $n$ values instead of 3:

Similarly, let $X_1$, $X_2$ and $X_3$ be three high variables such that $X_1 \leq X_2 \leq X_3$ and all of them $\in [1, \dots, n]$, $p_i,\ q_i,\ r_k$ to represent $P(X_1 = i), P(X_2 = j), P(X_3 = k)$ respectively, and with single constraint $\mathcal{H}(X_2) = a$ (it can be either $\mathcal{H}(X_1)$ or $\mathcal{H}(X_3)$).

By using the partition version of the entropy definition, we can get:

$$\mathcal{H}(X_1, X_2, X_3) = \mathcal{H}(X_2) + \sum_{j=1,\ i\leq j,\ j\leq k}^{n} q_j \mathcal{H}(\frac{p_{1,j,j}}{q_j}, \dots, \frac{p_{j,j,n}}{q_j}) \tag{5.40}$$

Similarly as the two high variable case, the uniform distribution in each part of $q_j$ maximizes the joint entropy. As a result the above equation can be further reduced to:

$$\mathcal{H}(X_1, X_2, X_3) = a + \sum_{j=1}^{n} q_j \log j(n+1-j) \tag{5.41}$$

Now together with the constraints of $\mathcal{H}(X_2) - a = 0$ and $\sum_{j=1}^{n} q_j - 1 = 0$, the Lagrangian is:

$$\Lambda = a + \sum_{j=1}^{n} q_j \log j(n+1-j) - \lambda_1(\mathcal{H}(X_2) - a) - \lambda_2(\sum_{j=1}^{n} q_j - 1) \tag{5.42}$$

which gives rise to the following family of partial derivative equations, $1 \leq j \leq n$:

- When $n$ is odd

$$\frac{\partial \Lambda}{\partial q_1} = \log n + \frac{\lambda_1}{\ln 2}(\ln q_1 + 1) - \lambda_2 = 0 \tag{5.43}$$

$$\frac{\partial \Lambda}{\partial q_2} = \log 2(n-1) + \frac{\lambda_1}{\ln 2}(\ln q_2 + 1) - \lambda_2 = 0 \tag{5.44}$$

$$\vdots \tag{5.45}$$

$$\frac{\partial \Lambda}{\partial q_{\frac{n-1}{2}}} = \log \frac{n-1}{2}\frac{n+3}{2} + \frac{\lambda_1}{\ln 2}(\ln q_{\frac{n-1}{2}} + 1) - \lambda_2 = 0 \tag{5.46}$$

$$\frac{\partial \Lambda}{\partial q_{\frac{n+1}{2}}} = \log \frac{n+1}{2}\frac{n+1}{2} + \frac{\lambda_1}{\ln 2}(\ln q_{\frac{n+1}{2}} + 1) - \lambda_2 = 0 \tag{5.47}$$

$$\vdots \tag{5.48}$$

$$\frac{\partial \Lambda}{\partial q_n} = \log n + \frac{\lambda_1}{\ln 2}(\ln q_n + 1) - \lambda_2 = 0 \tag{5.49}$$

- When $n$ is even

$$\frac{\partial \Lambda}{\partial q_1} = \log n + \frac{\lambda_1}{\ln 2}(\ln q_1 + 1) - \lambda_2 = 0 \tag{5.50}$$

$$\frac{\partial \Lambda}{\partial q_2} = \log 2(n-1) + \frac{\lambda_1}{\ln 2}(\ln q_2 + 1) - \lambda_2 = 0 \tag{5.51}$$

$$\vdots \tag{5.52}$$

$$\frac{\partial \Lambda}{\partial q_{\frac{n}{2}}} = \log \frac{n}{2}\frac{n}{2} + \frac{\lambda_1}{\ln 2}(\ln q_{\frac{n}{2}} + 1) - \lambda_2 = 0 \tag{5.53}$$

$$\frac{\partial \Lambda}{\partial q_{\frac{n+2}{2}}} = \log \frac{n+2}{2}\frac{n+1}{2} + \frac{\lambda_1}{\ln 2}(\ln q_{\frac{n+1}{2}} + 1) - \lambda_2 = 0 \tag{5.54}$$

$$\vdots \tag{5.55}$$

$$\frac{\partial \Lambda}{\partial q_n} = \log n + \frac{\lambda_1}{\ln 2}(\ln q_n + 1) - \lambda_2 = 0 \tag{5.56}$$

In both cases we can derive :

$$q_i = \frac{(i(n+1-i))^\alpha}{Z(\alpha)} \tag{5.57}$$

where $Z(\alpha) = \sum_{j=1}^{n} (j(n+1-j))^\alpha$.

Therefore, it ends up searching for a suitable $\alpha$, the same as the two high variable case.

The derivation of more high variables is more or less the same as above, and hence the result.

## 5.5 Generalization of Simple Problem

The further generalization of the problem is to generalize the linear constraints between program variables $X_i$ as $X_{i-1} \leq a_i X_i + b_i$ ($a_i,\ b_i \in \mathcal{R},\ i \in$

$\mathcal{N}$). The linear inequality constraint(s) between them can be derived using abstract interpretation, more precisely integer polyhedron. If we consider all the possible tuple of values of $X_1,\ X_2, \ldots, X_m$ as a set, we can divide it into a set of subsets according to the values of $X_i$. Then recall the definition of partition in Chapter 2, this is just a partition on the set according to the values of $X_i$. Let's assume that the sizes of each part of the partition are $n_1,\ n_2,\ \ldots,\ n_k$ respectively.

In other words, suppose $X$ and $Y$ are two random variables. Let $p_{i,j} = P(X = i, Y = j)$ and $q_i = P(Y = i)$, the problem is: suppose $\mathcal{H}(Y) = a$, what is the maximum possible value for $\mathcal{H}(X, Y) = \mathcal{H}(\vec{p})$?

More formally,

**Definition 5.5.1.** *Let $A$ be a finite set and let $\mathcal{P}$ be a set of distributions on $A$. We write $\mathcal{H}_{\max}(\mathcal{P})$ to mean the maximum entropy of all distributions in the set: $\mathcal{H}_{\max}(\mathcal{P}) = \sup_{P\in\mathcal{P}} \mathcal{H}(P)$.*

**Theorem 5.5.2.** *Let $A = \{1, \ldots, n\}$ and let $\{Q_j\}_{j\in J}$ be a partition of $A$. Given a distribution $p = \{p_1, \ldots, p_n\}$ on $A$, let $q[p]$ be the distribution induced on $J$ by the partition: $q[p]_j = \sum_{i\in Q_j} p_i$; call this the* marginal distribution *for $p$. Given $0 \le a \le \log |J|$, let $\mathcal{P}_a$ be the set of all distributions whose marginal distribution has $a$ as its entropy: $\mathcal{P}_a = \{p|\mathcal{H}(q[p]) = a\}$. Let $k$ be the size of the largest part of the partition ($k = \max_{j\in J} |Q_j|$) and let $m$ be the number of parts having size $k$ (hence $1 \le m \le |J|$). Then there are two cases:*

1. *if $a \le \log m$ then $\mathcal{H}_{\max}(\mathcal{P}_a) = a + \log k$*

2. *if $a > \log m$ then there exists a unique $\alpha \in [0; \infty)$ such that $\mathcal{H}(q(\alpha)) =$*

*a, where*

$$q(\alpha)_j = \frac{|Q_j|^\alpha}{Z(\alpha)}$$

*with* $Z(\alpha) = \sum_{j \in J} |Q_j|^\alpha$*; in this case*

$$\mathcal{H}_{\max}(\mathcal{P}_a) = a + \sum_{j \in J} q(\alpha)_j \log |Q_j|$$

In this general case, we need to consider the range of the value of marginal entropy constraint, which didn't come up in the last two sections. This is simply because in the special case of three-value and its generalized $n$-value situation above, there is only one part having the largest size (actually each part in the partition has different size), therefore $m = 1 \rightarrow \log m = 0$, as marginal entropy cannot be less than zero, we do not need to consider the range of its values.

We can observe from the special cases that the larger the size of the part is, the greater the contribution it makes to the joint entropy. Thus if there is any part(s) having zero probability, that should be of the smallest size(s).

However, in the current more general case, there is possibility that there are several parts having the same largest size of the partition. Suppose the number of such parts is $m$, then if marginal entropy of the probability distribution of such partition is less than or equal to $\log m$, which is the break point when all the largest size parts have equal probability while all the other smaller parts having zero probability (in this case the marginal entropy is exactly $\log m$). If the marginal entropy is less than $\log m$, this means that there are even some largest size part(s) (not all) have zero probability.

Or from a purely mathematical point of view, when $\lambda_1$ is zero, there is no $\alpha$ such that $\alpha = -\frac{1}{\lambda_1}$. However, actually under such circumstance, $\alpha \to \infty$ makes the largest size part having non-zero probability; if there are $m$ such parts then they all have the same probability of $\frac{1}{m}$. In terms of marginal entropy, it is exactly the breaking point of $\log m$.

Thus we need to treat this case separately. Luckily in this case, the proof when $a \leq \log m$ is quite simple, and the partition version of the joint entropy (refer to Section 5.3) is all that is needed:

*Proof.*

$$\begin{aligned} \mathcal{H}_{\max}(\mathcal{P}_a) &= a + \textstyle\sum_{j \in J} q(\alpha)_j \log |Q_j| \\ &\leq a + \textstyle\sum_{j \in J} q(\alpha)_j \log k \\ &= a + \log k \end{aligned}$$

□

This is actually an upper bound for any partition although it can be viewed as a special case of our conclusion.

The whole proof procedure for the second situation is more or less the same as that for the special cases in the last two sections:

Suppose that a joint distribution satisfies $\mathcal{H}(\vec{q}) = a$ and maximizes $\mathcal{H}(\vec{p})$. The original Lagrangian is as follows:

$$\Lambda = -\sum_{i,j} p_{i,j} \log p_{i,j} - \lambda_1 (-\sum_j q(\alpha)_j \log q(\alpha)_j - a) - \lambda_2 (\sum_{i,j} p_{i,j} - 1) \quad (5.58)$$

Partial derivative of $p_{i,j}$ respectively gives the following family of equations:

$$\frac{1}{\ln 2} + \log p_{i,j} + \lambda_1(\frac{1}{\ln 2} + \log q(\alpha)_j) - \lambda_2 = 0 \tag{5.59}$$

Therefore, for $\forall i, i'$ such that $p_{i,j}$, $p_{i',j}$ belong to the same part of the partition, $p_{i,j} = p_{i',j}$. Using the partition version of entropy we can rewrite our objective function as $\mathcal{H}(Y) + \sum_j q(\alpha)_j \log |Q_j| \ = a + \sum_j q(\alpha)_j \log |Q_j|$. Now applying the lagrange multiplier technique to this deduced function:

$$\Lambda = a + \sum_j q(\alpha)_j \log |Q_j| \ - \lambda_1(-\sum_j q(\alpha)_j \log q(\alpha)_j - a) - \lambda_2(\sum_j q(\alpha)_j - 1) \tag{5.60}$$

Partial derivative of each $q(\alpha)_j$ gives rise to the following set of equations:

$$\log |Q_j| + \lambda_1(\frac{1}{\ln 2} + \log q(\alpha)_j) - \lambda_2 = 0 \tag{5.61}$$

subtracting equations can result in:

$$\log \frac{|Q_j|^\alpha}{|Q_{j-1}|^\alpha} = \log \frac{q(\alpha)_i}{q(\alpha)_{i-1}} \tag{5.62}$$

where $\alpha = -\frac{1}{\lambda_1}$ ($\lambda_1 \neq 0$). Hence

$$q(\alpha)_j = \frac{|Q_j|^\alpha}{Z(\alpha)} \tag{5.63}$$

with $Z(\alpha) = \sum_{j=1}^{k} |Q_j|^\alpha$.

In this case, the proof for the property of monotonically decreasing of marginal entropy with positive $\alpha$, is more or less the same as that in the last section. Thus, a binary search also works for finding suitable $\alpha$ for maximum joint entropy in this general case. We now give the formal proof for the property of monotonically decreasing of marginal entropy in this general case:

**Proposition 5.5.3.** *Suppose* $0 < \alpha$, *then* $(\sum_{i=1}^{m} n_i^\alpha \ln n_i)^2 \leq \sum_{i=1}^{m} n_i^\alpha (\ln n_i)^2 \sum_{i=1}^{m} n_i^\alpha$ *where* $n_i$ *is the size of the* $i$*th part of the partition.*

We use mathematical induction to prove this proposition the same as above, but first we prove the proposition that our proof will relies on:

*Proof.* When $m = 2$:

$$(\sum_{i=1}^{2} n_i^\alpha \ln n_i)^2 = n_1^{2\alpha}(\ln n_1)^2 + 2n_1^\alpha n_2^\alpha \ln n_1 \ln n_2 + n_2^{2\alpha}(\ln n_2)^2$$

$$\begin{aligned}\textstyle\sum_{i=1}^{2} n_i^\alpha (\ln n_i)^2 \sum_{i=1}^{2} n_i^\alpha &= \left(n_1^\alpha(\ln n_1)^2 + n_2^\alpha(\ln n_2)^2\right)\left(n_1^\alpha + n_2^\alpha\right) \\ &= n_1^{2\alpha}(\ln n_1)^2 + n_1^\alpha n_2^\alpha(\ln n_2)^2 + n_1^\alpha n_2^\alpha(\ln n_1)^2 + n_2^{2\alpha}(\ln n_2)^2\end{aligned}$$

As

$$n_1^\alpha n_2^\alpha(\ln n_2)^2 + n_1^\alpha n_2^\alpha(\ln n_1)^2 - 2n_1^\alpha n_2^\alpha \ln n_1 \ln n_2 = n_1^\alpha n_2^\alpha(\ln n_1 - \ln n_2)^2 \geq 0$$

(with equality only when $n_1 = n_2$), thus

$$(\sum_{i=1}^{2} n_i^\alpha \ln n_i)^2 \leq \sum_{i=1}^{2} n_i^\alpha (\ln n_i)^2 \sum_{i=1}^{2} n_i^\alpha$$

holds.

As a result, the proposition holds when $m = 2$.

Now suppose it also holds when $m = k$, that is

$$(\sum_{i=1}^{k} n_i^\alpha \ln n_i)^2 \leq \sum_{i=1}^{k} n_i^\alpha (\ln n_i)^2 \sum_{i=1}^{k} n_i^\alpha$$

When $m = k + 1$:

$$\begin{aligned}(\textstyle\sum_{i=1}^{k+1} n_i^\alpha \ln n_i)^2 &= \left(\textstyle\sum_{i=1}^{k} n_i^\alpha \ln n_i + n_{k+1}^\alpha \ln n_{k+1}\right)^2 \\ &= \left(\textstyle\sum_{i=1}^{k} n_i^\alpha \ln n_i\right)^2 + 2n_{k+1}^\alpha \ln n_{k+1} \textstyle\sum_{i=1}^{k} n_i^\alpha \ln n_i \\ &\quad + n_{k+1}^{2\alpha} (\ln n_{k+1})^2\end{aligned}$$

$$\begin{aligned}\textstyle\sum_{i=1}^{k+1} n_i^\alpha (\ln n_i)^2 \sum_{i=1}^{k+1} n_i^\alpha &= \textstyle\sum_{i=1}^{k+1} n_i^\alpha (\ln n_i)^2 \left(\sum_{i=1}^{k} n_i^\alpha + n_{k+1}^\alpha\right) \\ &= \textstyle\sum_{i=1}^{k+1} n_i^\alpha (\ln n_i)^2 \sum_{i=1}^{k} n_i^\alpha + n_{k+1}^\alpha \sum_{i=1}^{k+1} n_i^\alpha (\ln n_i)^2 \\ &= \textstyle\sum_{i=1}^{k} n_i^\alpha (\ln n_i)^2 \sum_{i=1}^{k} n_i^\alpha + n_{k+1}^\alpha (\ln n_{k+1})^2 \sum_{i=1}^{k} n_i^\alpha \\ &\quad + n_{k+1}^\alpha \textstyle\sum_{i=1}^{k+1} n_i^\alpha (\ln n_i)^2\end{aligned}$$

Since $(\sum_{i=1}^{k} n_i^\alpha \ln n_i)^2 \leq \sum_{i=1}^{k} n_i^\alpha (\ln n_i)^2 \sum_{i=1}^{k} n_i^\alpha$:

$$\begin{aligned}(\textstyle\sum_{i=1}^{k+1} n_i^\alpha \ln n_i)^2 &\leq \textstyle\sum_{i=1}^{k} n_i^\alpha (\ln n_i)^2 \sum_{i=1}^{k} n_i^\alpha + 2n_{k+1}^\alpha \ln n_{k+1} \sum_{i=1}^{k} n_i^\alpha \ln n_i \\ &\quad + n_{k+1}^{2\alpha} (\ln n_{k+1})^2\end{aligned}$$

Therefore, we only need to prove that:

$$\begin{aligned}&2n_{k+1}^\alpha \ln n_{k+1} \textstyle\sum_{i=1}^{k} n_i^\alpha \ln n_i + n_{k+1}^{2\alpha} (\ln n_{k+1})^2 \\ \leq\ &n_{k+1}^\alpha (\ln n_{k+1})^2 \textstyle\sum_{i=1}^{k} n_i^\alpha + n_{k+1}^\alpha \sum_{i=1}^{k+1} n_i^\alpha (\ln n_i)^2\end{aligned}$$

$$\begin{aligned} & 2n_{k+1}^{\alpha}\ln n_{k+1}\textstyle\sum_{i=1}^{k} n_i^{\alpha}\ln n_i + n_{k+1}^{2\alpha}(\ln n_{k+1})^2 \\ = \; & n_{k+1}^{\alpha}\left(2\ln n_{k+1}\textstyle\sum_{i=1}^{k} n_i^{\alpha}\ln n_i + n_{k+1}^{\alpha}(\ln n_{k+1})^2\right) \end{aligned}$$

$$\begin{aligned} & n_{k+1}^{\alpha}(\ln n_{k+1})^2\textstyle\sum_{i=1}^{k} n_i^{\alpha} + n_{k+1}^{\alpha}\textstyle\sum_{i=1}^{k+1} n_i^{\alpha}(\ln n_i)^2 \\ = \; & n_{k+1}^{\alpha}\left(\textstyle\sum_{i=1}^{k} n_i^{\alpha}(\ln n_{k+1})^2 + \textstyle\sum_{i=1}^{k+1} n_i^{\alpha}(\ln n_i)^2\right) \\ = \; & n_{k+1}^{\alpha}\left(\textstyle\sum_{i=1}^{k} n_i^{\alpha}(\ln n_{k+1})^2 + \textstyle\sum_{i=1}^{k} n_i^{\alpha}(\ln n_i)^2 + n_{k+1}^{\alpha}(\ln n_{k+1})^2\right) \end{aligned}$$

Since

$$\begin{aligned} & \textstyle\sum_{i=1}^{k} n_i^{\alpha}(\ln n_{k+1})^2 + \textstyle\sum_{i=1}^{k} n_i^{\alpha}(\ln n_i)^2 - 2\ln n_{k+1}\textstyle\sum_{i=1}^{k} n_i^{\alpha}\ln n_i \\ = \; & \textstyle\sum_{i=1}^{k} n_i^{\alpha}\left(\ln n_{k+1} - \ln n_i\right)^2 \\ \geq \; & 0 \end{aligned}$$

Thus

$$\begin{aligned} & 2n_{k+1}^{\alpha}\ln n_{k+1}\textstyle\sum_{i=1}^{k} n_i^{\alpha}\ln n_i + n_{k+1}^{2\alpha}(\ln n_{k+1})^2 \\ \leq \; & n_{k+1}^{\alpha}(\ln n_{k+1})^2\textstyle\sum_{i=1}^{k} n_i^{\alpha} + n_{k+1}^{\alpha}\textstyle\sum_{i=1}^{k+1} n_i^{\alpha}(\ln n_i)^2 \end{aligned}$$

holds.

Therefore, the conclusion holds when $m = k + 1$, which is:

$$(\sum_{i=1}^{k+1} n_i^{\alpha}\ln n_i)^2 \leq \sum_{i=1}^{k+1} n_i^{\alpha}(\ln n_i)^2 \sum_{i=1}^{k+1} n_i^{\alpha}$$

As a result, the proposition holds for all $n$. Equality only when $n_1 = \ldots = n_m$. $\square$

Now let's prove the property of monotonically decreasing:

**Proposition 5.5.4.** *Suppose $Y$ is an integer variable and the probability*

*$P(Y = i)$ of $Y$ is $\{\frac{n_i^\alpha}{Z(\alpha)}\}$, if $0 \leq \alpha < \alpha'$, then $\mathcal{H}(Y(\alpha')) \leq \mathcal{H}(Y(\alpha))$ where $n_i$ is the size of the $i$th part of the partition and $Z(\alpha) = \sum_i n_i^\alpha$.*

Proposition 5.4.1 is proved by showing that the first derivative of marginal entropy is less than or equal to zero:

*Proof.*

$$\begin{aligned}
\mathcal{H}(Y(\alpha)) &= -\textstyle\sum_i \frac{n_i^\alpha}{Z(\alpha)} \log \frac{n_i^\alpha}{Z(\alpha)} \\
&= -\textstyle\sum_i \frac{n_i^\alpha}{Z(\alpha)} (\log n_i^\alpha - \log Z(\alpha)) \\
&= -\textstyle\sum_i \frac{n_i^\alpha}{Z(\alpha)} \log n_i^\alpha + \sum_i \frac{n_i^\alpha}{Z(\alpha)} \log Z(\alpha) \\
&= \log Z(\alpha) - \textstyle\sum_i \frac{\alpha n_i^\alpha \log n_i}{Z(\alpha)}
\end{aligned}$$

Its first derivative of $\alpha$ is:

$$\begin{aligned}
\mathcal{H}(Y(\alpha))' &= \tfrac{Z(\alpha)'}{Z(\alpha)\ln 2} - \tfrac{\sum_i (n_i^\alpha + \alpha n_i^\alpha \ln n_i) Z(\alpha) \log n_i - \alpha n_i^\alpha Z(\alpha)' \log n_i}{(Z(\alpha))^2} \\
&= \tfrac{Z(\alpha)'}{Z(\alpha)\ln 2} - \tfrac{\sum_i n_i^\alpha \log n_i Z(\alpha) + \sum_i \alpha n_i^\alpha \ln n_i \log n_i Z(\alpha) - \sum_i \alpha n_i^\alpha \log n_i Z(\alpha)'}{(Z(\alpha))^2} \\
&= \tfrac{Z(\alpha)'}{Z(\alpha)\ln 2} - \tfrac{\sum_i n_i^\alpha \ln n_i Z(\alpha) + \sum_i \alpha n_i^\alpha (\ln n_i)^2 Z(\alpha) - \sum_i \alpha n_i^\alpha \ln n_i Z(\alpha)'}{(Z(\alpha))^2 \ln 2} \\
&= \tfrac{Z(\alpha)' Z(\alpha) - Z(\alpha)' Z(\alpha) + \alpha((Z(\alpha)')^2 - \sum_i n_i^\alpha (\ln n_i)^2 Z(\alpha))}{(Z(\alpha))^2 \ln 2} \\
&= \tfrac{\alpha\big((\sum_i n_i^\alpha \ln n_i)^2 - \sum_i n_i^\alpha (\ln n_i)^2 \sum_i n_i^\alpha\big)}{(Z(\alpha))^2 \ln 2} \\
&\leq 0 (as\ \alpha > 0)
\end{aligned}$$

which is just Proposition 5.5.3, hence the proof completes. Equality only holds when all the parts of the partition are of the same size. □

## 5.6 Case Study and Comparison with CHM's Framework

Besides the simple example of $X := Y;\ Z := X + Y$; which already demonstrates that our algorithm is capable of deducing more precise upper bounds of information leakage, in this section we give detailed analyses of some other programs and compare them with CHM's framework.

### 5.6.1 Program with One Linear Constraint

Consider the program in Section 5.3 again with $X$ and $Y$ both high program variables where each has a positive integer domain of $[1, \ldots, n]$ and $Z$ is the low observable variable, that is:

**Example 5.6.1.**

```
if $X \leq Y$ then
    $Z := X + Y$;
else
    $Z := 0$;
```

Suppose $\mathcal{H}(Y) = a$, what is the possible maximum leakage that can be leaked by $Z := X + Y$?

**CHM's Analysis**

Using CHM's framework, only the [If](2) reference rule can apply, which states that

$$\text{If(2)}\frac{\Gamma \vdash B : [0,0] \ \ \vdash \Gamma\{C_i\}x : [a_i,\ b_i]}{\vdash \Gamma\{\textbf{if}\ B\ C_1\ C_2\}x : [\min(a_1,a_2),\max(b_1,b_2)]}$$

As neither the guard $X \leq Y$ nor the **else** branch $Z := 0$ leaks any information, all the information that can be leaked by this **if** statement is via its **then** branch, therefore, the maximum information that can be leaked by the above example is maximum $\mathcal{H}(Z)$ in the **then** branch.

Now by using the [Plus] inference rule:

$$\text{Plus}\frac{\Gamma \vdash E_i : [,b_i]}{\Gamma \vdash (E_1 + E_2) : [0, b_1 + b_2]}$$

Given $\mathcal{H}_{\max}(Y) = a$, as there is no other constraint on the maximum entropy of $X$, its maximum of $\log n$ can be assumed, thus $\mathcal{H}_{\max}(X + Y) = a + \log n$ which universally holds.

Finally by applying the assignment inference rule [Ass]:

$$\text{Ass}\frac{\Gamma \vdash E : [a,\ b]}{\vdash \Gamma\{x := E\}x : [a,b]}$$

we can infer that the maximum leakage into $Z$ is the same as $\mathcal{H}_{\max}(X+Y)$ which is $a + \log n$.

**Our Analysis**

If we build our analysis into the framework of CHM's, we can derive a much more precise upper bound, and here is how to do it:

First the abstract interpretation is expected to give the following linear constraints (although this example is so simple that even without abstract interpretation we can still derive the right linear constraints):

Suppose there isn't any constraint of $X$ and $Y$ before the **if** statement, in the **then** branch, it is $X \leq Y,\ Z = X + Y$ while in the **else** branch it is $X \geq Y + 1,\ Z = 0$.

Now just as above, using CHM's inference rule [If](2), the maximum information that can be leaked by the above example is $1 + \mathcal{H}_{\max}(Z)$, thus we only need to find out maximum $\mathcal{H}(Z)$ in the **then** branch. Using our algorithm which is explained in detail in Section 5.1-5.5 of this chapter, if we divide the set of $(X, Y)$ according to the values that $Y$ takes, then it is a partition of sizes $\{1,\ 2, \ldots, n\}$.

Recall that the marginal entropy constraint of $Y$ can be achieved by constructing a marginal probability distribution $P(Y = i) = q_i$ which is only based on the sizes of the parts and a parameter $\alpha$:

$$q_i = \frac{i^\alpha}{Z(\alpha)}$$

where $Z(\alpha) = \sum_i i^\alpha$ and using the partition version of entropy (refer to Section 5.3), the maximum entropy that $Z$ can have is:

$$\mathcal{H}(Z) = \mathcal{H}(Y) + q_i \log i$$

| $\mathcal{H}(Y)$ | $\alpha$ | $P(Y)$ | $\mathcal{H}_{\max}(Z)$ | $\mathcal{H}(Y) + \log n$ |
|---|---|---|---|---|
| 0.2 | 15.5386 | $\{0.0000, 0.0000, 0.0003, 0.0302, 0.9694\}$ | 2.5119 | 2.5219 |
| 0.4 | 11.4449 | $\{0.0000, 0.0000, 0.0027, 0.0720, 0.9253\}$ | 2.6968 | 2.7219 |
| 0.6 | 8.9541 | $\{0.0000, 0.0002, 0.0090, 0.1183, 0.8725\}$ | 2.8769 | 2.9219 |
| 0.8 | 7.1533 | $\{0.0000, 0.0012, 0.0210, 0.1648, 0.8130\}$ | 3.0518 | 3.1219 |
| 1 | 5.7495 | $\{0.0001, 0.0039, 0.0397, 0.2076, 0.7488\}$ | 3.2206 | 3.3219 |
| 1.2 | 4.6085 | $\{0.0004, 0.0100, 0.0647, 0.2436, 0.6813\}$ | 3.3817 | 3.5219 |
| 1.4 | 3.6568 | $\{0.0017, 0.0215, 0.0945, 0.2705, 0.6118\}$ | 3.5329 | 3.7219 |
| 1.6 | 2.8472 | $\{0.0055, 0.0399, 0.1264, 0.2868, 0.5414\}$ | 3.6709 | 3.9219 |
| 1.8 | 2.1411 | $\{0.0150, 0.0661, 0.1574, 0.2915, 0.4700\}$ | 3.7899 | 4.1219 |
| 2 | 1.4931 | $\{0.0358, 0.1006, 0.1845, 0.2835, 0.3956\}$ | 3.8785 | 4.3219 |
| 2.1493 | 1 | $\{0.0667, 0.1334, 0.2000, 0.2666, 0.3333\}$ | 3.9068 | 4.4712 |
| 2.3219 | 0 | $\{0.2, 0.2, 0.2, 0.2, 0.2\}$ | 3.7033 | 4.6439 |

Table 5.1: Maximum information leakage for Example 5.7.1 with $n = 5$

Thus, instead of adding the upper bounds of entropies of $X$ and $Y$ together, the maximum information that can be leaked to $Z$ is the maximum joint entropy of $X$ and $Y$ which equals $a + \sum_{i=1}^{n} \frac{i^\alpha}{Z(\alpha)} \log i$ and it is strictly $< a + \log n$.

Here we don't have to explicitly using separate [Plus] and [Ass] rules, as our algorithm unifies the assignment whose right hand side can involve addition, subtraction, multiplication and division.

### Concrete Results

Now we give some concrete numerical results for different marginal entropy $\mathcal{H}(Y)$ for given $n$, Table 5.1 is when $n = 5$, note in this case the sizes of parts are $\{1, 2, 3, 4, 5\}$; Table 5.2 are for $n = 10$.

In this table, the column of $\mathcal{H}(Y)$ is the value of the marginal entropy constraint, we choose to start from 0.2 and increase to 2 on an interval of 0.2 (please note that there is nothing special to choose these values, other values can also be chosen as well). The last two values (2.1493 and 2.3219) are chosen on purpose, as one (2.1493) gives the largest value of $\mathcal{H}_{\max}(Z)$ and the other corresponding to the maximum marginal entropy $\mathcal{H}(Y)$, however, as shown in the table, it won't give us the maximum information leakage of $Z$.

The column $\alpha$ is the value of $\alpha$ that is computed using our algorithm, under the corresponding marginal entropy constraint value of $\mathcal{H}(Y)$ and the linear constraint of $X \leq Y$ in this example, for example, if $\mathcal{H}(Y) = 0.2$, then according to our analysis, the probability distribution which satisfies this marginal entropy value and the constraint that $X \leq Y$ is: $q_i = \frac{i^\alpha}{Z(\alpha)}$, as here $n = 5$ which means $Y \in [1, \ldots, 5]$, the problem we need to solve is $0.2 = \sum_{i=1}^{5} -q_i \log q_i$, we use binary search to solve $\alpha$, starting from range [0, 20] (with precision $10^{-4}$), we stop by when $|\mathcal{H}(Y) - 0.2| \leq 10^{-3}$.

Once the $\alpha$ value is known, the column of $P(Y)$ (the probability of $Y$ which gives the maximum entropy) is just the calculation of $\frac{i^\alpha}{Z(\alpha)}$.

The column $\mathcal{H}_{\max}(Z)$ is the value of $\mathcal{H}(Y) + \sum_i q_i \log i$ where $q_i$ are the corresponding value under $P(Y)$ and $i \in [1, \ldots, 5]$.

The column $\mathcal{H}(Y) + \log n = \mathcal{H}(Y) + \log 5$ thus it varies with $\mathcal{H}(Y)$.

The explanation for Table 5.2 is exactly the same as for 5.1 except that here $n = 10$ not 5.

### 5.6.2 Program with Multiple Linear Constraints

Now let's consider programs with more complex linear constraints (Section 3.3):

| $\mathcal{H}(Y)$ | $\alpha$ | $P(Y)$ | $\mathcal{H}_{\max}(Z)$ | $\mathcal{H}(Y) + \log n$ |
|---|---|---|---|---|
| 0.1 | 41.2661 | $\{0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0001, 0.0128, 0.9871\}$ | 3.4200 | 3.4219 |
| 0.4 | 24.6453 | $\{0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0001, 0.0038, 0.0691, 0.9270\}$ | 3.7101 | 3.7219 |
| 0.7 | 17.6469 | $\{0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0001, 0.0016, 0.0166, 0.1323, 0.8494\}$ | 3.9956 | 4.0219 |
| 1 | 13.1881 | $\{0.0000, 0.0000, 0.0000, 0.0000, 0.0001, 0.0009, 0.0069, 0.0402, 0.1899, 0.7620\}$ | 4.2758 | 4.3219 |
| 1.3 | 9.9873 | $\{0.0000, 0.0000, 0.0000, 0.0001, 0.0007, 0.0041, 0.0190, 0.0722, 0.2340, 0.6701\}$ | 4.5496 | 4.6219 |
| 1.6 | 7.5598 | $\{0.0000, 0.0000, 0.0001, 0.0006, 0.0031, 0.0122, 0.0390, 0.1069, 0.2605, 0.5777\}$ | 4.8150 | 4.9219 |
| 1.9 | 5.6617 | $\{0.0000, 0.0001, 0.0005, 0.0027, 0.0096, 0.0271, 0.0648, 0.1380, 0.2689, 0.4882\}$ | 5.0690 | 5.2219 |
| 2.2 | 2.9228 | $\{0.0000, 0.0005, 0.0027, 0.0090, 0.0228, 0.0485, 0.0919, 0.1600, 0.2608, 0.4037\}$ | 5.3070 | 5.5219 |
| 2.5 | 2.1411 | $\{0.0004, 0.0029, 0.0096, 0.0223, 0.0429, 0.0731, 0.1147, 0.1695, 0.2391, 0.3254\}$ | 5.5208 | 5.8219 |
| 2.8 | 1.9092 | $\{0.0031, 0.0117, 0.0254, 0.0440, 0.0673, 0.0954, 0.1280, 0.1652, 0.2069, 0.2530\}$ | 5.6940 | 6.1219 |
| 3.1036 | 1 | $\{0.0182, 0.0364, 0.0545, 0.0727, 0.0909, 0.1091, 0.1273, 0.1455, 0.1636, 0.1818\}$ | 5.7813 | 6.4255 |
| 3.3219 | 0 | $\{0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1\}$ | 5.5010 | 6.6438 |

Table 5.2: Maximum information leakage for Example 5.7.1 with $n = 10$

**Example 5.6.2.**

```
**if** $X + Y \geq 5$ && $X - Y \geq -1$**then**
    $Z := 0;$
    **else**
    $X := 3;\ Y := 4;$
**fi**;
$Y := X + Y/2 + 1;$
$Z := f(X, Y);$
```

Where $X$ and $Y$ are both high variables with positive integer domains $[1, \ldots, n]$ and $[1, \ldots, 2n]$ respectively (the reasoning is exactly the same if both of them are $k$-bit two's complement variables, which means these variables are $k$ bits binary and they represent two's complement numbers, *e.g.* 8-bit two's complement $11111111 = -1_{10}$, not $255_{10}$), for simplicity and easier to understand, we use positive integer domains in the following analysis, $Z$ is a low program variable and $f(X, Y)$ is some linear function of $X$ and $Y$, *i.e.* addition, subtraction and *etc.*

## CHM's Analysis

Their analysis is pretty much the same as the reasoning for Example 5.6.1 and would end up with a maximum amount of $a + \log 2n$ of information leaking into $Z$ given that $\mathcal{H}(X) = a$.

**Our Analysis**

First the abstract interpretation is expected to derive the following linear constraints at each program point:

Suppose there isn't any other constraint of $X$ and $Y$, then after the **if** statement it is $\{X + Y \geq 5,\ X - Y \geq -1,\ Y \geq 0\}$ (the merge of $\{X + Y \geq 5,\ X - Y \geq -1\}$ and $X := 3;\ Y := 4$ is $\{X + Y \geq 5,\ X - Y \geq -1\}$), after the assignment it is $\{2Y - 2X \geq 2,\ 2Y - X \geq 7,\ -2Y + 3X \geq -3,\ Z = f(X, Y)\}$.

Please refer to Figure 5.5 for the polyhedron corresponding to the constraints before and after assignment, refer to section 3.2.4 for detail.

As above, if we divide the set of $(X, Y)$ according to the values that $X$ can take, we can have a partition of sizes $\{2, 2, 3, 3, \ldots, \frac{n}{2}, \frac{n}{2}\}$ when $n$ is even; $\{2, 2, 3, 3, \ldots, \frac{n-1}{2}, \frac{n-1}{2}, \frac{n+1}{2}\}$ when $n$ is odd.

Recall Theorem 5.5.2, let $k$ be the size of the largest part of the partition and $m$ be the number of parts having size $k$, then if $a \leq \log m$ then $\mathcal{H}_{\max}(P_a) = a + \log k$, otherwise we can use our method of searching for suitable $\alpha$ to obtain the maximal leakage.

Hence there are two cases to consider when $n$ is even:

- if $\mathcal{H}(X) = a < \log 2 = 1$, then we can have that $\mathcal{H}_{\max}(Z) = a + \log 2 = a + 1$;

- otherwise, the maximum leakage into $Z$ is $\mathcal{H}_{\max}(X, Y) = a + \sum_i^n q_i \log |Q_i|$ where $q_i$ is the probability $P(X = i)$ which can be derived based on the sizes of parts of the partition as $q_i = |Q_i|^\alpha / Z(\alpha)$ and $Z(\alpha) = 2(2^\alpha + 3^\alpha + \ldots + (\frac{n}{2})^\alpha)$. As the maximum $|Q_i \models \frac{n}{2}$ in this case, hence $\mathcal{H}_{\max}(X, Y) = a + \sum_i^n q_i \log |Q_i| < a + \log \frac{n}{2}$ hence strictly $< a + \log n$.

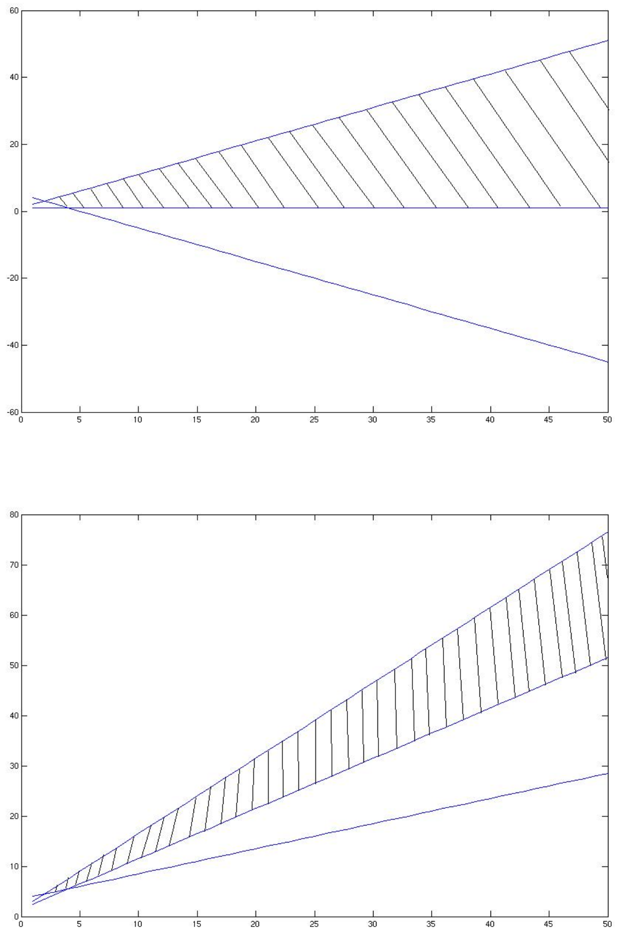

Figure 5.5: Polyhedron for Example 5.7.2 with horizontal axis being $X$ and vertical axis being $Y$

When $n$ is odd, it is the same as the second situation above.

## Concrete Result

Here are some concrete numerical results for different marginal entropy $\mathcal{H}(Y)$ for given $n$,

Table 5.3 is when $n = 5$, note in this case the sizes of parts are $\{2, 2, 3\}$;

the other table is for $n = 10$, in this case, there are two parts having the same largest size, hence different ranges of marginal entropy of $X$ will have different type of maximum leakage as shown in Table 5.4:

In more detail, in Table 5.3, the column of $\mathcal{H}(X)$ is the value of the marginal entropy constraint, we choose to start from 0.2 and increase to 1.4 on an interval of 0.2 (please note that there is nothing special to choose these values, other values can also be chosen as well). The last two values (1.5567 and 1.5850) are chosen on purpose, as one (1.5567) gives the largest value of $\mathcal{H}_{\max}(Z)$ and the other (1.5850) corresponding to the maximum marginal entropy $\mathcal{H}(X)$, however, as shown in the table, it won't give us the maximum information leakage of $Z$.

The column $\alpha$ is the value of $\alpha$ that is computed using our algorithm, under the corresponding marginal entropy constraint value of $\mathcal{H}(X)$ and the linear constraint in this example, for example, if $\mathcal{H}(X) = 0.2$, then according to our analysis, the probability distribution which satisfies this marginal entropy value and the constraint is: $q_i = \frac{|Q_i|^\alpha}{Z(\alpha)}$, as here $n = 5$ which means we have a partition of $\{2, 2, 3\}$, the problem we need to solve is $0.2 = -2\frac{2^\alpha}{Z(\alpha)} \log \frac{2^\alpha}{Z(\alpha)} - \frac{3^\alpha}{Z(\alpha)} \log \frac{3^\alpha}{Z(\alpha)}$ where $Z(\alpha) = 22^\alpha + 3^\alpha$, we use binary search to solve $\alpha$, starting from range [0, 20] (with precision $10^{-4}$), we stop by when $|\mathcal{H}(Y) - 0.2| \leq 10^{-3}$.

Once the $\alpha$ value is known, the column of $P(Y)$ (the probability of $X$ which gives the maximum entropy) is just the calculation of $\frac{i^\alpha}{Z(\alpha)}$.

The column $\mathcal{H}_{\max}(Z)$ is the value of $\mathcal{H}(X) + \sum_i q_i \log |Q_i|$ where $q_i$ are the corresponding value under $P(X)$.

The column $\mathcal{H}(X) + \log n = \mathcal{H}(X) + \log 5$ thus it varies with $\mathcal{H}(X)$.

| $\mathcal{H}(X)$ | $\alpha$ | $P(X)$ | $\mathcal{H}_{\max}(Z)$ | $\mathcal{H}(X)+\log n$ |
|---|---|---|---|---|
| 0.2 | 10.6429 | $\{0.0000, 0.0000, 0.0130, 0.0130, 0.9740\}$ | 1.7697 | 2.5219 |
| 0.4 | 8.3858 | $\{0.0000, 0.0000, 0.0313, 0.0313, 0.9374\}$ | 1.9484 | 2.7219 |
| 0.6 | 6.9279 | $\{0.0000, 0.0000, 0.0538, 0.0538, 0.8924\}$ | 2.1220 | 2.9219 |
| 0.8 | 5.7680 | $\{0.0000, 0.0000, 0.0809, 0.0809, 0.8382\}$ | 2.2904 | 3.1219 |
| 1 | 4.7298 | $\{0.0000, 0.0000, 0.1136, 0.1136, 0.7728\}$ | 2.4521 | 3.3219 |
| 1.2 | 3.6993 | $\{0.0000, 0.0000, 0.1543, 0.1543, 0.6914\}$ | 2.6045 | 3.5219 |
| 1.4 | 2.5150 | $\{0.0000, 0.0000, 0.2095, 0.2095, 0.5810\}$ | 2.7398 | 3.7219 |
| 1.5567 | 1 | $\{0.0000, 0.0000, 0.2858, 0.2858, 0.4284\}$ | 2.8073 | 3.8786 |
| 1.5850 | 0 | $\{0.0000, 0.0000, 0.3333, 0.3333, 0.3334\}$ | 2.7800 | 3.9069 |

Table 5.3: Maximum information leakage for Example 5.7.2 with $n = 5$

When $\mathcal{H}(X) \leq 1$ as we discussed earlier, $\mathcal{H}_{\max}(Z) = \mathcal{H}(X) + \log 2$, there is no need to calculate either $\alpha$ or $P(X)$, so we use $-$ to skip the calculation.

Note that in this example, even if $n = 10$, $X$ can only take 8 values, and this can be captured by the abstract interpretation, however, traditional quantitative information flow analysis (*e.g.* CHM's framework) has no way to know it hence cannot utilize it.

### 5.6.3 Program with While loop

Now let's look at an example of **while** loop and how our analysis deals with it:

| $\mathcal{H}(X)$ | $\alpha$ | $P(X)$ | $\mathcal{H}_{\max}(Z)$ | $\mathcal{H}(X) + \log n$ |
|---|---|---|---|---|
| 0.1 | - | - | 1.1 | 2.4219 |
| 0.4 | - | - | 1.4 | 2.7219 |
| 0.7 | - | - | 1.7 | 3.0219 |
| 1 | - | - | 2 | 3.3219 |
| 1.3 | 13.1693 | $\{0.0000, 0.0000, 0.0000, 0.00000.0006,$ $0.0006, 0.0251, 0.0251, 0.4743, 0.4743\}$ | 3.6049 | 4.6219 |
| 1.6 | 8.9540 | $\{0.0000, 0.0000, 0.0001, 0.0001, 0.0045,$ $0.0045, 0.0592, 0.0592, 0.4362, 0.4362\}$ | 3.8769 | 4.9219 |
| 1.9 | 6.4094 | $\{0.0000, 0.0000, 0.0011, 0.0011, 0.0148,$ $0.0148, 0.0935, 0.0935, 0.3906, 0.3906\}$ | 4.1370 | 5.2219 |
| 2.2 | 4.5834 | $\{0.0000, 0.0000, 0.0051, 0.0051, 0.0327,$ $0.0327, 0.1222, 0.1222, 0.3400, 0.3400\}$ | 4.3815 | 5.5219 |
| 2.5 | 3.1167 | $\{0.0000, 0.0000, 0.0163, 0.0163, 0.0578,$ $0.0578, 0.1417, 0.1417, 0.2841, 0.2841\}$ | 4.6023 | 5.8219 |
| 2.8 | 1.7208 | $\{0.0000, 0.0000, 0.0449, 0.0449, 0.0901,$ $0.0901, 0.1479, 0.1479, 0.2171, 0.2171\}$ | 4.7752 | 6.1219 |
| 2.9242 | 1 | $\{0.0000, 0.0000, 0.0714, 0.0714, 0.1071,$ $0.10710.1429, 0.1429, 0.1786, 0.1786\}$ | 4.8074 | 6.4255 |
| 3 | 0 | $\{0.0000, 0.0000, 0.1250, 0.1250, 0.1250,$ $0.1250, 0.1250, 0.1250, 0.1250, 0.1250\}$ | 4.7267 | 6.6438 |

Table 5.4: Maximum information leakage for Example 5.7.2 with $n = 10$

**Example 5.6.3.**

```
$Z := 0;$
while $X \leq Y$
    $Z := 0;$
    $Z := X + Y;$
    $X := X + 1;$
end while
```

with $X$ and $Y$ each can take integer values in the range of $[1, \ldots, n]$, and the marginal constraint $\mathcal{H}(Y) = a$. $Z$ is initialized to 0 and can be only observed as the output, it cannot be observed during the program.

This example is partially equivalent to the simple program of $X := Y + 1;\ Z =: X + Y - 1$ if the body of the **while** loop is executed at least once and the program terminates.

### CHM's framework

CHM's analysis towards **while** loop first conducts a dependence analysis to establish all the sources of information that flows into the loop. In this example, it is quite clear that $Z$ depends on $X$ and $Y$, $X$ depends on $Y$, namely $[Z \mapsto \{X, Y\}],\ [X \mapsto \{Y\}]]$. Thus, there are two information sources $X$ and $Y$, since the only place where information will be leaked is at $Z := X + Y$. According to the general data processing rule, it is easy to derive that the maximum amount of information that can be leaked is $a + \log n$.

### Our analysis

There are two cases to consider:

- $X \geq Y + 1$ before the loop:

  In this case, there is no information leakage as the **while** loop will not be executed.

- $X \leq Y$ before the loop:

  In this case, the abstract interpretation is expected to give $X \leq Y$, using our analysis, we can have $\mathcal{H}_{\max}(Z) = a + q_i \log i$, where $P(Y = i) = q_i = \frac{i^\alpha}{Z(\alpha)}$ the same as Example 5.6.1. This result is the best we can get based on current abstract interpretation analysis.

Thus in both situations, the maximum leakage does not exceed $\mathcal{H}_{\max}(Z)$.

However, the actual upper bound of information into $Z$ should be $a$ as when **while** loops terminates, $X$ is always equal to $Y$, hence actually $Z := 2Y$. If abstract interpretation can provide more precise linear constraint such as it can capture that fact that when the above loop terminates, $X = Y + 1$ is always held, then our analysis will, in turn, give more precise result as $\mathcal{H}_{\max}(Z) = a$.

### Concrete Result

The concrete results are shown in Table 5.5 for $n = 5$ and as Table 5.6 for $n = 10$, note that the most precise upper bound should be $\mathcal{H}_{\max}(Z) = \mathcal{H}(Y) = a$.

| $\mathcal{H}(Y)$ | $\alpha$ | $P(Y)$ | $\mathcal{H}_{\max}(Z)$ | $\mathcal{H}(Y) + \log n$ |
|---|---|---|---|---|
| 0.2 | 15.5386 | $\{0.0000, 0.0000, 0.0003, 0.0302, 0.9694\}$ | 2.5119 | 2.5219 |
| 0.4 | 11.4449 | $\{0.0000, 0.0000, 0.0027, 0.0720, 0.9253\}$ | 2.6968 | 2.7219 |
| 0.6 | 8.9541 | $\{0.0000, 0.0002, 0.0090, 0.1183, 0.8725\}$ | 2.8769 | 2.9219 |
| 0.8 | 7.1533 | $\{0.0000, 0.0012, 0.0210, 0.1648, 0.8130\}$ | 3.0518 | 3.1219 |
| 1 | 5.7495 | $\{0.0001, 0.0039, 0.0397, 0.2076, 0.7488\}$ | 3.2206 | 3.3219 |
| 1.2 | 4.6085 | $\{0.0004, 0.0100, 0.0647, 0.2436, 0.6813\}$ | 3.3817 | 3.5219 |
| 1.4 | 3.6568 | $\{0.0017, 0.0215, 0.0945, 0.2705, 0.6118\}$ | 3.5329 | 3.7219 |
| 1.6 | 2.8472 | $\{0.0055, 0.0399, 0.1264, 0.2868, 0.5414\}$ | 3.6709 | 3.9219 |
| 1.8 | 2.1411 | $\{0.0150, 0.0661, 0.1574, 0.2915, 0.4700\}$ | 3.7899 | 4.1219 |
| 2 | 1.4931 | $\{0.0358, 0.1006, 0.1845, 0.2835, 0.3956\}$ | 3.8785 | 4.3219 |
| 2.1493 | 1 | $\{0.0667, 0.1334, 0.2000, 0.2666, 0.3333\}$ | 3.9068 | 4.4712 |
| 2.3219 | 0 | $\{0.2, 0.2, 0.2, 0.2, 0.2\}$ | 3.7033 | 4.6439 |

Table 5.5: Maximum information leakage for Example 5.7.3 with $n = 5$

| $\mathcal{H}(Y)$ | $\alpha$ | $P(Y)$ | $\mathcal{H}_{\max}(Z)$ | $\mathcal{H}(Y) + \log n$ |
|---|---|---|---|---|
| 0.1 | 41.2661 | $\{0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0001, 0.0128, 0.9871\}$ | 3.4200 | 3.4219 |
| 0.4 | 24.6453 | $\{0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0001, 0.0038, 0.0691, 0.9270\}$ | 3.7101 | 3.7219 |
| 0.7 | 17.6469 | $\{0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0001, 0.0016, 0.0166, 0.1323, 0.8494\}$ | 3.9956 | 4.0219 |
| 1 | 13.1881 | $\{0.0000, 0.0000, 0.0000, 0.0000, 0.0001, 0.0009, 0.0069, 0.0402, 0.1899, 0.7620\}$ | 4.2758 | 4.3219 |
| 1.3 | 9.9873 | $\{0.0000, 0.0000, 0.0000, 0.0001, 0.0007, 0.0041, 0.0190, 0.0722, 0.2340, 0.6701\}$ | 4.5496 | 4.6219 |
| 1.6 | 7.5598 | $\{0.0000, 0.0000, 0.0001, 0.0006, 0.0031, 0.0122, 0.0390, 0.1069, 0.2605, 0.5777\}$ | 4.8150 | 4.9219 |
| 1.9 | 5.6617 | $\{0.0000, 0.0001, 0.0005, 0.0027, 0.0096, 0.0271, 0.0648, 0.1380, 0.2689, 0.4882\}$ | 5.0690 | 5.2219 |
| 2.2 | 2.9228 | $\{0.0000, 0.0005, 0.0027, 0.0090, 0.0228, 0.0485, 0.0919, 0.1600, 0.2608, 0.4037\}$ | 5.3070 | 5.5219 |
| 2.5 | 2.1411 | $\{0.0004, 0.0029, 0.0096, 0.0223, 0.0429, 0.0731, 0.1147, 0.1695, 0.2391, 0.3254\}$ | 5.5208 | 5.8219 |
| 2.8 | 1.9092 | $\{0.0031, 0.0117, 0.0254, 0.0440, 0.0673, 0.0954, 0.1280, 0.1652, 0.2069, 0.2530\}$ | 5.6940 | 6.1219 |
| 3.1036 | 1 | $\{0.0182, 0.0364, 0.0545, 0.0727, 0.0909, 0.1091, 0.1273, 0.1455, 0.1636, 0.1818\}$ | 5.7813 | 6.4255 |
| 3.3219 | 0 | $\{0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1\}$ | 5.5010 | 6.6438 |

Table 5.6: Maximum information leakage for Example 5.7.3 with $n = 10$

# Chapter 6

# Two Constraints Joint Entropy Maximization and Generalization

In the last chapter, the single constraint joint entropy maximization problem is discussed in details, however the more common situation is with multiple constraints, *e.g.* $\mathcal{H}(X) = a$ and $\mathcal{H}(Y) = b$. Although it looks like that there is just only one more marginal constraint compared with the single constraint problem, the complexity of the two constraints problem is more than double of that of the single one. This is mainly because the two constraint problem is three dimensional which is difficult to analyze in the first place, and the second is the complexity of the entropy function itself. This chapter shows some investigation into the problem.

## 6.1 Simple Problem

Let's still consider the example of the simple program 5.6.1 from the last chapter, from which the relationship between program variables $X$ and $Y$ is the same as $X \leq Y, X \geq 0$. To make the induction as simple as possible in this problem, let's assume that $X$ and $Y$ are both binary variables. Together with two marginal entropy constraints $\mathcal{H}(X) = a$ and $\mathcal{H}(Y) = b$, the question is the same: what is the maximal value for the joint entropy $\mathcal{H}(X, Y)$ of $X$ and $Y$?

Let $p_{i,j} = P(X = i, Y = j)$ and let $r_i = P(X = i)$, $q_j = P(Y = j)$. Note that, since $X \leq Y$, we may restrict attention to $p_{i,j}$ such that $i \leq j$.

Mathematically, the objective function we want to maximize is:

$$\mathcal{H}(X, Y)_{X \leq Y,\ X,Y \in [0,\ 1]} = -\sum_{i,\ j \in [0,\ 1],\ i \leq j} p_{i,j} \log p_{i,j} \tag{6.1}$$

Subject to the following constraints:

$$\mathcal{H}(\vec{r}) - a = 0 \tag{6.2}$$

$$\mathcal{H}(\vec{q}) - b = 0 \tag{6.3}$$

$$\sum_{i \leq j,\ i,j \in [0,1]} p_{i,j} - 1 = 0 \tag{6.4}$$

Now define the Lagrangian $\Lambda$ as

$$\Lambda = -p_{i,j}\log p_{i,j} - \lambda_1(\mathcal{H}(X) - a) - \lambda_2(\mathcal{H}(Y) - b) - \lambda_3(\sum_{i,j} p_{i,j} - 1) \quad (6.5)$$

which gives rise to the following family of partial derivative equations, $i \leq j,\ i, j \in [0,1]$:

$$\frac{\partial \Lambda}{\partial p_{0,0}} = -(\log p_{0,0} + \frac{1}{\ln 2}) + \lambda_1(\log r_0 + \frac{1}{\ln 2}) + \lambda_2(\log q_0 + \frac{1}{\ln 2}) - \lambda_3 = 0 \quad (6.6)$$

$$\frac{\partial \Lambda}{\partial p_{0,1}} = -(\log p_{0,1} + \frac{1}{\ln 2}) + \lambda_1(\log r_0 + \frac{1}{\ln 2}) + \lambda_2(\log q_1 + \frac{1}{\ln 2}) - \lambda_3 = 0 \quad (6.7)$$

$$\frac{\partial \Lambda}{\partial p_{1,1}} = -(\log p_{1,1} + \frac{1}{\ln 2}) + \lambda_1(\log r_1 + \frac{1}{\ln 2}) + \lambda_2(\log q_1 + \frac{1}{\ln 2}) - \lambda_3 = 0 \quad (6.8)$$

By subtracting the above equations from each other, we can get the following:

$$\log p_{0,1} - \log p_{0,0} = \lambda_2(\log q_1 - \log q_0) \quad (6.9)$$

$$\log p_{1,1} - \log p_{0,1} = \lambda_1(\log r_1 - \log r_0) \quad (6.10)$$

hence

$$\log \frac{p_{0,1}}{p_{0,0}} = \lambda_2(\log \frac{q_1}{q_0}) \quad (6.11)$$

$$\log \frac{p_{1,1}}{p_{0,1}} = \lambda_1(\log \frac{r_1}{r_0}) \quad (6.12)$$

Let $\alpha = \lambda_1$ and $\beta = \lambda_2$. Then we have:

$$p_{0,0} = \left(\frac{q_0}{q_1}\right)^{\beta} p_{0,1} \tag{6.13}$$

$$p_{1,1} = \left(\frac{r_1}{r_0}\right)^{\alpha} p_{0,1} \tag{6.14}$$

Substitute them into the constraint of $\sum p_{i,j} = 1$:

$$\left(\left(\frac{q_0}{q_1}\right)^{\beta} + \left(\frac{r_1}{r_0}\right)^{\alpha} + 1\right) p_{0,1} = 1 \tag{6.15}$$

$$\frac{q_0^{\beta} r_0^{\alpha} + r_1^{\alpha} q_1^{\beta} + r_0^{\alpha} q_1^{\beta}}{q_1^{\beta} r_0^{\alpha}} p_{0,1} = 1 \tag{6.16}$$

thus

$$p_{0,0} = \frac{r_0^{\alpha} q_0^{\beta}}{Z(\alpha, \beta)} \tag{6.17}$$

$$p_{0,1} = \frac{r_0^{\alpha} q_1^{\beta}}{Z(\alpha, \beta)} \tag{6.18}$$

$$p_{1,1} = \frac{r_1^{\alpha} q_1^{\beta}}{Z(\alpha, \beta)} \tag{6.19}$$

Where $Z(\alpha, \beta) = \sum_{i,j} r_i^{\alpha} q_j^{\beta} = r_0^{\alpha} q_0^{\beta} + r_0^{\alpha} q_1^{\beta} + r_1^{\alpha} q_1^{\beta}$.

### 6.1.1 Properties of $\mathcal{H}(X)$ and $\mathcal{H}(Y)$

We now show that $\mathcal{H}(X)$ is a concave function in terms of $\alpha$, and the same of that for $\mathcal{H}(Y)$ in terms of $\beta$.

**Proof for $\mathcal{H}(X)$**

We first show the concaveness of $\mathcal{H}(X)$ in terms of $\alpha$. The proof is conducted to show that there is a zero point in the first derivative of $\mathcal{H}(X)$, beyond this point, the first derivative is either larger or less than zero.

*Proof.* As $p_{i,j} = \frac{r_i^\alpha q_j^\beta}{Z(\alpha,\beta)}$ and $X \leq Y$,

$$r_0 = \frac{r_0^\alpha (q_0^\beta + q_1^\beta)}{Z(\alpha,\beta)},\ r_1 = \frac{r_1^\alpha q_1^\beta}{Z(\alpha,\beta)}$$

Thus,

$$\begin{aligned}
\mathcal{H}(X) &= -\tfrac{r_0^\alpha (q_0^\beta + q_1^\beta)}{Z(\alpha,\beta)} \log \tfrac{r_0^\alpha (q_0^\beta + q_1^\beta)}{Z(\alpha,\beta)} - \tfrac{r_1^\alpha q_1^\beta}{Z(\alpha,\beta)} \log \tfrac{r_1^\alpha q_1^\beta}{Z(\alpha,\beta)} \\
&= -\tfrac{r_0^\alpha (q_0^\beta + q_1^\beta)}{Z(\alpha,\beta)} \Big(\log r_0^\alpha (q_0^\beta + q_1^\beta) - \log Z(\alpha,\beta)\Big) - \tfrac{r_1^\alpha q_1^\beta}{Z(\alpha,\beta)} \Big(\log r_1^\alpha q_1^\beta - \log Z(\alpha,\beta)\Big) \\
&= \log Z(\alpha,\beta) - \tfrac{r_0^\alpha (q_0^\beta + q_1^\beta)}{Z(\alpha,\beta)} \log r_0^\alpha (q_0^\beta + q_1^\beta) - \tfrac{r_1^\alpha q_1^\beta}{Z(\alpha,\beta)} \log r_1^\alpha q_1^\beta \\
&= \log Z(\alpha,\beta) - \tfrac{r_0^\alpha (q_0^\beta + q_1^\beta)}{Z(\alpha,\beta)} \Big(\alpha \log r_0 + \log(q_0^\beta + q_1^\beta)\Big) - \tfrac{r_1^\alpha q_1^\beta}{Z(\alpha,\beta)} \Big(\alpha \log r_1 + \log q_1^\beta\Big) \\
&= \log Z(\alpha,\beta) - \tfrac{\alpha r_0^\alpha (q_0^\beta + q_1^\beta)}{Z(\alpha,\beta)} \log r_0 - \tfrac{r_0^\alpha (q_0^\beta + q_1^\beta)}{Z(\alpha,\beta)} \log(q_0^\beta + q_1^\beta) \\
&\quad -\alpha \tfrac{r_1^\alpha q_1^\beta}{Z(\alpha,\beta)} \log r_1 - \tfrac{r_1^\alpha q_1^\beta}{Z(\alpha,\beta)} \log q_1^\beta
\end{aligned}$$

partial derivative of $\alpha$ is:

$$\begin{aligned}
\frac{\partial\mathcal{H}(X)}{\partial\alpha} &= \frac{Z'(\alpha,\beta)_\alpha}{Z(\alpha,\beta)\ln 2} - \frac{((r_0^\alpha+\alpha r_0^\alpha \ln r_0)Z(\alpha,\beta)-\alpha r_0^\alpha Z'(\alpha,\beta)_\alpha)(q_0^\beta+q_1^\beta)\ln r_0}{(Z(\alpha,\beta))^2\ln 2} \\
&\quad - \frac{(r_0^\alpha \ln r_0 Z(\alpha,\beta)-r_0^\alpha Z'(\alpha,\beta)_\alpha)(q_0^\beta+q_1^\beta)\ln(q_0^\beta+q_1^\beta)}{(Z(\alpha,\beta))^2\ln 2} \\
&\quad - \frac{((r_1^\alpha+\alpha r_1^\alpha \ln r_1)Z(\alpha,\beta)-\alpha r_1^\alpha Z(\alpha,\beta)'_\alpha)q_1^\beta \ln r_1}{(Z(\alpha,\beta))^2\ln 2} \\
&\quad - \frac{(r_1^\alpha \ln r_1 Z(\alpha,\beta)-r_1^\alpha Z'(\alpha,\beta)_\alpha)q_1^\beta \ln q_1^\beta}{(Z(\alpha,\beta))^2\ln 2} \\
&= \frac{Z'(\alpha,\beta)_\alpha}{Z(\alpha,\beta)\ln 2} - \frac{(r_0^\alpha+\alpha r_0^\alpha \ln r_0)Z(\alpha,\beta)(q_0^\beta+q_1^\beta)\ln r_0}{(Z(\alpha,\beta))^2\ln 2} - \frac{r_0^\alpha \ln r_0 Z(\alpha,\beta)(q_0^\beta+q_1^\beta)\ln(q_0^\beta+q_1^\beta)}{(Z(\alpha,\beta))^2\ln 2} \\
&\quad - \frac{(r_1^\alpha+\alpha r_1^\alpha \ln r_1)Z(\alpha,\beta)q_1^\beta \ln r_1}{(Z(\alpha,\beta))^2\ln 2} - \frac{r_1^\alpha \ln r_1 Z(\alpha,\beta)q_1^\beta \ln q_1^\beta}{(Z(\alpha,\beta))^2\ln 2} \\
&\quad + \frac{\alpha r_0^\alpha (q_0^\beta+q_1^\beta)\ln r_0 Z'(\alpha,\beta)_\alpha}{(Z(\alpha,\beta))^2\ln 2} + \frac{r_0^\alpha (q_0^\beta+q_1^\beta)\ln(q_0^\beta+q_1^\beta)Z'(\alpha,\beta)_\alpha}{(Z(\alpha,\beta))^2\ln 2} \\
&\quad + \frac{\alpha r_1^\alpha q_1^\beta \ln r_1 Z'(\alpha,\beta)_\alpha}{(Z(\alpha,\beta))^2\ln 2} + \frac{r_1^\alpha q_1^\beta \ln q_1^\beta Z'(\alpha,\beta)_\alpha}{(Z(\alpha,\beta))^2\ln 2} \\
&= -\frac{\Big(r_0^\alpha (q_0^\beta+q_1^\beta)(\ln r_0^\alpha (q_0^\beta+q_1^\beta))\ln r_0 + r_1^\alpha q_1^\beta (\ln r_1^\alpha q_1^\beta)\ln r_1\Big)(r_0^\alpha (q_0^\beta+q_1^\beta)+r_1^\alpha q_1^\beta)}{(Z(\alpha,\beta))^2\ln 2} \\
&\quad + \frac{\Big(r_0^\alpha (q_0^\beta+q_1^\beta)(\ln r_0^\alpha (q_0^\beta+q_1^\beta))+r_1^\alpha q_1^\beta (\ln r_1^\alpha q_1^\beta)\Big)((r_0^\alpha (q_0^\beta+q_1^\beta))\ln r_0 + r_1^\alpha q_1^\beta \ln r_1)}{(Z(\alpha,\beta))^2\ln 2} \\
&= -\frac{r_0^\alpha (q_0^\beta+q_1^\beta) r_1^\beta q_1^\beta \Big(\ln r_1^\alpha q_1^\beta \ln r_1 + \ln r_0^\alpha (q_0^\beta+q_1^\beta)\Big)}{(Z(\alpha,\beta))^2\ln 2} \\
&+ \frac{r_0^\alpha (q_0^\beta+q_1^\beta) r_1^\beta q_1^\beta \Big(\ln r_1^\alpha q_1^\beta \ln r_0 + \ln r_0^\alpha (q_0^\beta+q_1^\beta)\ln r_1\Big)}{(Z(\alpha,\beta))^2\ln 2} \\
&= \frac{r_0^\alpha (q_0^\beta+q_1^\beta) r_1^\beta q_1^\beta \ln \frac{r_1^\alpha q_1^\beta}{r_0^\alpha (q_0^\beta+q_1^\beta)} \ln \frac{r_0}{r_1}}{(Z(\alpha,\beta))^2\ln 2}
\end{aligned}$$

as $r_0^\alpha (q_0^\beta + q_1^\beta) r_1^\beta q_1^\beta > 0$, when the partial derivative equals zero, it is

$$\ln \frac{r_1^\alpha q_1^\beta}{r_0^\alpha (q_0^\beta + q_1^\beta)} \ln \frac{r_0}{r_1} = 0$$

$$\left(\alpha \ln \frac{r_1}{r_0} + \ln \frac{q_1^\beta}{q_0^\beta + q_1^\beta}\right) \ln \frac{r_0}{r_1} = 0$$

If $r_0 \neq r_1$, hence , $\ln \frac{r_0}{r_1} \neq 0$:

$$\alpha = \frac{\ln \frac{q_0^\beta + q_1^\beta}{q_1^\beta}}{\ln \frac{r_1}{r_0}}$$

Please note that $r_0 > r_1$ and this can be argued simply as $r_0 = p(0,0) + p(0,1),\ r_1 = p(1,1)$. In order to let joint entropy gets as large as possible, $p(0,0),\ p(0,1),\ p(1,1)$ should go towards uniform as much as possible, which will definitely make $r_0 > r_1$.

Thus when $\alpha > \frac{\ln \frac{q_0^\beta + q_1^\beta}{q_1^\beta}}{\ln \frac{r_1}{r_0}}$, the above partial derivative is always $< 0$ while when $\alpha < \frac{\ln \frac{q_0^\beta + q_1^\beta}{q_1^\beta}}{\ln \frac{r_1}{r_0}}$, the above partial derivative is always $> 0$.

If $r_0 = r_1$, then in this case where $X$ and $Y$ can only take $[0,\ 1]$, we can know that $r_0 = r_1 = 0.5$, then the problem can be transferred back to the single constraint problem with marginal constraint of $\mathcal{H}(Y)$. $\square$

**Proof for $\mathcal{H}(Y)$**

We can conduct the partial derivative of $\beta$ based on the same reasoning:

*Proof.* As $p_{i,j} = \frac{r_i^\alpha q_j^\beta}{Z(\alpha,\beta)}$ and $X \leq Y$,

$$q_0 = \frac{r_0^\alpha q_0^\beta}{Z(\alpha,\beta)},\ q_1 = \frac{(r_0^\alpha + r_1^\alpha) q_1^\beta}{Z(\alpha,\beta)}$$

Thus,

$$
\begin{aligned}
\mathcal{H}(Y) &= \tfrac{r_0^\alpha q_0^\beta}{Z(\alpha,\beta)} \log \tfrac{r_0^\alpha q_0^\beta}{Z(\alpha,\beta)} - \tfrac{(r_0^\alpha + r_1^\alpha) q_1^\beta}{Z(\alpha,\beta)} \log \tfrac{(r_0^\alpha + r_1^\alpha) q_1^\beta}{Z(\alpha,\beta)} \\
&= -\tfrac{r_0^\alpha q_0^\beta}{Z(\alpha,\beta)} \Big(\log r_0^\alpha q_0^\beta - \log Z(\alpha,\beta)\Big) - \tfrac{(r_0^\alpha + r_1^\alpha) q_1^\beta}{Z(\alpha,\beta)} \Big(\log (r_0^\alpha + r_1^\alpha) q_1^\beta - \log Z(\alpha,\beta)\Big) \\
&= \log Z(\alpha,\beta) - \tfrac{r_0^\alpha q_0^\beta}{Z(\alpha,\beta)} \log r_0^\alpha q_0^\beta) - \tfrac{(r_0^\alpha + r_1^\alpha) q_1^\beta}{Z(\alpha,\beta)} \log (r_0^\alpha + r_1^\alpha) q_1^\beta \\
&= \log Z(\alpha,\beta) - \tfrac{r_0^\alpha q_0^\beta}{Z(\alpha,\beta)} (\log r_0^\alpha + \beta \log q_0) - \tfrac{(r_0^\alpha + r_1^\alpha) q_1^\beta}{Z(\alpha,\beta)} (\log (r_0^\alpha + r_1^\alpha) + \beta \log q_1) \\
&= \log Z(\alpha,\beta) - \tfrac{r_0^\alpha q_0^\beta}{Z(\alpha,\beta)} \log r_0^\alpha - \tfrac{\beta r_0^\alpha q_0^\beta}{Z(\alpha,\beta)} \log q_0 \\
&\quad - \tfrac{(r_0^\alpha + r_1^\alpha) q_1^\beta}{Z(\alpha,\beta)} \log (r_0^\alpha + r_1^\alpha) - \tfrac{\beta (r_0^\alpha + r_1^\alpha) q_1^\beta}{Z(\alpha,\beta)} \log q_1
\end{aligned}
$$

partial derivative of $\beta$ is:

$$\begin{aligned}
\frac{\partial \mathcal{H}(Y)}{\partial \beta} &= \frac{Z'(\alpha,\beta)_\beta}{Z(\alpha,\beta)\ln 2} - \frac{\Big((q_0^\beta+\beta q_0^\beta \ln q_0)Z(\alpha,\beta)-\beta q_0^\beta Z'(\alpha,\beta)_\alpha\Big) r_0^\alpha \ln q_0}{(Z(\alpha,\beta))^2 \ln 2} \\
&\quad - \frac{\Big(q_1^\beta \ln q_1 Z(\alpha,\beta)-q_1^\beta Z'(\alpha,\beta)_\beta\Big)(r_0^\alpha+r_1^\alpha)\ln(r_0^\alpha+r_1^\alpha)}{(Z(\alpha,\beta))^2 \ln 2} \\
&\quad - \frac{\Big((q_1^\beta+\beta q_1^\beta \ln q_1)Z(\alpha,\beta)-\beta q_1^\beta Z'(\alpha,\beta)_\beta\Big)(r_0^\alpha+r_1^\alpha)\ln q_1}{(Z(\alpha,\beta))^2 \ln 2} \\
&\quad - \frac{\Big(q_0^\beta \ln q_0 Z(\alpha,\beta)-q_0^\beta Z'(\alpha,\beta)_\beta\Big) r_0^\alpha \ln r_0^\alpha}{(Z(\alpha,\beta))^2 \ln 2} \\
&= \frac{Z'(\alpha,\beta)_\beta}{Z(\alpha,\beta)\ln 2} - \frac{(q_0^\beta+\beta q_0^\beta \ln q_0)Z(\alpha,\beta) r_0^\alpha \ln q_0}{(Z(\alpha,\beta))^2 \ln 2} - \frac{q_1^\beta \ln q_1 Z(\alpha,\beta)(r_0^\alpha+r_1^\alpha)\ln(r_0^\alpha+r_1^\alpha)}{(Z(\alpha,\beta))^2 \ln 2} \\
&\quad - \frac{(q_1^\beta+\beta q_1^\beta \ln q_1)Z(\alpha,\beta)(r_0^\alpha+r_1^\alpha)\ln q_1}{(Z(\alpha,\beta))^2 \ln 2} - \frac{q_0^\beta \ln q_0 Z(\alpha,\beta) r_0^\alpha \ln r_0^\alpha}{(Z(\alpha,\beta))^2 \ln 2} \\
&\quad + \frac{\beta q_0^\beta r_0^\alpha \ln q_0^\beta Z'(\alpha,\beta)_\beta}{(Z(\alpha,\beta))^2 \ln 2} + \frac{q_1^\beta (r_0^\alpha+r_1^\alpha)\ln(r_0^\alpha+r_1^\alpha) Z'(\alpha,\beta)_\beta}{(Z(\alpha,\beta))^2 \ln 2} \\
&\quad + \frac{\beta q_1^\beta (r_0^\alpha+r_1^\alpha)\ln q_1 Z'(\alpha,\beta)_\beta}{(Z(\alpha,\beta))^2 \ln 2} + \frac{q_0^\beta r_0^\alpha \ln r_0^\alpha Z'(\alpha,\beta)_\beta}{(Z(\alpha,\beta))^2 \ln 2} \\
&= -\frac{\Big(r_0^\alpha q_0^\beta \ln q_0 \ln r_0^\alpha q_0^\beta+(r_0^\alpha+r_1^\alpha)q_1^\beta \ln(r_0^\alpha+r_1^\alpha)q_1^\beta \ln q_1\Big)(r_0^\alpha q_0^\beta+(r_0^\alpha+r_1^\alpha)q_1^\beta)}{(Z(\alpha,\beta))^2 \ln 2} \\
&\quad + \frac{\Big(r_0^\alpha q_0^\beta \ln r_0^\alpha q_0^\beta+(r_0^\alpha+r_1^\alpha)q_1^\beta \ln(r_0^\alpha+r_1^\alpha)q_1^\beta\Big)((r_0^\alpha q_0^\beta \ln q_0+(r_0^\alpha+r_1^\alpha)q_1^\beta \ln q_1)}{(Z(\alpha,\beta))^2 \ln 2} \\
&= -\frac{r_0^\alpha q_0^\beta (r_0^\alpha+r_1^\alpha)q_1^\beta \Big(\ln r_0^\alpha q_0^\beta \ln q_0+\ln(r_0^\alpha+r_1^\alpha)q_1^\beta \ln q_1\Big)}{(Z(\alpha,\beta))^2 \ln 2} \\
&+ \frac{r_0^\alpha q_0^\beta (r_0^\alpha+r_1^\alpha)q_1^\beta \Big(\ln r_0^\alpha q_0^\beta \ln q_1+\ln(r_0^\alpha+r_1^\alpha)q_1^\beta \ln q_0\Big)}{(Z(\alpha,\beta))^2 \ln 2} \\
&= \frac{r_0^\alpha q_0^\beta (r_0^\alpha+r_1^\alpha)q_1^\beta \ln \frac{r_0^\alpha q_0^\beta}{(r_0^\alpha+r_1^\alpha)q_1^\beta} \ln \frac{q_1}{q_0}}{(Z(\alpha,\beta))^2 \ln 2}
\end{aligned}$$

as $r_0^\alpha q_0^\beta (r_0^\alpha + r_1^\alpha) q_1^\beta > 0$, when the partial derivative equals zero, it is

$$\ln \frac{r_0^\alpha q_0^\beta}{(r_0^\alpha + r_1^\alpha) q_1^\beta} \ln \frac{q_1}{q_0} = 0$$

$$\left(\beta \ln \frac{q_0}{q_1} + \ln \frac{r_0^\alpha}{r_0^\alpha + r_1^\alpha}\right) \ln \frac{q_1}{q_0} = 0$$

If $q_0 \neq q_1$, hence , $\ln \frac{q_1}{q_0} \neq 0$:

$$\beta = \frac{\ln \frac{r_0^\alpha + r_1^\alpha}{r_0^\alpha}}{\ln \frac{q_0}{q_1}}$$

Please note that $r_0 > r_1$ and this can be argued simply as $r_0 = p(0,0) + p(0,1),\ r_1 = p(1,1)$. In order to let joint entropy gets as large as possible, $p(0,0),\ p(0,1),\ p(1,1)$ should go towards uniform as much as possible, which will definitely make $r_0 > r_1$.

Thus when $\beta > \frac{\ln \frac{q_0^\beta + q_1^\beta}{q_1^\beta}}{\ln \frac{r_1}{r_0}}$, the above partial derivative is always $< 0$ while when $\beta < \frac{\ln \frac{q_0^\beta + q_1^\beta}{q_1^\beta}}{\ln \frac{r_1}{r_0}}$, the above partial derivative is always $> 0$.

Same as before, if $q_0 = q_1$, the problem can be simplified as single constraint problem which has been discussed in Chapter 5. □

Then the only task left is to find suitable $\alpha$ and $\beta$ to make the marginal entropy satisfy $\mathcal{H}(X) = a$ and $\mathcal{H}(Y) = b$ simultaneously. For $X$ and $Y$ only taking two values each, the search is more or less direct as for each value of entropies of $X$ and $Y$, there are only two marginal distributions satisfying them, hence, there are only four situations to consider in total which will be discussed later in this chapter. However, once $X$ and $Y$ can take more values each, the situation will become more and more complicated.

### 6.1.2 Three Variable Case of Simple Problem

In last section, there are only two high variables involved, in this section, we show how more than two variables can also be derived using our analysis:

In this section, we use $X_1$, $X_2$ and $X_3$ to represent three high variables such that $X_1 \leq X_2 \leq X_3$ and all of them $\in [0,1]$, $r_i$, $q_i, p_{i,j,k}$ to represent $P(X_1 = i), P(X_2 = j), P(X_1 = i, X_2 = j, X_3 = k)$ respectively, and with constraints $\mathcal{H}(X_1) = a$ and $\mathcal{H}(X_2) = b$ (it can be entropies of any two of the three). Please note that because $X_1 \leq X_2 \leq X_3$, we only interested in $p_{i,j,k}$ such that $i \leq j \leq k$.

The Lagrangian is:

$$\Lambda = -\sum_{i,j,k} p_{i,j,k} \log p_{i,j,k} - \lambda_1(\mathcal{H}(X_1) - a) - \lambda_2(\mathcal{H}(X_2) - b) - \lambda_3(\sum_{i,j,k} p_{i,j,k} - 1) \tag{6.20}$$

which gives rise to the following family of partial derivative equations, $i \leq j \leq k, i,\ j,\ k \in [0,1]$:

$$\begin{aligned}
\frac{\partial \Lambda}{\partial p_{0,0,0}} &= -(\log p_{0,0,0} + \frac{1}{\ln 2}) + \frac{\lambda_1}{\ln 2}(\ln r_0 + 1) + \frac{\lambda_2}{\ln 2}(\ln q_0 + 1) - \lambda_3 = 0 \\
\frac{\partial \Lambda}{\partial p_{0,0,1}} &= -(\log p_{0,0,1} + \frac{1}{\ln 2}) + \frac{\lambda_1}{\ln 2}(\ln r_0 + 1) + \frac{\lambda_2}{\ln 2}(\ln q_0 + 1) - \lambda_3 = 0 \\
\frac{\partial \Lambda}{\partial p_{0,1,1}} &= -(\log p_{0,1,1} + \frac{1}{\ln 2}) + \frac{\lambda_1}{\ln 2}(\ln r_0 + 1) + \frac{\lambda_2}{\ln 2}(\ln q_1 + 1) - \lambda_3 = 0 \\
\frac{\partial \Lambda}{\partial p_{1,1,1}} &= -(\log p_{1,1,1} + \frac{1}{\ln 2}) + \frac{\lambda_1}{\ln 2}(\ln r_1 + 1) + \frac{\lambda_2}{\ln 2}(\ln q_1 + 1) - \lambda_3 = 0
\end{aligned}$$

Then we can derive:

$$p_{0,0,0} = p_{0,0,1} \tag{6.21}$$

$$\log \frac{p_{0,1,1}}{p_{0,0,1}} = \lambda_2(\log \frac{q_1}{q_0}) \tag{6.22}$$

$$\log \frac{p_{1,1,1}}{p_{0,1,1}} = \lambda_1(\log \frac{r_1}{r_0}) \tag{6.23}$$

Let $\alpha = \lambda_1$ and $\beta = \lambda_2$:

$$p_{0,0,0} = p_{0,0,1} \tag{6.24}$$

$$p_{0,1,1} = \left(\frac{q_1}{q_0}\right)^{\beta} p_{0,0,1} \tag{6.25}$$

$$p_{1,1,1} = \left(\frac{r_1}{r_0}\right)^{\beta} p_{0,1,1} \tag{6.26}$$

Thus:

$$p_{0,0,0} = p_{0,0,1} = \frac{r_0^{\alpha} q_0^{\beta}}{Z(\alpha,\beta)} \tag{6.27}$$

$$p_{0,1,1} = \frac{r_0^{\alpha} q_1^{\beta}}{Z(\alpha,\beta)} \tag{6.28}$$

$$p_{1,1,1} = \frac{r_1^{\alpha} q_1^{\beta}}{Z(\alpha,\beta)} \tag{6.29}$$

$$\tag{6.30}$$

where $Z(\alpha) = 2r_0^{\alpha} q_0^{\beta} + r_0^{\alpha} q_1^{\beta} + r_1^{\alpha} q_1^{\beta}$.

Therefore, it also ends up searching for a suitable $\alpha$ and $\beta$, the same as the two high variable case with two constraints.

The same derivation can be applied to the situation where $X_1 \leq X_2 \leq X_3;\ X_1,\ X_2,\ X_3 \in [1, \ldots, n]$.

The derivation of more high variables with two marginal entropy constraints is more or less the same as above, hence the result is also similar.

## 6.2 General Case of Simple Problem

As in the case of single constraint problem, the further generalization is to generalize the linear constraints between program variables $X$ and $Y$ as $0 < X \leq aY + b$ ($a, b \in \mathcal{R}$). Again, our abstract domain in integer polyhedra. And for simplicity, we still assume that $X, Y \in [1, \ldots, n]$.

The same as above, let $p_{i,j} = P(X = i, Y = j)$, $r_i = P(X = i)$ and $q_j = P(Y = j)$. Since $0 < X \leq aY + b$, we may only concern $p_{i,j}$ such that $i \leq aj + b \wedge i,\ j \in \mathcal{N}$.

Suppose $\mathcal{H}(X) = \mathcal{H}(\vec{r}) = a$ and $\mathcal{H}(Y) = \mathcal{H}(\vec{q}) = b$, what is the maximum possible value for $\mathcal{H}(X, Y) = \mathcal{H}(\vec{p})$?

Define the Lagrangian $\Lambda$ as

$$\Lambda(\vec{p}) = \sum_{i,j\ aj+b \geq i}^{n} -p_{i,j} \log p_{i,j} - \lambda_1(\mathcal{H}(X) - a) - \lambda_2(\mathcal{H}(Y) - b) - \lambda_3(\sum_{i,j}^{n} p_{i,j} - 1) \tag{6.31}$$

Partial derivative of each $p_{i,j}$ gives rise to the following equations, $i \leq aj + b,\ i, j \in \{1, \ldots, n\}$:

$$\frac{\partial \Lambda}{\partial p_{i,j}} = -(\log p_{i,j} + \frac{1}{\ln 2}) + \lambda_1(\log r_i + \frac{1}{\ln 2}) + \lambda_2(\log q_j + \frac{1}{\ln 2}) - \lambda_3 = 0 \tag{6.32}$$

More or less the same reasoning as before gives us the following proportion equations:

$$\log \frac{p_{i,j}}{p_{i,j'}} = \lambda_2 \left( \log \frac{q_j}{q'_j} \right) \tag{6.33}$$

$$\log \frac{p_{i,j}}{p_{i',j}} = \lambda_1 \left( \log \frac{r_i}{r'_i} \right) \tag{6.34}$$

Then,

$$\frac{p_{i,j}}{p_{i,j'}} = \frac{q_j^{\beta}}{q_j'^{\beta}} \tag{6.35}$$

$$\frac{p_{i,j}}{p_{i',j}} = \frac{r_i^{\alpha}}{r_i'^{\alpha}} \tag{6.36}$$

Suppose $p_{i,j} = \frac{r_i^{\alpha} q_j^{\beta}}{Z(\alpha,\beta)}$ where $Z(\alpha, \beta) = \sum_{i,j}^{n} q_i^{\alpha} r_j^{\beta}$, it clearly satisfies the requirements of Equations 6.35; then we only need to show that there exists $\lambda_1$, $\lambda_2$, $\lambda_3$ such that the partial derivative of $p_{i,j}$ holds. Substitute into Equation 6.32:

$$\begin{aligned}
\frac{\partial \Lambda}{\partial p_{i,j}} = -(\frac{r_i^{\alpha} q_j^{\beta}}{Z(\alpha, \beta)} + \ln 2) + \lambda_1(r_i + \ln 2) + \lambda_2(q_j + \ln 2) - \lambda_3 = 0 \\
-(\alpha \log r_i + \beta \log q_j - \log Z(\alpha, \beta) + \frac{1}{\ln 2}) + \lambda_1(\log r_i + \frac{1}{\ln 2}) + \lambda_2(\log q_j + \frac{1}{\ln 2}) - \lambda_3 = 0 \\
(\lambda_1 - \alpha) \log r_i + (\lambda_2 - \beta) \log q_j + \log Z(\alpha, \beta) + \frac{\lambda_1 + \lambda_2 - 1}{\ln 2} - \lambda_3 = 0
\end{aligned}$$

Thus,

$\alpha = \lambda_1$

$\beta = \lambda_2$

$\lambda_3 = \log Z(\alpha, \beta) + \frac{\alpha+\beta-1}{\ln 2}$.

Therefore, $p_{i,j} = \frac{r_i^{\alpha} q_j^{\beta}}{Z(\alpha,\beta)}$ is the correct form for joint probability distribution.

### 6.2.1 Properties of $\mathcal{H}(X)$ and $\mathcal{H}(Y)$ in General Case

Similarly, in this sub-section, we prove the concaveness of $\mathcal{H}(X)$ and $\mathcal{H}(Y)$ using the same reasoning as the last section.

**Proof for $\mathcal{H}(X)$**

*Proof.*

$$\begin{aligned}
\mathcal{H}(X) &= -\sum_{i=1}^{n} \frac{r_i^\alpha \sum_{aj+b\geq i}^{n} q_j^\beta}{Z(\alpha,\beta)} \log \frac{r_i^\alpha \sum_{aj+b\geq i}^{n} q_j^\beta}{Z(\alpha,\beta)} \\
&= -\sum_{i=1}^{n} \frac{r_i^\alpha \sum_{aj+b\geq i}^{n} q_j^\beta}{Z(\alpha,\beta)} \left(\log r_i^\alpha \sum_{aj+b\geq i}^{n} q_j^\beta - \log Z(\alpha,\beta)\right) \\
&= \log Z(\alpha,\beta) - \sum_{i=1}^{n} \frac{r_i^\alpha \sum_{aj+b\geq i}^{n} q_j^\beta}{Z(\alpha,\beta)} \left(\log r_i^\alpha \sum_{aj+b\geq i}^{n} q_j^\beta\right) \\
&= \log Z(\alpha,\beta) - \sum_{i=1}^{n} \frac{r_i^\alpha \sum_{aj+b\geq i}^{n} q_j^\beta}{Z(\alpha,\beta)} \left(\alpha \log r_i + \log(\sum_{aj+b\geq i}^{n} q_j^\beta)\right) \\
&= \log Z(\alpha,\beta) - \sum_{i=1}^{n} \frac{\alpha r_i^\alpha \sum_{aj+b\geq i}^{n} q_j^\beta}{Z(\alpha,\beta)} \log r_i - \sum_{i=1}^{n} \frac{r_i^\alpha \sum_{aj+b\geq i}^{n} q_j^\beta}{Z(\alpha,\beta)} \log(\sum_{aj+b\geq i}^{n} q_j^\beta)
\end{aligned}$$

partial derivative of $\alpha$ is:

$$\begin{aligned}
\frac{\partial \mathcal{H}(Y)}{\partial \alpha} &= \frac{Z'(\alpha,\beta)_\alpha}{Z(\alpha,\beta)\ln 2} - \sum_{i=1}^{n} \frac{(r_i^\alpha + \alpha r_i^\alpha \ln r_i) Z(\alpha,\beta) - \alpha r_i^\alpha Z'(\alpha,\beta)_\alpha}{(Z(\alpha,\beta))^2 \ln 2} \sum_{aj+b\geq i}^{n} q_j^\beta \ln r_i \\
&\quad - \sum_{i=1}^{n} \frac{r_i^\alpha \ln r_i Z(\alpha,\beta) - r_i^\alpha Z'(\alpha,\beta)_\alpha}{(Z(\alpha,\beta))^2 \ln 2} \sum_{aj+b\geq i}^{n} q_j^\beta \ln \sum_{aj+b\geq i}^{n} q_j^\beta \\
&= -\sum_{i=1}^{n} \frac{\alpha r_i^\alpha \ln r_i Z(\alpha,\beta) - \alpha r_i^\alpha Z'(\alpha,\beta)_\alpha}{(Z(\alpha,\beta))^2 \ln 2} \sum_{aj+b\geq i}^{n} q_j^\beta \ln r_i \\
&\quad - \sum_{i=1}^{n} \frac{r_i^\alpha \ln r_i Z(\alpha,\beta) - r_i^\alpha Z'(\alpha,\beta)_\alpha}{(Z(\alpha,\beta))^2 \ln 2} \sum_{aj+b\geq i}^{n} q_j^\beta \ln \sum_{aj+b\geq i}^{n} q_j^\beta \\
&= -\sum_{i=1}^{n} \frac{r_i^\alpha \ln r_i \sum_{aj+b\geq i}^{n} q_j^\beta \ln r_i^\alpha}{(Z(\alpha,\beta))^2 \ln 2} Z(\alpha,\beta) - \sum_{i=1}^{n} \frac{r_i^\alpha \ln r_i \sum_{aj+b\geq i}^{n} q_j^\beta \ln \sum_{aj+b\geq i}^{n} q_j^\beta}{(Z(\alpha,\beta))^2 \ln 2} Z(\alpha,\beta) \\
&\quad + \sum_{i=1}^{n} \frac{r_i^\alpha \sum_{aj+b\geq i}^{n} q_j^\beta \ln r_i^\alpha}{(Z(\alpha,\beta))^2 \ln 2} Z'(\alpha,\beta)_\alpha + \sum_{i=1}^{n} \frac{r_i^\alpha \sum_{aj+b\geq i}^{n} q_j^\beta \ln \sum_{aj+b\geq i}^{n} q_j^\beta}{(Z(\alpha,\beta))^2 \ln 2} Z'(\alpha,\beta)_\alpha \\
&= -\sum_{i=1}^{n} \frac{r_i^\alpha \sum_{aj+b\geq i}^{n} q_j^\beta \ln r_i \ln(r_i^\alpha \sum_{aj+b\geq i}^{n} q_j^\beta) \sum_{i=1}^{n} r_i^\alpha \sum_{aj+b\geq i}^{n} q_j^\beta}{(Z(\alpha,\beta))^2 \ln 2} \\
&\quad + \sum_{i=1}^{n} \frac{r_i^\alpha \sum_{aj+b\geq i}^{n} q_j^\beta \ln(r_i^\alpha \sum_{aj+b\geq i}^{n} q_j^\beta) \sum_{i=1}^{n} r_i^\alpha (\ln r_i) \sum_{aj+b\geq i}^{n} q_j^\beta}{(Z(\alpha,\beta))^2 \ln 2}
\end{aligned}$$

As $r_i^\alpha \sum_{aj+b\geq i}^n q_j^\beta = r_i$, thus the above equation can be simplified as:

$$\begin{aligned}
&= -\sum_{i=1}^n \frac{r_i \ln r_i \ln(r_i^\alpha \sum_{aj+b\geq i}^n q_j^\beta) \sum_{i=1}^n r_i}{(Z(\alpha,\beta))^2 \ln 2} + \sum_{i=1}^n \frac{r_i \ln(r_i^\alpha \sum_{aj+b\geq i}^n q_j^\beta) \sum_{i=1}^n r_i \ln r_i}{(Z(\alpha,\beta))^2 \ln 2} \\
&= -\sum_{i=1}^n \frac{r_i \ln r_i (\alpha \ln r_i + \ln \sum_{aj+b\geq i}^n q_j^\beta) \sum_{i=1}^n r_i}{(Z(\alpha,\beta))^2 \ln 2} \\
&\quad + \sum_{i=1}^n \frac{r_i (\alpha \ln r_i + \ln \sum_{aj+b\geq i}^n q_j^\beta) \sum_{i=1}^n r_i \ln r_i}{(Z(\alpha,\beta))^2 \ln 2}
\end{aligned}$$

A careful calculation can get when the above equation is zero, the value of $\alpha$ is:

$$\alpha = \frac{\sum_{i\neq j,\, aj+b\geq i,\, i=1}^{n-1} r_i r_j \ln \frac{r_j}{r_i} \ln \frac{\sum_{m=i}^n q_m^\beta}{\sum_{l=j}^n q_l^\beta}}{\sum_{i\neq j,\, aj+b\geq i,\, i=1}^{n-1} r_i r_j \left(\ln \frac{r_j}{r_i}\right)^2}$$

Thus when $\alpha > \frac{\sum_{i\neq j,\, aj+b\geq i,\, i=1}^{n-1} r_i r_j \ln \frac{r_j}{r_i} \ln \frac{\sum_{m=i}^n q_m^\beta}{\sum_{l=j}^n q_l^\beta}}{\sum_{i\neq j,\, aj+b\geq i,\, i=1}^{n-1} r_i r_j \left(\ln \frac{r_j}{r_i}\right)^2}$, the above partial derivative is always $< 0$ while when $\alpha < \frac{\sum_{i\neq j,\, aj+b\geq i,\, i=1}^{n-1} r_i r_j \ln \frac{r_j}{r_i} \ln \frac{\sum_{m=i}^n q_m^\beta}{\sum_{l=j}^n q_l^\beta}}{\sum_{i\neq j,\, aj+b\geq i,\, i=1}^{n-1} r_i r_j \left(\ln \frac{r_j}{r_i}\right)^2}$, the above partial derivative is always $> 0$.

□

**Proof for $\mathcal{H}(Y)$**

*Proof.*

$$\begin{aligned}
\mathcal{H}(Y) &= -\sum_{i=1}^n \frac{q_i^\beta \sum_{j\leq ai+b} r_j^\alpha}{Z(\alpha,\beta)} \log \frac{q_i^\beta \sum_{j\leq ai+b} r_j^\alpha}{Z(\alpha,\beta)} \\
&= -\sum_{i=1}^n \frac{q_i^\beta \sum_{j\leq ai+b} r_j^\alpha}{Z(\alpha,\beta)} \left(\log q_i^\beta \sum_{j\leq ai+b}^n r_j^\alpha - \log Z(\alpha,\beta)\right) \\
&= \log Z(\alpha,\beta) - \sum_{i=1}^n \frac{q_i^\beta \sum_{j\leq ai+b}^n r_j^\alpha}{Z(\alpha,\beta)} \left(\log q_i^\beta \sum_{j\leq ai+b}^n r_j^\alpha\right) \\
&= \log Z(\alpha,\beta) - \sum_{i=1}^n \frac{q_i^\beta \sum_{j\leq ai+b}^n r_j^\alpha}{Z(\alpha,\beta)} \left(\beta \log q_i + \log(\sum_{j\leq ai+b}^n r_j^\alpha)\right) \\
&= \log Z(\alpha,\beta) - \sum_{i=1}^n \frac{\beta q_i^\beta \sum_{j\leq ai+b}^n r_j^\alpha}{Z(\alpha,\beta)} \log q_i - \sum_{i=1}^n \frac{q_i^\beta \sum_{j\leq ai+b}^n r_j^\alpha}{Z(\alpha,\beta)} \log(\sum_{j\leq ai+b}^n r_j^\alpha)
\end{aligned}$$

partial derivative of $\beta$ is:

$$\begin{aligned}
\frac{\partial \mathcal{H}(Y)}{\partial \beta} &= \frac{Z'(\alpha,\beta)_\beta}{Z(\alpha,\beta)\ln 2} - \sum_{i=1}^{n} \frac{(q_i^\beta + \beta q_i^\beta \ln q_i) Z(\alpha,\beta) - \beta q_i^\beta Z'(\alpha,\beta)_\beta}{(Z(\alpha,\beta))^2 \ln 2} \sum_{j\leq ai+b}^{n} r_j^\alpha \ln q_i \\
&\quad - \sum_{i=1}^{n} \frac{q_i^\beta \ln q_i Z(\alpha,\beta) - q_i^\beta Z'(\alpha,\beta)_\beta}{(Z(\alpha,\beta))^2 \ln 2} \sum_{j\leq ai+b}^{n} r_j^\alpha \ln \sum_{j\leq ai+b}^{n} r_j^\alpha \\
&= -\sum_{i=1}^{n} \frac{\beta q_i^\beta \ln q_i Z(\alpha,\beta) - \beta q_i^\beta Z'(\alpha,\beta)_\beta}{(Z(\alpha,\beta))^2 \ln 2} \sum_{j\leq ai+b}^{n} r_j^\alpha \ln q_i \\
&\quad - \sum_{i=1}^{n} \frac{q_i^\beta \ln q_i Z(\alpha,\beta) - q_i^\beta Z'(\alpha,\beta)_\beta}{(Z(\alpha,\beta))^2 \ln 2} \sum_{j\leq ai+b}^{n} r_j^\alpha \ln \sum_{j\leq ai+b}^{n} r_j^\alpha \\
&= -\sum_{i=1}^{n} \frac{q_i^\beta \ln q_i \sum_{j\leq ai+b}^{n} r_j^\alpha \ln q_i^\beta}{(Z(\alpha,\beta))^2 \ln 2} Z(\alpha,\beta) - \sum_{i=1}^{n} \frac{q_i^\beta \ln q_i \sum_{j\leq ai+b}^{n} r_j^\alpha \ln \sum_{j\leq ai+b}^{n} r_j^\alpha}{(Z(\alpha,\beta))^2 \ln 2} Z(\alpha,\beta) \\
&\quad + \sum_{i=1}^{n} \frac{q_i^\beta \sum_{j\leq ai+b}^{n} r_j^\alpha \ln q_i^\beta}{(Z(\alpha,\beta))^2 \ln 2} Z'(\alpha,\beta)_\beta + \sum_{i=1}^{n} \frac{q_i^\beta \sum_{j\leq ai+b}^{n} r_j^\alpha \ln \sum_{j\leq ai+b}^{n} r_j^\alpha}{(Z(\alpha,\beta))^2 \ln 2} Z'(\alpha,\beta)_\beta \\
&= -\sum_{i=1}^{n} \frac{q_i^\beta \sum_{j\leq ai+b}^{n} r_j^\alpha \ln q_i \ln(q_i^\beta \sum_{j\leq ai+b}^{n} r_j^\alpha) \sum_{i=1}^{n} q_i^\beta \sum_{j\leq ai+b}^{n} r_j^\alpha}{(Z(\alpha,\beta))^2 \ln 2} \\
&\quad + \sum_{i=1}^{n} \frac{q_i^\beta \sum_{j\leq ai+b}^{n} r_j^\alpha \ln(q_i^\beta \sum_{j\leq ai+b}^{n} r_j^\alpha) \sum_{i=1}^{n} q_i^\beta (\ln q_i) \sum_{j\leq ai+b}^{n} r_j^\alpha}{(Z(\alpha,\beta))^2 \ln 2}
\end{aligned}$$

As $q_i^\beta \sum_{j\leq ai+b}^{n} r_j^\alpha = q_i$, thus the above equation can be simplified as:

$$\begin{aligned}
&= -\sum_{i=1}^{n} \frac{q_i \ln q_i \ln(q_i^\beta \sum_{j\leq ai+b}^{n} r_j^\alpha) \sum_{i=1}^{n} q_i}{(Z(\alpha,\beta))^2 \ln 2} + \sum_{i=1}^{n} \frac{q_i \ln(q_i^\beta \sum_{j\leq ai+b}^{n} r_j^\alpha) \sum_{i=1}^{n} q_i \ln q_i}{(Z(\alpha,\beta))^2 \ln 2} \\
&= -\sum_{i=1}^{n} \frac{q_i \ln q_i (\beta \ln q_i + \ln \sum_{j\leq ai+b}^{n} r_j^\alpha) \sum_{i=1}^{n} q_i}{(Z(\alpha,\beta))^2 \ln 2} \\
&\quad + \sum_{i=1}^{n} \frac{q_i (\beta \ln q_i + \ln \sum_{j\leq ai+b}^{n} r_j^\alpha) \sum_{i=1}^{n} q_i \ln q_i}{(Z(\alpha,\beta))^2 \ln 2}
\end{aligned}$$

A careful calculation can get, when the above equation is zero, the value of $\beta$ is:

$$\beta = \frac{\sum_{i\neq j,\, j\leq ai+b,\, i=1}^{n-1} q_i q_j \ln \frac{q_j}{q_i} \ln \frac{\sum_{m=1}^{i} r_m^\alpha}{\sum_{l=1}^{j} r_l^\alpha}}{\sum_{i\neq j,\, j\leq ai+b,\, i=1}^{n-1} q_i q_j \left(\ln \frac{q_j}{q_i}\right)^2}$$

Thus when $\beta > \frac{\sum_{i\neq j,\, j\leq ai+b,\, i=1}^{n-1} q_i q_j \ln \frac{q_j}{q_i} \ln \frac{\sum_{m=1}^{i} r_m^\alpha}{\sum_{l=1}^{j} r_l^\alpha}}{\sum_{i\neq j,\, j\leq ai+b,\, i=1}^{n-1} q_i q_j \left(\ln \frac{q_j}{q_i}\right)^2}$, the above partial derivative is always $< 0$ while when $\beta < \frac{\sum_{i\neq j,\ j\leq ai+b,\ i=1}^{n-1} q_i q_j \ln\frac{q_j}{q_i}\ln\frac{\sum_{m=1}^{i} r_m^\alpha}{\sum_{l=1}^{j} r_l^\alpha}}{\sum_{i\neq j,\ j\leq ai+b,\ i=1}^{n-1} q_i q_j \left(\ln\frac{q_j}{q_i}\right)^2}$, the above partial derivative is always $> 0$. □

### 6.2.2 Three Variable Case of Simple Problem

In this section, we extend the discussion above to three variable case.

We still use $X_1$, $X_2$ and $X_3$ to represent three high variables such that $c'X_1 + d' \leq X_2 \leq cX_3 + d$ and all of them $\in [1, n]$, $r_i$, $q_i, p_{i,j,k}$ to represent $P(X_1 = i), P(X_2 = j), P(X_1 = i, X_2 = j, X_3 = k)$ respectively, and with constraints $\mathcal{H}(X_1) = a$ and $\mathcal{H}(X_2) = b$ (it can be entropies of any two of the three). Please note that because $c'X_1 + d \leq X_2 \leq cX_3 + d$, we only interested in $p_{i,j,k}$ such that $c'i + d' \leq j \leq ck + d$.

The Lagrangian is:

$$\Lambda = -\sum_{i,j,k} p_{i,j,k}\log p_{i,j,k} - \lambda_1(\mathcal{H}(X_1) - a) - \lambda_2(\mathcal{H}(X_2) - b) - \lambda_3(\sum_{i,j,k} p_{i,j,k} - 1) \tag{6.37}$$

which gives rise to the following family of partial derivative equations, $c'i + d' \leq j \leq ck + d, i,\ j,\ k \in [0, 1]$:

$$\frac{\partial\Lambda}{\partial p_{i,j,k}} = -(\log p_{i,j,k} + \frac{1}{\ln 2}) + \frac{\lambda_1}{\ln 2}(\ln r_i + 1) + \frac{\lambda_2}{\ln 2}(\ln q_j + 1) - \lambda_3 = 0$$

Then we can derive:

$$p_{i,j,k} = p_{i,j,k'} \; k \neq k' \tag{6.38}$$

$$\log \frac{p_{i,j,k}}{p_{i,j',k}} = \lambda_2 (\log \frac{q_j}{q'_j}) \tag{6.39}$$

$$\log \frac{p_{i,j,k}}{p_{i',j,k}} = \lambda_1 (\log \frac{r_i}{r'_i}) \tag{6.40}$$

Let $\alpha = \lambda_1$ and $\beta = \lambda_2$:

$$p_{i,j,k} = p_{i,j,k'} \; k \neq k' \tag{6.41}$$

$$p_{i,j',k} = \left(\frac{q'_j}{q_j}\right)^{\beta} p_{i,j,k} \tag{6.42}$$

$$p_{i',j,k} = \left(\frac{r'_i}{r_i}\right)^{\beta} p_{i,j,k} \tag{6.43}$$

Thus:

$$p_{i,j,k} = p_{i,j,k'} = \frac{r_i^{\alpha} q_j^{\beta}}{Z(\alpha, \beta)} \tag{6.44}$$

where $Z(\alpha) = \sum_{i,j,k} p_{i,j,k}$.

Therefore, it also ends up searching for a suitable $\alpha$ and $\beta$, the same as the two high variable case with two constraints.

The derivation of more high variables with two marginal entropy constraints is more or less the same as above, hence the result is also similar.

## 6.3 Generalization of Simple Problem

The further generalization is to generalize the linear constraints between program variables $X$ and $Y$ to an arbitrary system of $0 < a'_i Y + b'_i < X \leq a_i Y + b_i, \ (a_i, \ a'_i, \ b_i, \ b'_i \in \mathcal{R})$. And $X, Y \in [1, \ldots, n]$.

The whole derivation procedure is exactly the same as the last section which is omitted for the sake of space. The only thing which changes is that in stead of considering $p_{i,j}$ such that $i \leq aj + b$, now we consider $p_{i,j}$ such that $a'_i j + b'_i < i \leq a_i j + b_i$.

Please note that if $i \leq aj + b_1$ and $i \leq aj + b_2$ where $b_1 \leq b_2$, then the constraint $i \leq aj + b_2$ will not be considered, similarly for other cases. The work of [6] can be used to derive a most precise integer polyhedra from the system of constraints. Therefore, in our analysis, we always assume that we work on the smallest integer polyhedra derived from the system of constraints.

The result $p_{i,j} = \frac{r_i^\alpha q_j^\beta}{Z(\alpha,\beta)}$ still holds, where $Z(\alpha, \beta) = \sum_{i,j}^n q_i^\alpha r_j^\beta$.

## 6.4 Case Study

In this section, the case of $X$ and $Y$ taking two values each is explored, some interesting properties have been discovered. We choose this as a demonstration also because it is not too difficult to verify.

Recall the entropy of binary variable (Figure 6.1), we can see that for each value of entropy, there are only two corresponding possible probability distributions. Thus there are at most four joint probabilities from which to choose the one with maximum joint entropy.

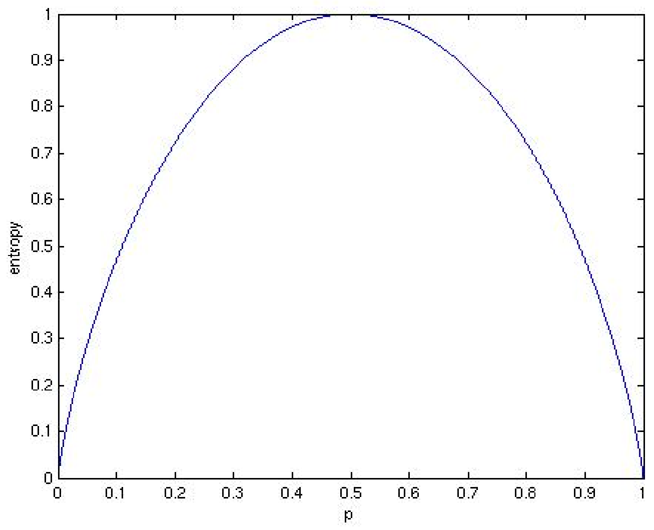

Figure 6.1: Concaveness of Entropy Function

The following Table 6.1 shows the values of $\alpha$ and $\beta$ for the maximum joint entropy given different values of marginal entropy constraints.

Column $\mathcal{H}(X)$ is the value of the marginal constraint of $X$, we start from 0.4690 and increase it to its maximum 1, other values can be chosen freely as well.

Similarly, Column $\mathcal{H}(Y)$ is the value of the marginal constraint of $Y$, we start from 0.4690 and increase it to its maximum 1, other values can be chosen freely as well. Together with $\mathcal{H}(X)$, they work as a pair of constraints.

Column $\alpha$ and column $\beta$ are the corresponding values for $\alpha$ and $\beta$ that gives us the maximum joint entropy under two marginal constraints, where $-$ means such $\alpha$ or $\beta$ don't exist.

As we can see from Table 6.1, there are cases when there are no suitable

| $\mathcal{H}(X)$ | $\mathcal{H}(Y)$ | $\alpha$ | $\beta$ | $\mathcal{H}_{\max(X,Y)}$ | $\mathcal{H}(X)+\mathcal{H}(Y)$ | $P(X,Y)$ |
|---|---|---|---|---|---|---|
| 0.4690 | 0.4690 | 0 | -0.9464 | 0.9219 | 0.9380 | (0.1,0.8,0.1) |
| 0.4690 | 0.7219 | 0.8856 | 0.9037 | 1.1568 | 1.1909 | (0.2,0.7,0.1) |
| 0.4690 | 0.8813 | 0.8155 | 0.8181 | 1.2955 | 1.3503 | (0.3,0.6,0.1) |
| 0.4690 | 0.9710 | 0.7325 | 0.5503 | 1.3610 | 1.4400 | (0.4,0.5,0.1) |
| 0.4690 | 1 | - | - | 1.3610 | 1.4690 | (0.5,0.4,0.1) |
| 0.7219 | 0.4690 | 0.9037 | 0.8856 | 1.1568 | 1.1909 | (0.1,0.7,0.2) |
| 0.7219 | 0.7219 | 0.7925 | 0.7925 | 1.3710 | 1.4438 | (0.2,0.6,0.2) |
| 0.7219 | 0.8813 | 0.6610 | 0.6029 | 1.4855 | 1.6032 | (0.3,0.5,0.2) |
| 0.7219 | 0.9710 | 0.5 | 0 | 1.5219 | 1.6929 | (0.4,0.4,0.2) |
| 0.7219 | 1 | - | - | 1.4855 | 1.7219 | (0.5,0.3,0.2) |
| 0.8813 | 0.4690 | 0.8181 | 0.8155 | 1.2955 | 1.3503 | (0.1,0.6,0.3) |
| 0.8813 | 0.7219 | 0.6029 | 0.6610 | 1.4855 | 1.6032 | (0.2,0.5,0.3) |
| 0.8813 | 0.8813 | 0.3395 | 0.3395 | 1.5710 | 1.7626 | (0.3,0.4,0.3) |
| 0.8813 | 0.9710 | 0 | -0.7095 | 1.5710 | 1.8523 | (0.4,0.3,0.3) |
| 0.8813 | 1 | - | - | 1.4855 | 1.8813 | (0.5,0.2,0.3) |
| 0.9710 | 0.4690 | 0.5503 | 0.7325 | 1.3610 | 1.4400 | (0.1,0.5,0.4) |
| 0.9710 | 0.7219 | 0 | 0.5 | 1.5219 | 1.6929 | (0.2,0.4,0.4) |
| 0.9710 | 0.8813 | -0.7095 | 0 | 1.5710 | 1.8523 | (0.3,0.3,0.4) |
| 0.9710 | 0.9710 | -1.7095 | -1.7095 | 1.5219 | 1.9420 | (0.4,0.2,0.4) |
| 0.9710 | 1 | - | - | 1.3610 | 1.9710 | (0.5,0.4,0.1) |
| 1 | 0.4690 | - | - | 1.3610 | 1.4690 | (0.1,0.4,0.5) |
| 1 | 0.7219 | - | - | 1.4855 | 1.7219 | (0.2,0.3,0.5) |
| 1 | 0.8813 | - | - | 1.4855 | 1.8823 | (0.3,0.2,0.5) |
| 1 | 0.9710 | - | - | 1.3610 | 1.9710 | (0.4,0.1,0.5) |
| 1 | 1 | - | - | 1 | 2 | (0.5,0,0.5) |

Table 6.1: Values of $\alpha$ and $\beta$ for Maximum Joint Entropy

$\alpha$ or $\beta$, this situation usually happens when one of the marginal entropy hits its maximum, which means that in the case where $X$ and $Y$ are binary, there is only one joint probability distribution that satisfies both marginal entropies, hence there is only one joint entropy which is also the maximum joint entropy.

**Proposition 6.4.1.** *Let $X$ and $Y$ be binary variables such that $X \leq Y$, let $\mathcal{H}(X) = a$ and $\mathcal{H}(Y) = b$, suppose $\alpha$ and $\beta$ corresponding to the maximum joint entropy $\mathcal{H}(X,Y)$ exist, then given two other binary variables $X'$ and $Y'$, if $\mathcal{H}(X') = b$ and $\mathcal{H}(Y') = a$, $\alpha'$ and $\beta'$ for maximum joint entropy $\mathcal{H}(X',Y')$ also exist, then $\alpha' = \beta$ and $\beta' = \alpha$.*

*Proof.* Because $X$ and $Y$ are binary variables, given their marginal entropies, there are at most four possible joint entropies, let's denote the corresponding probabilities as $P(X=0) = x$, $P(X=1) = 1-x$, $P(Y=0) = y$, $P(Y=1) = 1-y$, thus the following holds:

$$P(X=0, Y=0) = y = \frac{x^{\alpha} * y^{\beta}}{Z(\alpha,\beta)}$$

$$P(X=0, Y=1) = x - y = \frac{x^{\alpha} * (1-y)^{\beta}}{Z(\alpha,\beta)}$$

$$P(X=1, Y=1) = 1 - x = \frac{(1-x)^{\alpha} * (1-y)^{\beta}}{Z(\alpha,\beta)}$$

where $Z(\alpha,\beta) = \sum_{i,j} P(X=i)^{\alpha} * P(Y=j)^{\beta}$

Now since the values of two marginal entropies are the same if we don't consider which one takes which value, if the above joint distribution is the one with maximum joint entropy for $X$ and $Y$, it also should be the joint distribution with maximum joint entropy for $X'$ and $Y'$. In order to get the same joint distribution (don't consider which number takes which probability), we can construct the probabilities as $P(X'=0)=1-y$, $P(X'=1)=y$, $P(Y'=0)=1-x$ and $P(Y'=1)=1-y$, thus the following holds:

$$1-x=\frac{(1-y)^{\alpha'}*(1-x)^{\beta'}}{Z(\alpha',\beta')}$$

$$x-y=\frac{(1-y)^{\alpha'}*x^{\beta'}}{Z(\alpha',\beta')}$$

$$y=\frac{y^{\alpha'}*x^{\beta'}}{Z(\alpha',\beta')}$$

where $Z(\alpha,\beta)=\sum_{i,j}P(X'=i)^{\alpha'}*P(Y'=j)^{\beta'}$.

Comparing with the joint distribution for $X$ and $Y$, it is not difficult to see that $\alpha'=\beta$ and $\beta'=\alpha$. $\square$

# Chapter 7

# Conclusions

This chapter summarizes the contributions of this thesis and outlines some directions for possible future work.

## 7.1 Summary

Focused on the problem of quantifying maximum information leakage in programs, this thesis connects information security with Shannon's information theory and mathematical techniques such as Lagrange multiplier.

Noticing the intrinsic issues (*i.e.* double counting problem) that exist against current framework of quantitative information flow analysis (CHM's), this thesis looks to improve the accuracy of maximum information leakage. More precisely, we concern quantifying maximum information leakage under certain constraints, maximum entropy of individual high program variable, which is non-linear.

The improvement of accuracy is achieved by using joint entropy as the quantity of maximum information leakage instead of the sum of entropies of high program variables; then the Lagrange multiplier method is applied to obtain the maximum information leakage under constraints. For the single marginal entropy constraint problem, the partition version of joint entropy is applied (Section 5.3) to further simplify the derivation. We have shown that for the single marginal constraint problem, the maximum joint entropy problem can be brought down to search for a parameter $\alpha$ which when applied to the structure of probability distribution derived from our analysis, guarantees the maximum information leakage. For the two marginal constraints problem, it turns out that a search for a pair of parameters $\alpha$ and $\beta$ gives us the result.

As a matter of fact, our analysis is general enough to be applied to arbitrary number of high program variables with an arbitrary system of linear constraints under arbitrary number of marginal entropy constraints. However, please note that in order to get the maximum information leakage as precise as possible, the system of linear constraints is required to be as small as possible, in other words, for example if there are both $X \leq cY + d$ and $X \leq cY + d'$ in the set, where $d \leq d'$ then $X \leq cY + d'$ is discarded. Therefore in our work, we always assume that the linear constraint set is the most precise, and hence the integer polyhedron derived from it that we work on is also the smallest.

### 7.1.1 Most General Case

Therefore, the most general case would be:

Suppose we have $n$ variables $X_1, \ \ldots, X_n \in \mathcal{N}$, and a system of linear constraints $S$ $(X_i \leq a_1X_1 + \ldots + a_nX_n + b_i, \ a_i, b_i \in \mathcal{R})$ between these variables, together with $m$ marginal entropy constraints, let's denote the variables which have marginal entropy constraints as $X_{l_1}, \ldots, X_{l_m}$ and of course $m \leq n$. $p_{i_1,\ldots,i_n}$ denotes the joint probability of $P(X_1 = x_1, \ \ldots, \ X_n = x_n)$, and $p_{l_i}$to represent the probability of $X_{l_i} = x_{l_i}$.

By following the exact derivation process, it is not too difficult to obtain the general form of probability distribution which gives the maximum joint entropy:

$$p_{i_1,\, \ldots,\, l_1,\ldots,\, l_m,\, \ldots,i_n} = p_{i'_1,\, \ldots,\, l_1,\ldots,\, l_m,\, \ldots,i'_n} = \frac{p_{l_1}^{\alpha_1} \cdots p_{l_m}^{\alpha_m}}{Z(\alpha_1, \ldots, \alpha_m)}$$

Where $Z(\alpha_1, \ldots, \alpha_m) = \sum_{i_1,\ldots,i_n} p_{i_1,\ldots,i_n}$.

## 7.1.2 Limitation

Although theoretically our analysis is very general, and an analytical formula is provided to construct the probability distribution which gives the maximum joint entropy. However, only the single constraint problem is feasible, a search for a single parameter $\alpha$ which a binary search does the job.

For more than one marginal entropy constraint, tuple of parameters needed to obtain the probability distributions, and we haven't found an efficient method to search for such tuple of parameters yet, and it is also possible that some parameters don't exist, nevertheless the probability distribution can still be obtained without these parameters sometimes as shown in Chapter 6.

## 7.2 Future Work

The future work from ours may include the following:

### 7.2.1 Scalability

Tables 5.1 to 5.6 show that when the number $n$ of values that $X$ and $Y$ can take increases in the single constraint problem, the ratio of our result and CHM's result keeps increasing as well.

This is mainly due to the fact that when $n \to \infty$, and suppose that we do the partition according to $Y$ (or $X$), no matter what linear constraint that may exist between $X$ and $Y$, the size of largest part in the partition goes towards $\infty$ as well. By adopting our analysis, the marginal probability distribution $q_i$ of $Y$ which purely depends on the sizes of the parts will lean towards $0, 0, \ldots, 1$. Then our analysis whose result is $\mathcal{H}_{\max}(X, Y) = \mathcal{H}(Y) + \sum_i q_i \log i$, will become $\mathcal{H}(Y) + \log \infty$ which is $\infty$. And CHM's analysis which is $\mathcal{H}(Y) + \log n$, also has the same result of $\infty$. Therefore in this extreme case, our result doesn't have any advantage over CHM's.

Hence it seems that our analysis only has a significant superiority when $n$ is small. When $n$ increase, the maximum joint entropy also increases, thus a completely new way to deal with large $n$ with the aim to make the result more precise than that of our current analysis is needed.

### 7.2.2 Effective Search Method for More Constraints Problem and Complexity

Our analysis is theoretically quite general in terms of the system of linear constraints and marginal entropy constraints. And it all has the nice property such that in the end only tuple of parameters are needed to construct the probability distribution which gives the maximum joint entropy. Therefore, it is worth investigating how to effectively search for such tuple of suitable parameters.

For the two marginal entropy constraints problem, as we cannot derive the structure of either marginal probability of $X$ or $Y$ at the moment, this makes the search of suitable $\alpha$ and $\beta$ extremely difficult. One possible area for future investigation is to transform the question into a single constraint problem, and search for $\alpha$ and $\beta$ alternatively. In more detail, two random marginal probability distributions (except uniform distribution) can be chosen, one for $X$ the other for $Y$. First we fix $\beta$ to 1, and substitute these two original distributions into the formula:

$$p_{i,\,j} = \frac{r_i^{\alpha} q_j^{\beta}}{Z(\alpha, \beta)}$$

then do a binary search, as in the single constraint problem, to look for a marginal distribution for $X$ such that it satisfies the marginal entropy $\mathcal{H}(X)$ constraint. Then this new marginal probability $P(X)$ is substituted into $r_i$, and $\alpha$ is fixed to 1, together with the original marginal probability distribution $P(Y)$, a new binary search for $\beta$ which to make the new probability distribution satisfy the marginal entropy constraint $\mathcal{H}(Y)$ is conducted the same way as that for $\mathcal{H}(X)$ constraint, then the new marginal probability distribution of $P(Y)$ is substituted into $q_j$.

This whole process is, thereafter, iterated, until $\alpha$ and $\beta$ converges, in other words, the two marginal probability distributions calculated from joint probability distribution converge. However, investigation into the convergence issue of this proposed thought algorithm will be needed first which itself can be a very difficult problem.

The investigation of more constraint problem can be more difficult and complicated.

The complexity of search algorithm can grow exponentially with the number of marginal entropy constraints, methods to reduce the complexity is also worth working on.

### 7.2.3 Implementation

The ultimate goal would be to implement our analysis into a practical analysis tool, which automatically bounds the maximum information leakage that certain programs or systems may have.

The automation for the single constraint problem can be straightforward, breaks the problem into automation for three parts: linear constraints (integer polyhedron) obtained by abstract interpretation, a partition based on the integer polyhedron, and a search for $\alpha$. The automation of first part is already been implemented in [22]; while the automation of the second part can be turned into solve a system of linear inequalities for which automatic algorithms exist; while for the third part a binary search algorithm satisfies our needs.

However, for the problems with more marginal constraints, the automation has to search for tuple of parameters simultaneously, hence it is very likely that some new algorithm would be needed to solve the problem.

# Bibliography

[1] S. Abramsky and D. M. Gabbay and T. S. E. Maibaum. Handbook of logic in computer science: semantic modeling. *Handbook of logic in computer science (vol. 4): semantic modeling*, ISBN: 0-19-853780-8. Oxford University Press, Oxford, UK, 1995.

[2] D. P. Bertsekas. Constrained Optimization and Lagrange Multiplier Methods, ISBN:1-886529-04-3. *Academic Press*, 1996.

[3] D. P. Bertsekas, A. Nedic and A. E. Ozdaglar. Convex Analysis and Optimization, ISBN: 1-886529-45-0. *Athena Scientific*, April 2003.

[4] M. Boreale. Quantifying information leakage in process calculi. *Information and Computation*, 207(6): 699-725, 2009.

[5] S. Boyd and L. Vandenberghe. Convex optimization. ISBN: 978-0521833783. Cambridge University Press. March, 2004.

[6] P. J. Charles and J. M. Howe and A. King. Integer Polyhedra for Program Analysis. In *Proceedings of the 5th International Conference on Algorithmic Aspects in Information and Management*, San Francisco, CA, USA, 2009.

[7] H. Chen and P. Malacaria. Quantitative analysis of leakage for multi-threaded programs. In *PLAS '07: Proceedings of the 2007 workshop on Programming languages and analysis for security*, pages 31-40. San Diego, California, USA, 2007.

[8] H. Chen and P. Malacaria. The optimum leakage principle for analyzing multi-threaded programs. In *Proc. of the 4th International Conference on Information Theoretical Security*, pages 177-193, Shizuoka, Japan, 2009.

[9] H. Chen and P. Malacaria. Quantifying maximal loss of anonymity in protocols. In *ASIACCS '09: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pages 206-217, Sydney, Australia, 2009.

[10] H. Chen and P. Malacaria. Studying maximum information leakage using Karush-Kuhn-Tucker conditions. In *Proceedings of 7th International Workshop on Security Issues in Concurrency*, pages 1-15, Bologna, 2009.

[11] D. Clark and S. Hunt and P. Malacaria. Quantitative analysis of the leakage of confidential data. QAPL'01, Quantitative Aspects of Programming Languages (Satellite Event of PLI 2001), *Electronic Notes in Theoretical Computer Science*, 59(3):238-251, Firenze, Italy, 2001.

[12] D. Clark and S. Hunt and P. Malacaria. Quantified interference: information theory and information flow. *Presented at Workshop on Issues in the Theory of Security (WITS'04)*, 2004.

[13] D. Clark and S. Hunt and P. Malacaria. Quantified interference for a while language. Electronic Notes in Theoretical Computer Science, 112:149-166, 2005.

[14] D. Clark and S. Hunt and P. Malacaria. Quantitative information flow, relations and polymorphic types. *Journal of Logic and Computation, Special Issue on Lambda-calculus, type theory and natural language*, 18(2):181-199, 2005.

[15] D. Clark and S. Hunt and P. Malacaria. A static analysis for quantifying information flow in a simple imperative language. *Journal of Computer Security*, 15(3):321-371, 2007.

[16] D. Clark and S. Hunt and P. Malacaria. Quantified interference for a while language. *Technical Report TR-03-07*, Department of Computer Science, King's College London, King's College London, 2003.

[17] P. Cousot and R. Cousot. Static determination of dynamic properties of programs. In *Proceedings of the Second International Symposium on Programming*, pages 106-130. Dunod, France, 1976.

[18] P. Cousot and R. Cousot. Abstract interpretation: an unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 238-252, 1977.

[19] P. Cousot and R. Cousot. Automatic synthesis of optimal invariant assertions: Mathematical foundations. In *Proceedings of the 1977 symposium*

*on Artificial intelligence and programming languages*, pages 1-12, New York, USA.

[20] P. Cousot and R. Cousot. Systematic design of program analysis frameworks. *POPL '79: Proceedings of the 6th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 269-282, New York, USA, 1979.

[21] P. Cousot and R. Cousot. Comparing the Galois connection and widening/narrowing approaches to abstract interpretation, invited paper. In *Proceedings of the International Workshop Programming Language Implementation and Logic Programming, PLILP'92*, LNCS 631: 269-295. Leuven, Belgium, 1992.

[22] P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Conference Record of the Fifth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. Tucson, Arizona, 1978.

[23] T.M. Cover, J.A. Thomas. Elements of Information Theory. *Wiley-Interscience*, ISBN: 0-471-24195-4. New York, NY, USA, 2006.

[24] D. E. R. Denning. Secure information flow in computer systems. PhD thesis, Purdue University. West Lafayette, IN, USA, 1975.

[25] D. E. R. Denning. A lattice model of secure information flow. *Communications of the ACM*, 19(5): 236-243, 1976.

[26] D. E. R. Denning. Cryptography and data security, ISBN: 0-201-10150-5. *Addison-Wesley Longman Publishing Co., Inc.* Boston, MA, USA, 1982.

[27] D. E. R. Denning and P. J. Denning. Certification of programs for secure information flow. *Communications of the ACM*, 20(7): 504-513, 1977.

[28] D. E. R. Denning and P. J. Denning. Data security. *ACM Computing Surveys*, 11(3): 227-249, 1979.

[29] R. Giacobazzi and I. Mastroeni. Abstract non-interference: parameterizing non-interference by abstract interpretation. *Annual Symposium on Principles of Programming Languages, Proceedings of the 31st ACM symposium on Principles of programming languages*, pages 186-197, Venice, Italy, 2004.

[30] J. A. Goguen and J. Meseguer. Security Policies and Security Models. In *IEEE Symposium on Security and Privacy*, pages 11-20. 1982.

[31] J. W. Gray III. Towards a mathematical foundation for information flow security. In *Proc. of IEEE Symposium on Security and Privacy*, pages 21-34. IEEE Computer Society Press, 1991.

[32] N. Halbwachs. Determination automatique de relations linear verifiees par les variables d'un programme. PhD thesis, OCLC, 1979.

[33] N. Halbwachs, Y. E. Proy and P. Raymond. Verification of linear hybrid systems by means of convex approximation.*SAS*, pages 223-237, 1994.

[34] C.Hankin, A. Di Pierro and H. Wiklicky. Measuring the confinement of probabilistic systems. *Theoretical Computer Science*, 340(1): 3-56, Essex, UK, 2005.

[35] J. M. Howe and A. King. A semantic basis for specialising domain constraints. In *the International Workshop for Object-oriented and Constraint Programming for Time Critical Applications*, Lisbon, Portugal, 1999.

[36] J. M. Howe and A. King. Specialising finite domain programs using polyhedra. *LOPSTR* , pages 118-135, Venezia, Italy, 1999.

[37] S. Hunt and D. Sands. On flow-sensitive security types. In *Proc. Principles of Programming Languages, 33rd Annual ACM Symposium (POPL'06)*, pages 79-90. Charleston, South Carolina, USA, 2006.

[38] S. Hunt and I. Mastroeni. The PER model of abstract non-interference. *Proc. Static Analysis, 12th International Symposium (SAS'05)*, LNCS 3184: 100-115. Springer-Verlag, 2005.

[39] N. D. Jones and F. Nielson. Abstract Interpretation: A Semantics-Based Tool for Program Analysis. *Handbook of logic in computer science (vol. 4): semantic modeling*, pages 527-636, Oxford University Press, Oxford, UK, 1995.

[40] R. Joshi and K. R. M. Leino. A semantic approach to secure information flow. *Science of Computer Programming*, 37(1-3): 113-138, 2000.

[41] Moritz Kuhn. The Karush-Kuhn-Tucker theorem. *Wireless and mobile communication (electrical and electronic engineering)*, issue: November, pages 1-14, 2006.

[42] P. Li and S. Zdancewic. Downgrading policies and relaxed noninterference. In *Proceedings of the 32nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. ACM SIGPLAN Notices 40(1): 158 - 170. New York, NY, USA, 2005.

[43] G. Lowe. Quantifying Information Flow. *Computer Security Foundations Workshop, IEEE*, pages 18-27, 2002.

[44] P. Malacaria. Assessing security threats of looping constructs. In *POPL '07: Proceedings of the 34th annual ACM symposium on Principles of programming languages*, pages 225–235, Nice, France, 2007.

[45] P. Malacaria and H. Chen. Lagrange multipliers and maximum information leakage in different observational models. In *PLAS '08: Proceedings of the third ACM SIGPLAN workshop on Programming Languages and Analysis for Security*, pages 135-146, Tucson, AZ, USA, 2008.

[46] I. Mastroeni. Abstract Non-Interference: An Abstract Interpretation-based approach to Secure Information Flow. *Ph.D. Thesis*, Department of Informatics, University of Verona, 2005.

[47] S. McCamant and M. D. Ernst. Quantitative Information-Flow Tracking for C and Related Languages. *MIT Computer Science and Artificial Intelligence Laboratory Technical Report*, MIT-CSAIL-TR-2006-076, 2006.

[48] S. McCamant and M. D. Ernst. A simulation-based proof technique for dynamic information flow. In *PLAS '07: Proceedings of the 2007 workshop on Programming languages and analysis for security*, pages 41-46, San Diego, California, USA, 2007.

[49] S. McCamant and M. D. Ernst. Quantitative information flow as network flow capacity. In *PLDI 2008, Proceedings of the ACM SIGPLAN 2008 Conference on Programming Language Design and Implementation*, pages 193-205, Tucson, AZ, USA, 2008.

[50] J. Millen. Security models and information flow. In *Proc. of IEEE Symposium. on Security and Privacy*, pages 180-187, Oakland, Canada, 1990.

[51] J. K. Millen. Covert channel capacity. *IEEE Symposium on Security and Privacy*, pages 60-66, 1987.

[52] A. Miné. A new numerical abstract domain based on difference-bound matrices. In *PADO '01: Proceedings of the Second Symposium on Programs as Data Objects*, pages 155-172, London, UK, 2001.

[53] D. Monniaux. An abstract Monte-Carlo method for the analysis of probabilistic programs (extended abstract). In *28th Symposium on Principles of Programming Languages (POPL '01)*, pages 93-101, 2001.

[54] D. Monniaux. Abstract interpretation of probabilistic semantics. In *SAS '00: Proceedings of the 7th International Symposium on Static Analysis*, LNCS 1824: 322-339, 2000.

[55] D. Monniaux. Backwards abstract interpretation of probabilistic programs. In *European Symposium on Programming Languages and Systems (ESOP '01)*, LNCS 2028: 367-382, 2001.

[56] C. Mu and D. Clark. Quantitative Analysis of Secure Information Flow via Probabilistic Semantics. *International Conference on Availability, Reliability and Security (ARES)*, pages 49-57. IEEE Computer Society, Los Alamitos, CA, USA, 2009.

[57] A. C. Myers, M. R. Clarkson and F. B. Schneider. Belief in Information Flow. In *CSFW '05: Proceedings of the 18th IEEE workshop on Computer Security Foundations*, pages 31 - 45. IEEE Computer Society, Washington, DC, USA, 2005.

[58] A. Myers, A. Sabelfeld and S. Zdancewic. Enforcing robust declassification. In *CSFW '04: Proceedings of the 17th IEEE workshop on Computer Security Foundations, IEEE Computer Society*, page 172, Washington, DC, USA, 2004.

[59] A. Myers, A. Sabelfeld and S. Zdancewic. Enforcing robust declassification and qualified robustness. *Journal of Computer Security, Special issue on CSFW17*, 14(2): 157-196. , Amsterdam, The Netherlands, 2006.

[60] A. D. Pierro and R. D. Pierro and C. Hankin and H. Wiklicky. On approximate non-interference. *Journal of Computer Security*, 12(1): 1-17. IEEE Computer Society Press, 2002.

[61] A. D. Pierro and R. Di Pierro and H. Wiklicky and C. Hankin. Approximate Non-Interference. In *Proceedings of 15th IEEE Computer Security Foundations Workshop*, pages 1-17, IEEE Computer Society Press, 2002.

[62] P. Ryan and J. McLean and J. Millen and V. Gligor. Non-Interference: Who Needs It? In *CSFW '01: Proceedings of the 14th IEEE workshop on Computer Security Foundations, IEEE Computer Society*, pages 237-238, Cape Breton, Novia Scotia, Canada, 2001.

[63] P. Naur. Checking of operand types in algol compilers. *BIT Numerical Mathematics*, 5(3): 151-163, 1965.

[64] A. Sabelfeld and A. C. Myers. Language-Based Information-Flow Security. *IEEE Journal on Selected Areas in Communications, special issue on Formal Methods for Security*, 21(1):519, 2003.

[65] A. Sabelfeld and A. C. Myers. A model for delimited information release. In *Proceedings of International Symposium on Software Security (ISSS'03)*, LNCS 3233: 174-191, 2004.

[66] A. Sabelfeld and D. Sands. A Per model of secure information flow in sequential programs. In *Higher-Order and Symbolic Computation*, 14(1): 59-91, Kluwer Academic Publishers, Hingham, MA, USA, 2001.

[67] C. E. Shannon. A mathematical theory of communication. *ACM Mobile Computing and Communications Review, SPECIAL ISSUE: Special issue dedicated to Claude E. Shannon*, 5(1): 3-55, ACM New York, NY, USA, 2001.

[68] A. Simon and A. King and J. M. Howe. Two variables per linear inequality as an abstract domain. In *Proceedings of the 12th international conference on Logic based program synthesis and transformation.* LNCS 2664: 71-89, 2002.

[69] G. Smith. On the foundations of quantitative information flow. *Proceedings of the 12th International Conference on Foundations of Software Science and Computational Structures: Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009*, LNCS 5504: 288-302, York, UK, 2009.

[70] D. Volpano and G. Smith. Verifying secrets and relative secrecy. In *Annual Symposium on Principles of Programming Languages, Proceedings of the 27th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 268-276. Boston, MA, USA, 2000.

[71] S. Zdancewic. A type system for robust declassification. In *Proceedings of the Nineteenth Conference on the Mathematical Foundations of Programming Semantics. Electronic Notes in Theoretical Computer Science*, 2003.