



UNIVERSIDAD DE BUENOS AIRES
FACULTAD DE CIENCIAS EXACTAS Y NATURALES
DEPARTAMENTO DE COMPUTACIÓN

Studying the notion of Equity in Privacy Conflict Resolution

Thesis presented to request the degree of
Licenciado en Ciencias de la Computación

Mario Julián Sackmann

Advisors: Dr. Guillaume Piolle and Lic. Rodolfo Baader

Coadvisor: Dr. Valérie Viet Triem Tong

Rennes - Buenos Aires, 2015

ESTUDIANDO LA NOCIÓN DE EQUIDAD EN RESOLUCIÓN DE CONFLICTOS DE POLÍTICAS DE PRIVACIDAD

En muchos sistemas los usuarios proveen contenido y lo comparten entre si, indicando sus deseos sobre cómo exactamente quieren que dichos contenidos sean compartidos mediante la expresión de una **política de privacidad**. En el caso en que múltiples usuarios expresen sus políticas sobre el mismo contenido puede ocurrir un **conflicto** si sus deseos difirieran.

En susodichos sistemas existe un mecanismo (llamado **estrategia**) para resolver los conflictos de políticas de privacidad mediante el que se toma una decisión con respecto a cuáles de las políticas respetar y cuáles ignorar. Dependiendo del funcionamiento de esos mecanismos, pueden darse situaciones injustas en las que algunos usuarios ven sus políticas respetadas más frecuentemente que otros.

En este trabajo proponemos formalizar esta clase de situaciones y definir un modelo de sistema basado en la noción de **equidad** con el objetivo de analizar y comparar diferentes estrategias de resolución de conflictos de políticas de privacidad. Para ello usaremos las **redes sociales** como caso de estudio para nuestro modelo y analizaremos las características de equidad de las estrategias más usadas por estos sitios.

Con el objetivo de medir la equidad nuestro modelo usará el **coeficiente de Gini**, una métrica muy utilizada en las ciencias económicas para medir la inequidad. En nuestro trabajo desarrollamos como prueba de concepto una estrategia de resolución que incorpora la noción de equidad y la utilizamos como base de comparación para las estrategias tradicionales y las utilizadas en redes sociales.

Nuestro análisis evidencia una mejora significativa en la equidad de nuestra estrategia de resolución comparada con las otras estrategias. Particularmente, los peores resultados fueron evidenciados por la estrategia utilizada por las redes sociales más populares, permitiéndonos categorizarla como no conforme con la privacidad.

Palabras clave: Privacidad, Equidad, Políticas, Modelo, Gini, Redes Sociales.

STUDYING THE NOTION OF EQUITY IN PRIVACY CONFLICT RESOLUTION

In many systems, users provide content and share it amongst themselves, indicating their wishes on how they want that content to be shared by the means of expressing a **privacy policy**. In the case in which more than one user express their policies over the same content **conflicts** might arise. In such systems a mechanism (called a **strategy**) is in place to resolve said conflicts in order to make a determination as to which policy should be enforced and which policy ignored. Depending on how those mechanisms operate, unfair situations may occur, in which some users see their policy enforced more frequently than others.

In this work we propose to formalize this situation and define a system model based on the notion of **equity** to analyze and compare different strategies. We will use **social networking systems** as a case study for our model and analyze the equity characteristics of conflict resolution strategies currently used in such sites.

To measure equity, our model will use the **Gini coefficient**, a very widely used metric for inequity in the field of economics. In our work we develop an equity-aware conflict resolution strategy to use as a basis for comparison against strategies used by SNSs and other common strategies.

Our analysis evidences that our conflict resolution strategy yields better results in regards to equity than all other resolution strategies. The poorest results were shown by the strategy adopted by most popular SNSs, making it not privacy-aware.

Keywords: Privacy, Equity, Policy, Model, Gini, Social Networking Systems.

To my parents, who always supported me.

To my girlfriend, who always had confidence in me.

To my friends, who helped me get here.

Thank you

CONTENTS

1. Introduction	1
1.1 Sharing in Social Networking Systems	1
1.1.1 Facebook	1
1.2 Several users decide on the same information: a problem	2
1.2.1 Facebook’s conflict resolving strategy	2
1.3 Equity	4
1.4 Contributions	4
1.5 Organization	4
2. Equity	5
2.1 Intuitive approach	5
2.2 Vertical parity vs Horizontal parity	5
2.3 Equity in other domains	6
2.3.1 Economics	7
2.3.2 Law	7
2.3.3 Health Care	7
3. Contribution: a model for analyzing equity	9
3.1 Three magnitudes	9
3.1.1 Box example	10
3.2 Formalizing equity	10
3.2.1 Example	11
4. Our case study: SNS	13
4.1 Preferences, Policies, Settings and Rulings	13
4.1.1 Preference	13
4.1.2 Policy	13
4.1.3 SACLS	14
4.1.4 Rulings	14
4.2 Defining conflicts	14
4.2.1 Conflict in SNS	15
4.3 Strategies	16
4.3.1 Composition	17
4.3.2 Static strategies	18
4.3.3 Majority strategy	19
4.3.4 Topology strategies	21
5. Modeling our system for equity analysis	23
5.1 The system as an automaton	23
5.1.1 Underlying graph	23
5.1.2 Equity in the states	24
5.1.3 The transition function	24
5.2 Equity for the system	24

5.2.1	Algorithms	27
5.2.2	Measuring equity - the Gini coefficient	31
5.3	Model for Social Networking Systems	32
5.3.1	Automaton representation for SNS	33
6.	The simulator	39
6.1	Programmed strategies	39
6.1.1	EIS Strategy	40
6.2	Experimentations and Results	44
6.2.1	Random	44
6.2.2	Worst case scenario	46
7.	Conclusion	49
7.1	Conclusions and future work	49

1. INTRODUCTION

1.1 Sharing in Social Networking Systems

In the last years, the world has seen substantial improvements of the technologies used to build websites (both in quality and quantity), making them much more interactive, responsive and reliable. The term *Web 2.0*, which illustrates this, was coined by O'Reilly in late 2004 [1]. This, accompanied by a massification of the internet access in all social classes [2], has resulted in the notorious upsurge of platforms whose main purpose is to allow users to interact by sharing interests, activities, opinions, pictures, videos, etc. These platforms are called **SNSs** (*Social Networking Systems*) and are described by Boyd and Ellison as “*web-based services that allow individuals to construct a public or semi-public profile within a bounded system, articulate a list of other users with whom they share a connection, and view and traverse their list of connections and those made by others within the system*” [3].

Throughout the introduction we will use Facebook as an example in order to illustrate the concepts we will be presenting.

1.1.1 Facebook

With over 1.23 billion¹ users, Facebook is the largest **SNS** in existence. It provides a complex means to interact with other users, via sharing documents, *tagging* them, pictures, videos, etc.

Facebook has one of the most comprehensive and detailed user-driven privacy settings of all **SNS** analyzed. Nearly every aspect of one's account can be configured with a specific privacy setting.

Facebook's general privacy settings works by specifying with whom a particular **type** of information is to be shared. For example, a user may specify that, by default, all his uploaded photo albums should be visible to:

- Only himself.
- His friends.
- His friends of friends.
- Everyone on Facebook (public).

It is also possible for a user to override his own default policies for a specific piece of data to be shared. For instance, a users' policy might determine that his photo albums be shared with no one but still determine that a particular album be shared with all his friends.

One problem that arises from this is that the sheer volume of customizable settings can be overwhelming, and most users don't bother to read and update them, therefore not specifying any customized privacy policy. Facebook has over 170 different privacy settings a user may use to enforce his policy. An interesting study done by Liu and

¹ We consider the American billion. 1 billion = 1.000.000.000

Krishnamurthy [4] states that the way users have their privacy settings configured in Facebook only matches their expectations around 37% of the time. Almost all of the other 63% of the time that people’s settings did not properly reflect their policy, the content was more exposed than users expected.

1.2 Several users decide on the same information: a problem

In an SNS, users interact and share content amongst themselves. Via the means of *tagging*, a user states that “this user is related to this content”, explicitly establishing a link between content and user.

The restricted access theory of informational privacy sees privacy achieved in a system if one is able to limit and restrict others from access to personal information [5]. To that end, SNSs need to provide some access control mechanism so that users can take a certain determination regarding that information, such as sharing, deleting, *tagging* other users, etc.

In most SNSs the users’ ability to express their wishes in regard to their information is reduced because the system only takes into consideration the decision of the user who provided the information. We consider this approach completely unacceptable because it reduces the **users’ control over their own information**. In our present work we will consider systems in which all users linked to a certain piece of information are entitled to a form of control over it.

However, a scenario might rise in such a system in which a single piece of content relates to several users. In such a scenario, users would not agree on the uses of a certain piece of information (within a context), leading to a **conflict**. For example two users could disagree if a certain information is to be shared with a third party or not. The system then must make a determination as to whether it shares the information or not.

Generally, upon making a determination the system is respecting the wishes expressed by some individuals and disregarding others. Considering a global view, with multiple interactions, depending on how the system makes the aforementioned determination, this might lead to situations in which certain individuals consistently see their wishes ignored.

1.2.1 Facebook’s conflict resolving strategy

Facebook has a very complex strategy system regarding privacy conflicts.

For example, when adding data to the service in the form of a “post”, the user can specifically select amongst his friends who will be able to see it and who won’t by putting them into lists.

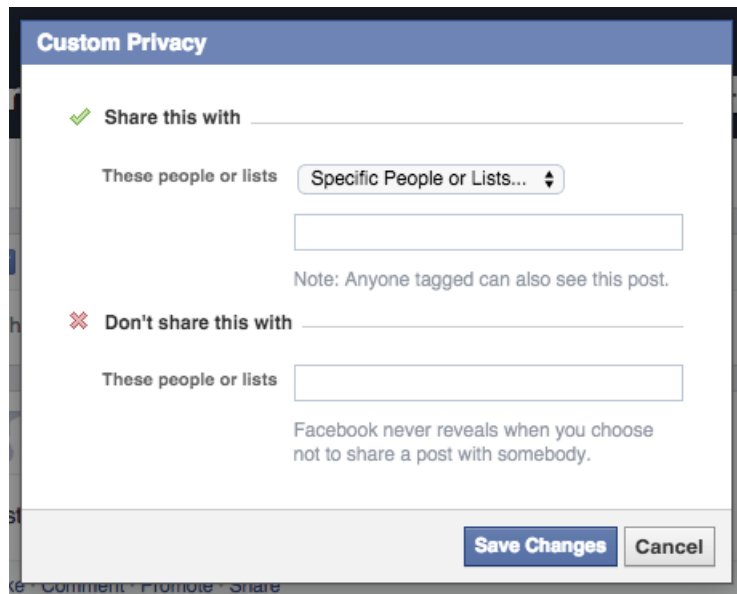


Fig. 1.1: Facebook's sharing setting

Tagging in Facebook can't constitute a conflicting operation for user-defined privacy policies in friends lists, because **only the photo's² owner's settings are taken into account**. Tagged users cannot determine with whom this picture is shared: their only allowed action is to remove their tag from the photo (or require the owner to remove the picture entirely). This concept is part of what has been defined by the Trusted Computer System Evaluation Criteria as *Discretionary access control* (DAC)[6].

However, if a given user is tagged in a photo, regardless of whether he is in the "don't share this with" list, he will be able to see it. Therefore, if we consider an implicit policy that states that a user wants to see every place he is tagged in, a conflict exists between this implicit policy and the photo owner's policy. Facebook resolves this with a permissive strategy, letting the tagged user see the picture.

Other Software

There is a lot of other software that has to deal with information rights. Most of this software considers, like Facebook, that a piece of information has one owner and he is the one that determines how that information is used. In essence, only his privacy policy is taken in to account.

However, most of this software also has a special user (or role) in the system for a user with "special" privileges that can control any piece of information as if it were his own. In other words, he can *take ownership* of any piece of information in the system. This can be seen in most operating systems (for example, the *root* user in UNIX based systems or the *Administrator* role in Windows) and also on most forum boards like PhpBB or vBulletin.

Another technique currently being used to determine the visibility of a given piece of information not by the owner or the *tagged* users but on **where** the user has shared the

² Also applies to tags in comments and posts.

information. This is the concept applied by most forums (vBulletin, PhpBB, IPB, Vanilla Forums, etc).

The owner usually does have some privileges over the file like edition or deletion of the information, but not who has access to it. The premise is that if he shared that information in a given space, he intended to share it with everyone who can access this space. The aforementioned *special user* determines the spaces, their users and visibilities.

1.3 Equity

In this work we propose to explore and characterize the unfairness in which some users see their policies enforced more than others (described in section 1.2) by developing the notion of **equity** in systems.

Equity is an important notion because it evolves on the concept of fairness, considering the inherent differences in each party. This relates to the possibility that every party should have to affect their environment in the same amount.

In fields such as health care, education, finance, law, amongst others, equity has been properly investigated and developed. However, in computer science there have been very few works regarding on the notion of equity, and even fewer that approach it in a formal way, despite the aforementioned upsurge of SNSs, cloud based document sharing and other applications in which users express heterogeneous decisions regarding sharing.

1.4 Contributions

In this paper we propose to formally analyze the equity characteristics of conflict situations, the different techniques the system might use to resolve said conflict and their impact in equity. To this end, we will need a formalization of the notion of equity and a model of the system tailor-made to equity-analysis.

This model will help us transform our intuition about how certain strategies impact equity into formal analysis with metrics that can be will be used to establish optimization goals. Based on our formal model and its metrics we intend to build a simulator that will allow us to perform systematic comparisons between strategies.

To the best of our knowledge, there is no other work that presents a formal model for the study of equity in privacy conflict resolution. The work presented by Marin et al [7] disserts on similar subjects but presents a more system-wide view of the problem.

1.5 Organization

The remainder of this work is organized as follows: in chapter 2 we will discuss the notion of equity from an informal point of view, motivating the formalization and metrics presented in chapter 3. In chapter 4 we will discuss SNSs and present the classical strategies for solving conflicts between user defined privacy policies. In chapter 5 we will build on the notions presented in the previous chapters and introduce a system model suitable for equity analysis. In section 6 we will present the simulator that follows the aforementioned system model, with the experiments designed to test different strategies and their results. Finally, in chapter 7 we will present the conclusions and future work.

2. EQUITY

In this chapter we are going to talk in a general sense of **equity** from an intuitive and human approach. Its ideas, notions and facets. We will also do a very succinct exposition on what other different branches of the human knowledge consider equity, their similarities and differences.

This chapter is intended to present a broad view on the subject, but its contents are important to understand some of the motivations and decisions of the subsequent work.

2.1 Intuitive approach

The word **equity** has its roots in the Latin word *aequitas* which stood for the roman concept of justice, equality, symmetry and fairness [8, p. 49]. Today, the word equity has many specific meanings in different areas of knowledge, but most of them still share this Roman notion.

Building on this concept, the idea of equity stands on the goal of obtaining a situation that is uniform for every party involved without the assumption that every party starts in identical predicament. This is what distinguishes it from the notion of *equality*, in which each party is provided the same amount, regardless of its initial condition.

2.2 Vertical parity vs Horizontal parity

In a general and intuitive sense the notion of equity stands for all parties being at the same level, as opposed to the notion of **equality**, that is related to everyone having the same amount.

Existing informal literature on the subject (most notably J.Y. Duclos in [9]) disserts on two aspects of the notion of equity, which we will adopt for the context of this work:

- **Horizontal parity:** parties who already have the same amount should receive the same amount. This is intended to avoid unjust treatment of equal parties.

In our work we shall use a more general approach to horizontal parity, in which we don't care that both parties receive the same amount, but actually we want to ensure that they get *some* amount (possibly none) such that the resulting situation is equitable¹.

- **Vertical disparity:** parties with differences should receive different treatment. This is intended to restore the balance to an already unlevelled situation.

This is a way of expressing the notion of “leveling the playing field”, as every party should end up *in the same condition*, even if it implies giving a *different amount* to each party. This notions of “condition” and “amount” are left vague on purpose for the time being since they are extremely case-specific. We will be illustrating in the following sections.

¹ This subsums the previous definition. If we assume a monotonic evaluation function, both definitions are true. We adopt the second definition to be able to assume nothing about such monotonicity.

This distinction between the current status of each party is probably the most important facet of the whole concept of equity.

Let us present an example in order to illustrate the difference between equity and equality. In the following image we can see said difference in a graphical way.

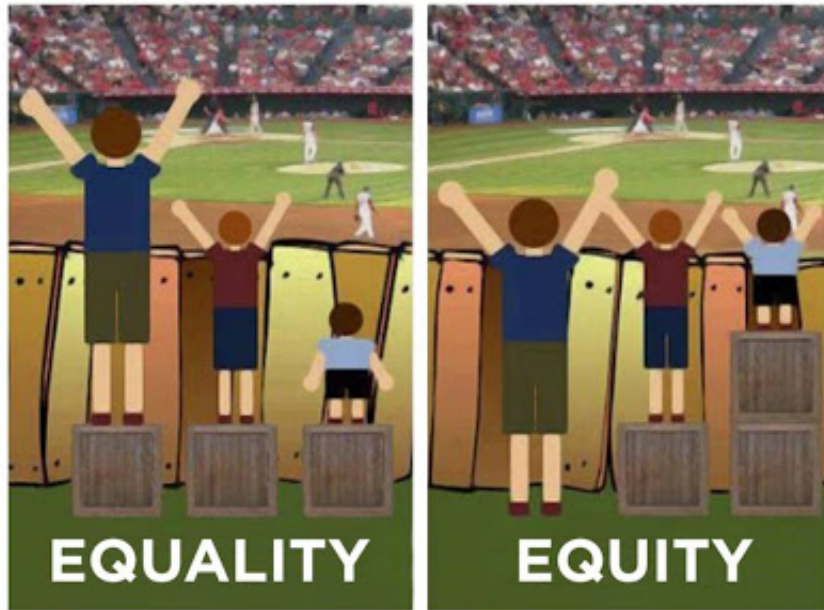


Fig. 2.1: Boxes example

Lets consider the case of example of figure 2.1 [10]. The situation in the left is equal (ie has the property of *equality*) because the goods (represented by the boxes) are distributed evenly amongst all parties: each person has one box. However, even though we have not formally defined equity yet, we can intuitively see that there is a certain unfairness to the situation, since the person in the left could see over the fence without a box, whilst the rightmost one needs two boxes. So giving one box to each person does not “level the playing field”.

On the other hand, on the situation on the left, even though the goods are not distributed evenly, every person can see over the fence, effectively leaving them in a more similar situation.

2.3 Equity in other domains

In this section we will provide some examples on different domains and how they define equity.

2.3.1 Economics

In the field of economics, the notion of equity relates to the idea that people with similar income should pay similar taxes. As in the general case, the two facets of equity determine this idea:

- Horizontal parity: all people who make/have the same amount of money should pay the same taxes.
- Vertical disparity: people who make/have more money should pay more taxes.

These two aspects of equity achieve the desirable objectives of *tax neutrality* and *wealth redistribution* [11], respectively.

2.3.2 Law

Law regards the notion of equity in a slightly different manner. It is seen as the application of the justice to the particular case for which the generality of law leaves a loophole or ambiguity, or some change in culture or habits has left the law obsolete. The intention of this application is to “level the playing field” when either the law or the circumstances benefit one party over another.

An example of this can be seen in the Argentinian law, in which some old precepts are written in a way that treats differently men and women: if a man works in a public office and has to travel, under certain circumstances he can request a ticket for **his wife** to accompany him.

This precept was written in a different time, in which women working in public office and same-sex marriage² were not conceivable. However, if interpreted directly, this precept does not allow the husband of a public officer to request a ticket. This does not have the intention of imposing gender or sexual orientation discrimination, therefore the principle of equity is applied to allow husbands to ask for tickets too.[12]

2.3.3 Health Care

The notion of equity is also present in the regulation of health care systems. In this context its application is intended to guarantee the proper health care of the population within the scope of the regulation.

In this field, the magnitude we intend to level is the severity of the patient’s ailments, that the health care system can affect it by providing treatment.

- Horizontal parity: stands for an equal access to medical services to all individuals: people who have similar ailments should expect similar conditions in treatment. For example, two people with a broken arm should expect similar delays in getting an x-ray.
- Vertical disparity: the discrimination in medical treatment should only be done according to the severity of the patient’s condition or injury. This is intended so that all people quickly regain good health [13]. For example, if a patient has a twisted ankle and a car crash victim comes in with broken ribs, he should have priority in getting his x-rays.

² Now legal in Argentina.

Health care is still one of the fields for which equity is most far from being achieved [14, 15]. There are still numerous disparities regarding ethnicity, income, sexual orientation, social class, etc.

In the following chapters we intend to analyze the different conflict-resolving strategies. To that end, we build on the view of equity presented here, formalizing and analyzing it and, ultimately, build a formal model of the aspects of equity in systems in which there are policy conflicts.

3. CONTRIBUTION: A MODEL FOR ANALYZING EQUITY

In this chapter we will present our proposed model for equity, with its corresponding formalism and a brief but illustrating example.

Our model, though simple, allows for a quantification of equity, which will act as grounds for the subsequent goal of comparing existing conflict resolution strategies. This is a necessary intermediate step because, as mentioned in the introduction (section 1.4), existing literature does not provide a comparable model to analyze the particular aspect of equity in conflict scenarios.

3.1 Three magnitudes

In order to structure and formalize the analysis of equity we have conceived three separate but closely related magnitudes. These magnitudes will be the basis for our analysis as their values shall determine the status of a given *situation* in relation to equity.

The aforementioned *situation* we consider is a scenario involving several parties and an entity that can affect some part of it. This entity may or may not have as an objective to balance the situation, or to restore equity, but through its actions it will affect it.

For the context of our study, this entity will always be the system (and we shall therefore employ that name from this point forward). However, this is not a restriction imposed by the model, since it can be applied to other situations in which the entity might be different. For example, in a situation in which a parent is distributing money amongst his children, the entity that evaluates the situation and tries to restore equity is the parent.

The three magnitudes we will use to model equity are:

- **Wealth.** It represents the intrinsic state of each party in the situation, without the system's compensation. It affects the measurement the system will use to measure equity and it will be part of the determination of how much each party should be compensated. It is important to note that the system cannot affect this magnitude: wealth is inherent to each party.
- **Requital.** It is the magnitude that the system can assign or distribute amongst the parties to help restore equity in the case of an imbalance.
- **Mensuration.** It is the magnitude the system uses to measure equity. It is a function \mathfrak{M} of *wealth* and *requital* that determines the state of the situation considering the system's compensation.

There is no requirement that these magnitudes be of the same nature. They can be different. In the latter case however, they must be related in some fashion if the system is to be able to affect equity: if the system measures equity by some magnitude, but it can't affect it, it will not be able to restore equity if it were to get unbalanced. The boxes example from section 2.1 works to illustrate this point: *number of boxes* and *height* are not measurements of the same nature, but they are related. We will detail this in the following section.

It is also important to notice that *wealth* is not always present on systems. There are systems in which the mensuration is entirely artificial (i.e., created by the system), and no part of it is inherent to the party. We will see such an example in section 5.3.1.

3.1.1 Box example

Let us reconsider the situations in the figure 2.1, from the second chapter (section 2.1), following the magnitudes we have just informally defined.

In that scenario we consider each person as a party. The *wealth* magnitude is the person's height and the *requital* magnitude are the boxes that are to be distributed amongst the persons. The *mensuration* magnitude (for which we will use the letter \mathfrak{M}) is the height at which the head of each person ends up. In this case, the mensuration can be computed as the sum of the person's height and the size of all the boxes on which he is standing.

$$\mathfrak{M}(\text{height}, \text{boxes}) = \text{height} + \text{count}(\text{boxes}) \cdot \text{boxes.edge}$$

We can also observe that the magnitudes are related because each box that a person gets adds to the height where his head is. Therefore assigning more boxes to a person increases the height of his head. This is an indirect way in which the system (in this case, whoever assigns the boxes) can affect the *mensuration* and restore equity from the situation on the left (which is equal, but not equitable) to the situation on the right.

3.2 Formalizing equity

The two facets of equity introduced in section 2.1 can be formalized using the magnitudes we defined in section 3.1 in the following way:

Let p_1, p_2, \dots, p_n be the different parties in situation q , with:

- R_i^q is the *requital* of party i in situation q
- W_i is the *wealth* of party i

Let $\mathfrak{M}(R_i^q, W_i)$ be the magnitude *mensuration* for party i in q . $\mathfrak{M}(R_i^q, W_i)$ is a value that the system must compute in order to determine the equity status of situation q . It is specific to each system.

Formally, we define the situation q as **equitable** if and only if

$$\boxed{\forall i, j \in [1, \dots, n] : \mathfrak{M}(R_i^q, W_i) = \mathfrak{M}(R_j^q, W_j)} \quad (3.1)$$

This allows to represent the facets of *vertical* and *horizontal* parity by controlling R_i^q to level $\mathfrak{M}(R_i^q, W_i)$. If two parties have the same *wealth*, then by assigning them *requital*, the system can compensate and have them end up with the same *mensuration*. In particular, if we assume the measurement we are using is a monotonically increasing function, then this implies that both of them should receive the same amount, which is the first definition of *horizontal parity* by Duclos et al [9].

Let's suppose that for two parties i and j in situation q , it is true that $W_i = W_j$. Therefore assuming \mathfrak{M} monotonically increasing,

$$\mathfrak{M}(R_i^q, W_i) = \mathfrak{M}(R_j^q, W_j) \Leftrightarrow R_i^q = R_j^q$$

It is worth observing that many systems have a staggered consideration of equity: a situation is considered equitable if the *mensuration* for each party are within a certain range, as opposed to being strictly equal. This is not specified in this formalism because it can be embedded in the function \mathfrak{M} . We will see an example of such behavior in section 5.3.1.

3.2.1 Example

We will reuse our box example and metrics from section 3.1.1. Assuming each box has a height of 1 and the heights of the three individuals are: 3, 2 and 1 respectively (from left to right), we can use formula 3.2 to determine the equity of each of the situations.

On the left we can see that the situation is not equitable because:

- $\mathfrak{M}(R_1^{left}, W_1) = height + count(boxes) \cdot boxes.edge = 3 + 1 * 1 = 4$
- $\mathfrak{M}(R_2^{left}, W_2) = height + count(boxes) \cdot boxes.edge = 2 + 1 * 1 = 3$
- $\mathfrak{M}(R_3^{left}, W_3) = height + count(boxes) \cdot boxes.edge = 1 + 1 * 1 = 2$

However, the situation on the right:

- $\mathfrak{M}(R_1^{right}, W_1) = height + count(boxes) \cdot boxes.edge = 3 + 0 * 1 = 3$
- $\mathfrak{M}(R_2^{right}, W_2) = height + count(boxes) \cdot boxes.edge = 2 + 1 * 1 = 3$
- $\mathfrak{M}(R_3^{right}, W_3) = height + count(boxes) \cdot boxes.edge = 1 + 2 * 1 = 3$

As we can see in this very simple example, our formalization allows us to effectively determine whether a given situation is equitable. This shall prove useful in our attempt to analyze equity in several conflict resolution strategies, which we shall do in the following sections.

4. OUR CASE STUDY: SNS

In this chapter we will focus the model and formalism presented so far onto one particular case study: *Social Networking Systems*. As mentioned in the introduction (section 1.1), there have been a notorious increase in the use of SNSs in the latest years. This exacerbates the importance of how fairly this platforms are to their users, as it targets a very large proportion of users who use the system.

We will start dissertating about a typical SNS, its components, typical behaviour, etc. We will define several of their aspects and provide examples of typical Social Networking Systems and how they handle/avoid user-defined policy's conflicts. Afterwards we will present different strategies to resolve conflicts, with a very succinct analysis comparing their respective advantages and disadvantages.

4.1 Preferences, Policies, Settings and Rulings

In our attempt to study conflict and resolution strategies in SNSs, we were hindered by the lack of consistency in the nomenclature of existing literature. Therefore, we will devote this section to discussing four different but very closely related concepts whose understanding is important as they motivate decisions for the following chapters.

4.1.1 Preference

The Oxford English Dictionary defines a **preference** as *a greater liking for one alternative over another or others*. Circumscribed to the present work, we will use the term **user preferences** to refer to the abstract notion of what the user's desires are for sharing his private information. This is an abstract process that takes place inside a user's mind to evaluate whether he wants to authorize or not access to a certain picture to a certain individual and under which circumstances.

In a more colloquial way, we shall use the term "preferences" for the decision at a human level.

4.1.2 Policy

Since user preferences happen in the boundaries of the user's mind, systems require some way of expressing those preferences in a way that allows users to store and operate them. When those preferences are expressed in a formal way they are called **policies**.

Several formalisms have been devised to express user preferences in systems. For instance EPAL (Enterprise Privacy Authorization Language) is "a formal language for writing enterprise privacy policies to govern data handling practices in IT systems according to fine-grained positive and negative authorization rights"[16]. Other examples are XACML (eXtensible Access Control Markup Language) and P3P (Platform for Privacy Preferences Project).

The aforementioned formalisms have several advantages such as being very expressive, versatile and most have already some efficient means of solving conflicts.

However, most social network sites do not utilize such complex means for expressing privacy policies. Their main disadvantage (and the main reason we suspect SNSs do not

bother with them) is that in those formalisms, policies are extremely complex to write. In a scenario in which millions of end users have to express their respective policies, this kind of complexity makes its application unfeasible since it would overwhelm most of the users.

4.1.3 SACLS

Most SNSs allow users to express their privacy policies via **Social Access Control Lists** (SACLS) [17]. SACLS and other settings can still be quite complex (as seen in section 1.1.1), but they are much more manageable than any of the aforementioned formalisms, enough for end-users to comprehend and use.

However, SACLS are a serious drawback in expressivity, allowing users only to express the most basic policies. Such limited settings usually only allow users to express policies in terms of sets of users who can and can't access this information. In a way, this formalism, that most SNSs provide to express policies, is nothing more than predefined rulings grouped into sets of people (e.g. "for this picture, my ruling will be *disallow* for this group of people").

The main problem with this approach is that it does not allow additional parameters to be inputted into the user's policy. For instance, a feasible scenario would be a policy stating: *I want all users to be able to access this picture if they are doing so from France*. This is a usual scenario with picture-sharing because most countries have different legislations with entitlement and copyright. It is not possible to apply this policy with the extreme expressivity limitations the SALCs formalizations have in existing SNSs.

4.1.4 Rulings

It is important not to confuse the process with the end result. When a user's privacy policy gets evaluated with certain parameters it returns a **ruling**: a pronouncement regarding the authorization of a certain request.

In its most general sense, we define a *ruling space*, that shall include all the possible determinations obtained when a user's privacy policy gets evaluated. In most cases, this space contains two or three rulings (*allow* and *deny*, with an optional *do not care*). However, this is a practical limitation and not a theoretical one, as there is no formal restriction on this subject.

For the context of this work we will consider that a ruling does not impose jurisprudence, or intends any other determination to the particular request that originated the response. The scope of a ruling is specific to that particular request. Another possible approach to this, that we will not pursue in this work is to consider that a ruling sets precedence and any future request consisting of the same involved parties (in the exact same role) must have the same ruling.

4.2 Defining conflicts

In an informal way, we state that a **conflict** arises when two or more users *disagree* on the sharing of a certain information *that involves them*.

This informal definition presents us with two aspects of conflict: defining *involvement*, and *disagreement*. We will abstract ourselves from the issue of entitlement for the time being. As it is very system and context dependent, we will abstract this by assuming that

each piece of information has a given set of users who should decide upon its sharing. Exactly how that set of users is determined is a problem we will leave unattended since we consider that this exceeds the scope of this work.

To formally construct the notion of **conflict** we need to define a nomenclature.

- U is the set of all users in the system;
- P is the set of all pieces of protected¹ information in the system;
- T_p is the set of all users whose privacy policy involves the information p ;
- R is the ruling space;
- $\mathcal{R}(a, c, p, ts) = r$ is a function that determines the ruling ($r \in R$) determined by user $a \in U$ when user $c \in U$ requests access to piece of information $p \in P$, with timestamp ts .

Using this nomenclature and the formalism provided in the previous section, we can formally define a conflict as follows:

There exists a conflict when

$$\boxed{\exists p \in P; \exists a, b \in T_p; \exists c \in U : \mathcal{R}(a, c, p, ts) \neq \mathcal{R}(b, c, p, ts)} \quad (4.1)$$

This definition will serve as our basis for our future analysis and simulations of conflict resolution strategies.

4.2.1 Conflict in SNS

From this point forward we will abandon our general case definition and work with the specific case of conflict in SNSs, as defined in section 1.1 and by Boyd and Ellison in [3].

For this scenario, we will consider **pictures** as pieces of protected information and we propose that only the users tagged in the picture can determine with whom the aforementioned picture is to be shared. Since we can abstract a tag to be a link between a picture and a user, we consider that a user is automatically tagged to any pictures he uploads and therefore that it is within the reach of his privacy policy.

Because of the assumptions mentioned in 4.2, we will work under the simplification that any user tagged in a given piece of information (pictures for this study scenario) has equal rights over its sharing.

Our ruling space will consist of:

- *Allow*
- *Deny*

Concrete example with conflict

In this section we shall provide a concrete example of a hypothetical SNS as described in 4.2

Lets suppose the following social network:

¹ i.e. information that is reached by at least one user preference.

- $U = \{alice, charlie, daniel\}$
- $P = \{p\}$
- $T_p = \{alice, charlie\}$

And the following rulings:

- $\mathcal{R}(alice, daniel, p, ts) = deny$
- $\mathcal{R}(charlie, daniel, p, ts) = allow$

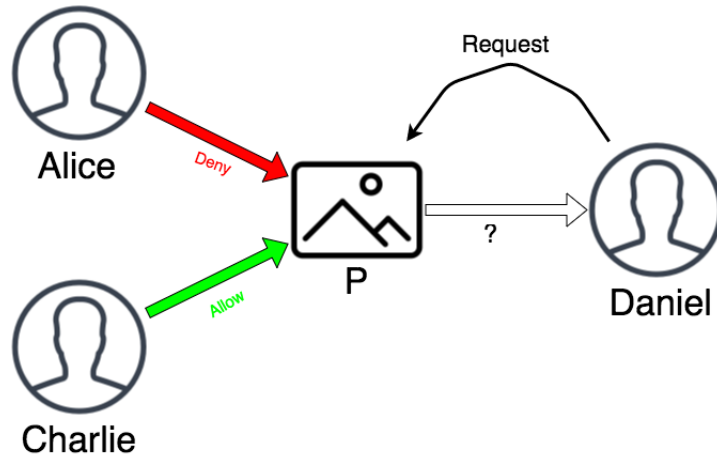


Fig. 4.1: Conflict example

Informally, there is a conflict because user *alice* does not want user *daniel* to see the picture but user *charlie* does. Formally, we can see here that there is a conflict because all of the following conditions are true simultaneously:

- $alice \in U$
- $charlie \in U$
- $daniel \in U$
- $p \in P$
- $(p, alice) \in T$
- $(p, charlie) \in T$
- $daniel \in A_c^p$
- $daniel \in D_a^p$

4.3 Strategies

When a system encounters a situation with **conflict**, it must make a determination about whether it allows or denies the request (or, more generally, it must choose a ruling $r \in R$ to enforce). A **strategy** is the means by which the system makes the determination. In this section we will present different strategies that a system might employ to resolve a conflicting situation.

To formalize each strategy, we will present its **composition function**.

4.3.1 Composition

In its most general sense, we define a **composition** as the process of combining two or more elements to produce a final element. We will refer to the abstraction of this process as **composition function**.

There are countless examples of compositions. For instance the mathematical function composition operator (\circ), can take two (or more) functions f and g and produce a third function $(f \circ g)(x) = f(g(x))$. The way in which the elements are combined is completely arbitrary, and it defines the composition.

In the case of the current work we are interested to study the combination of user policies in order to ultimately produce a determination for a request. There are two possible paths to follow down this road; **policy composition** and **ruling composition**.

Policy composition

One option is to determine a composition function that works by combining all the policies involving a given piece of information and constructing a new policy which will be evaluated for all access requests to that information.

The policy composition approach also has advantages: not only it provides more flexibility and information to work with, but also in the use context of the SNS it allows for the creation of a policy *for the given piece of information* (i.e. the result of the composition of the policies in scope). This approach could avoid the cost of having to compute the composition of all the involved policies for all involved parties for every single request (depending on the system design). This operation can be particularly costly considering that there is no requirement for the policies to be centralized, and the corresponding cost of fetching them could be prohibitive. The fetching and computation only needs to take place when something changes related to the involved users (e.g. some involved user changes its policy or a new user is registered as “involved” for that piece of information).

Policy composition also has a few drawbacks, as having to have a centralized entity that will have to receive and combine users’ policies, thus not being able to keep them truly private (at least trivially).

In fact, for some of the formalisms discussed in section 4.1.2 there already exist solutions for policy composition and ordering based on different premises, preferences, etc. However since SNSs do not typically employ such formalisms we will circumscribe the formalism, composition discussion to the composition of settings.

Ruling composition

Another option is to have a distributed policy system, in which each user applies his policy to his own accord and determines a ruling. After that, a centralized entity collects all involved rulings and composes them to obtain a final ruling.

This approach was taken by Marin et al [7] and has several advantages such as the possibility to keep the policies private and to use simpler algorithms. Also, in a truly decentralized environment, it could allow each user to choose whatever formalism he wants to express his policy, provided that the parameters and end result comply with a determinate signature.

On the other hand, because of the limited information provided by the rulings, this approach is restrictive on the possibilities for the composition function. This is not possible

to work with the users' policies to obtain a finer-grained determinism. Colloquially, it does not allow us to work with the reasons behind the ruling.

One way to alleviate this problem is to add meta-input to the composition process. That is, not only make a determination based on the rulings, but add information such as the user who issued that ruling, the SNS's social graph, etc. It is dependent on the composition function chosen.

Formally, we will define the ruling composition function that takes all the user's rulings and returns a final ruling. The signature for this function will be:

$$\mathcal{C} : \{\mathcal{R}(i, c, p, ts)\}_{i \in T_p} \rightarrow R \quad (4.2)$$

Because of its advantages, this is the composition alternative that we will use to present the different conflict resolution strategies.

4.3.2 Static strategies

Static strategies provide the simplest way to resolve user defined privacy policy conflicts. Static strategies always resolve conflict by defaulting to a predetermined option.

Informally, if all tagged users' rulings agree on the outcome of a given request, then that common agreement is enforced. Otherwise, the system's default option will be enforced.

In general, where the ruling space is bounded to two different options (*allow* and *deny*), there exist two static strategies, based on which option is the default.

Allow strategy

Informally, the system follows the **allow** strategy if it only denies access if all tagged users' rulings agree to deny access.

Formally, the result from the composition of rulings for user u requesting access to picture p is:

$$\mathcal{C}(\{\mathcal{R}(i, c, p, ts)\}_{i \in T_p}) = \begin{cases} deny & \text{if } \forall i \in T_p : \mathcal{R}(i, c, p, ts) = deny \\ allow & \text{otherwise} \end{cases}$$

Deny strategy

Informally, the system follows the **deny** strategy if it only allows access if all tagged users' rulings agree to allow access.

Again, formally, the resulting ruling for user u trying to access picture p will be:

$$\mathcal{C}(\{\mathcal{R}(i, c, p, ts)\}_{i \in T_p}) = \begin{cases} allow & \text{if } \forall i \in T_p : \mathcal{R}(i, c, p, ts) = allow \\ deny & \text{otherwise} \end{cases}$$

Pros and Cons

Static strategies are widely used because of their simplicity both for implementation and end-user understanding. This is a major advantage for strategies in real world scenarios because its simplicity redounds in a very high impact on the SNS's user-experience.

However, one must consider that such extreme decisions pose problems with the usability-security relationship. Most times this concepts pull in opposite directions: increasing the security of a system tends to hinder its usability [18].

In the case of the deny strategy, even though it improves privacy by obtaining less unwanted accesses than any other strategy (zero in fact), it negatively affects the usability of the system as users have less content to see. This may ultimately affect the finances of the company in charge of the SNS [19], which is why most commercial products do not use it.

On the other hand the allow strategy may greatly benefit the usability and user sharing (with corresponding finances), but it is a very poor choice regarding privacy since it is the strategy that maximizes the number of unwanted accesses.

Equity

Regarding the notion of equity that was preliminary discussed in section 1.2, systems that implement static strategies are the ones that present the most inequitable scenarios. It is trivial to construct an example in which one user's policy gets constantly violated.

To construct such a scenario, we suppose a system in which there are a series uploaded pictures (p_1, p_2, \dots, p_n) with 2 tagged users *Alice* and *Bob*. User *Charles*, friend of both of them, requests access to all those pictures. We suppose that *Alice's* ruling is always to authorize and *Bob's* ruling is to deny the request.

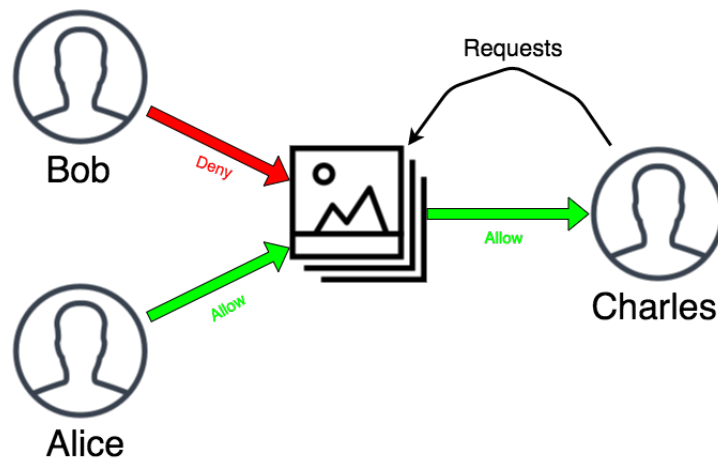


Fig. 4.2: Inequitable situation with allow strategy

If the system follows an allow strategy, it can be deemed inequitable because *Alice's* and *Daniel's* policies will get constantly enforced, while user *Bob* will always see his policy disregarded.

An analogous problem exists if the system follows a deny strategy.

4.3.3 Majority strategy

Majority strategies are an alternative to the static strategies in which the system enforces a ruling that will satisfy the majority of the tagged users' privacy policies.

Formally, the enforced ruling for user u trying to access picture p will be:

$$\mathcal{C}(\{\mathcal{R}(i, c, p, ts)\}_{i \in T_a}) = \begin{cases} allow & \text{if } \#\{\mathcal{R}(i, c, p, ts) = allow\}_{i \in T_p} > \#\{\mathcal{R}(i, c, p, ts) = deny\}_{i \in T_p} \\ deny & \text{if } \#\{\mathcal{R}(i, c, p, ts) = allow\}_{i \in T_p} < \#\{\mathcal{R}(i, c, p, ts) = deny\}_{i \in T_p} \end{cases} \quad (4.3)$$

If s is a set, the symbol $\#s$ denotes the cardinality of set s , ie. the number of elements that s contains.

It is necessary to note that equation 4.3 does not cover all cases. When the two sets have the same cardinal, one of the static strategies must be put in place to resolve the conflict. In colloquial terms, if there is a tie, the system must still take a decision.

Pros and Cons

Majority strategies share the main advantages of static strategies (simplicity to implement and understand). They also improve on the number of unwanted accesses and equity [7].

However, they improve over static strategies for **some** users in the system: the users whose privacy policy is most popular. Majority strategy is the worst possible strategy for a user whose privacy policy is the least popular for every request made. This is not a mere theoretical concern since a case like that can happen.

Equity

Equity-wise, majority strategies generally improve over static strategies since there is no possibility for a minority of users to impose their rulings over a majority of users. However, there is still the possibility that a small number of users (strictly less than half the users in the system) never see their policies enforced. This is an inequitable situation for those users.

To construct an example of such a scenario, we will expand the example presented in section 4.3.2 by adding a new user: *Daniel*, whose ruling, as *Alice*'s is to allow the requests.

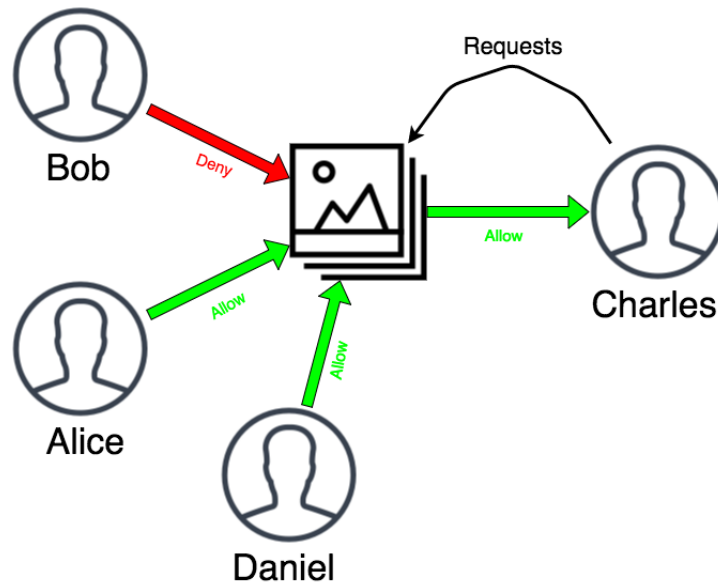


Fig. 4.3: Inequitable situation with majority strategy

If the system follows a majority strategy, it can be deemed inequitable because *Alice's* will get constantly enforced, while *Bob's* will not.

4.3.4 Topology strategies

Topology strategies take into consideration the topology of the social graph in order to make a determination regarding a given request.

One possible example of a topology strategy would be to assign a certain weight to the ruling of a user based on a distance in the social graph, so the ruling of a tagged user will not affect so heavily people who are far away. It is worth noting that this example, in the case of a complete social graph, would be equivalent to a majority strategy.

Another possibility is to expand the previous example by weighting more heavily for denials those people who are farther away; and for approvals the opposite (closer to the tagged user implies heavier weight). This is based on the assumption that people usually like to share their pictures with their friends, whereas they do not want strangers seeing them.

It is important to notice that this kind of strategy cannot be applied with the signature of the composition function \mathcal{C} previously defined, since this function does not take as parameter the graph topology. Let G be the social graph. \mathcal{C} 's signature would have to be changed to

$$\boxed{\mathcal{C}(\{\mathcal{R}(i, c, p, ts)\}_{i \in T_p}, G)} \quad (4.4)$$

Pros and Cons

The main disadvantage of the topology strategies is that its implementation can be more costly, both in difficulty and computational complexity. Also it is not as easy to understand for the end users as the static or majority strategies.

However, the topology strategies more accurately represent the way most humans behave in their day to day information sharing activities. Even though humans use the majority strategy a lot (the entire notion of democracy is based on it), it is usually reserved for circumstances in which all affected parties must be seen equally. Sharing private information is not one of these circumstances since the entity that makes the determination (it can be a person, a group of people, an authority, etc.) usually takes into consideration the relationship between the requester and the owners of the policies in scope.

This similarity to a “real life” scenario implies that even though end users probably will not be able to fully understand the reason why a given resolution was taken, they will have a certain instinctive justification for the final ruling.

Equity

If we consider the unfair situations briefly discussed in section 1.2, topology strategies do not necessarily present an improvement over the previously discussed strategies, since it is also possible to construct a scenario for which a user’s privacy policy is consistently not enforced. However this notion of equity (as the other strategies) does not consider the topology of the social graph in order to measure if a given strategy is “equitable” or not.

5. MODELING OUR SYSTEM FOR EQUITY ANALYSIS

5.1 The system as an automaton

In section 3.2 we discussed a formalization for the notion of equity suitable for scenario analysis. However, most systems involve actions and reactions as well as changing states and therefore cannot be properly represented by a static scenario.

To overcome this issue, we need to use a formalism that allows to represent dynamic systems such as SNS. To that effect we have chosen the automaton model, that allows to represent systems as a set of states (possibly infinite) and transitions that allow moving from one to another.

Hopcroft and Ullman [20] define an automaton as a 5-tuple $\langle Q, \sigma, \delta, q_0, F \rangle$ where:

- Q is a set of states.
- σ is a set of symbols, called the alphabet of the automaton.
- δ is the transition function, that is, $\delta : Q \times \sigma \rightarrow Q$.
- q_0 is the start state, that is, the state of the automaton before any input has been processed, where $q_0 \in Q$.
- F is a subset of states (i.e. $F \subseteq Q$) called *final*¹ states.

We will work with a small modification of this formalism that better fits our needs: instead of the σ set of symbols (*alphabet*) we will use σ to identify the transition in the form of pairs of states.

$$\sigma = \{(q, t) | q, t \in Q, q \neq t\}$$

We are able to do so because of the restrictions to the automaton's underlying graph discussed in the following section.

5.1.1 Underlying graph

In the general case, if we restrict our view of an automaton to its states and transitions, we can describe it purely as a directed graph. In the case of our modification to the traditional formalization it is simpler, since it is fully described in the components Q and σ .

However, because of the manner in which we construct the automaton and the fact that our representation constrains the ruling space to two possibilities, we know that the underlying graph is in fact a binary tree in which each node's two children reflect the status of the system (with the corresponding following request if applicable) if the original node's request was either allowed or denied.

¹ Some literature also calls them *accepting* states.

5.1.2 Equity in the states

For the purpose of this work, we will analyze the equity of the system in each state $q \in Q$, and how different transitions through the δ function affect equity. In order to do this, we need to know, in each state, the *mesuration* magnitude for each party in the system.

Considering that the *mesuration* magnitude is a function of *wealth*, and *requital*, in each state $q \in Q$ and for each of the n parties, the system stores the tuple $\langle w_i, r_i \rangle$, corresponding to party i 's *wealth* and *requital* respectively.

However, each party's *wealth* is immutable. Therefore, there is no requirement to store this value in each state, leaving only the current *requital* for each party. We will hence introduce the following notation:

- R_p^q represents the *requital* for party p in state q .
- W_p represents the *wealth* for party p .

5.1.3 The transition function

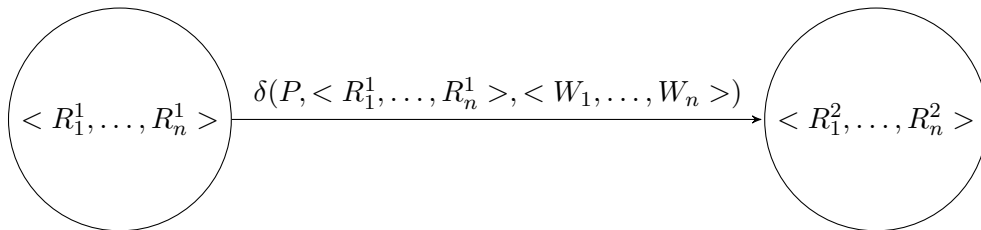
The semantics of both the states and the transition function has not been specified in the previous sections because we are discussing systems in an abstract way. The aforementioned semantics must be particularized in the context of the specific system. In particular the transition function always moves the system from one state to another (possibly the same) and will affect the requital for a (possibly empty) subset of parties in the system.

Exactly how the transition function changes the system and what information it uses to make this determination is particular to each system, but it can take into consideration the current equity status of the system (as well as other case-specific parameters).

Bearing this in mind and following the nomenclature defined in section 5.2, we shall define the transition function δ as a function that takes a set of requitals (and possibly other parameters) and returns a new set of requitals for the next state. Let s be a system with p_1, \dots, p_n parties and q_1, \dots, q_m states. The signature for δ is

$$\delta : P, \langle R_{p_1}^{q_j}, R_{p_2}^{q_j}, \dots, R_{p_n}^{q_j} \rangle \times \langle W_1, W_2, \dots, W_n \rangle \rightarrow \langle R_{p_1}^{q_{j+1}}, R_{p_2}^{q_{j+1}}, \dots, R_{p_n}^{q_{j+1}} \rangle \quad (5.1)$$

where P is the set of case-specific parameters.



5.2 Equity for the system

We still need to determine a notion of equity for the system. There are several possible candidates for this. Let us consider an automaton-based system, as the one described in

section 5.1. Lets suppose we have a function $\mathfrak{E} : Q \rightarrow Bool$ that given a state $q \in Q$ determines whether said state is equitable or not.

Using this, we can define the system's equity based on:

- **Final states:**

- The system is equitable iff² all of its final states are equitable: $\forall q \in F : \mathfrak{E}(q)$. We shall call such systems **final-all-equitable**.
- The system is equitable iff the majority of its final states are equitable: $\#\{q|q \in F, \neg\mathfrak{E}(q)\} \leq \#\{q|q \in F, \mathfrak{E}(q)\}$. We shall call such systems **final-majority-equitable**.
- The system is equitable iff at least one of its final states is equitable: $\exists q \in F : \mathfrak{E}(q)$. We shall call such systems **final-single-equitable**.

- **All states:**

- The system is equitable iff all of its states are equitable: $\forall q \in Q : \mathfrak{E}(q)$. We shall call such systems **all-equitable**.
- The system is equitable iff the majority of its states are equitable: $\#\{q|q \in Q, \neg\mathfrak{E}(q)\} \leq \#\{q|q \in Q, \mathfrak{E}(q)\}$. We shall call such systems **majority-equitable**.

- **Some subset of distinguished states:** let $\overline{Q} \subseteq Q$ be a subset of distinguished states of the automaton.

- The system is equitable iff all of its distinguished states are equitable: $\forall q \in \overline{Q} : \mathfrak{E}(q)$. We shall call such systems **distinguished-all-equitable**.
- The system is equitable iff the majority of its distinguished states are equitable: $\#\{q|q \in \overline{Q}, \neg\mathfrak{E}(q)\} \leq \#\{q|q \in \overline{Q}, \mathfrak{E}(q)\}$. We shall call such systems **distinguished-majority-equitable**.
- The system is equitable iff at least one of its distinguished states is equitable: $\exists q \in \overline{Q} : \mathfrak{E}(q)$. We shall call such systems **distinguished-single-equitable**.

It is worth noting that the **final** definitions are a particular case of this one.

- **The transitions:** another approach to consider the equity for the entire system is to evaluate how transitions affect the equity of states. This is strongly dependent of the properties of the case-specific δ function.

- The system is equitable iff every transition leaving an equitable state reaches an equitable state: $\forall (q_1, q_2) \in \sigma : \mathfrak{E}(q_1) \Rightarrow \mathfrak{E}(q_2)$. We shall call such systems **transition-equitable**.

Subsumption

There are some definitions that subsume others:

- *final-all-equitable* \preceq *final-majority-equitable*.
- *all-equitability* \preceq *final-all-equitable*.

² If and only if.

- *all-equitability* \preceq *final-majority-equitable*.
- *all-equitability* \preceq *final-single-equitable*. This subsumption only holds for a non-empty set of final states.
- *all-equitability* \preceq *majority-equitable*.
- *all-equitability* \preceq *transition-equitable*.
- *final-majority-equitable* \preceq *final-single-equitable*. This subsumption only holds for a non-empty set of final states.
- *transition-equitable* \preceq *final-single-equitable*. This subsumption only holds for systems with at least one equitable non-dead-end state³.

The proof of most of these subsumptions is trivial. We will only show the proof for *final-all-equitable* \preceq *final-majority-equitable* and *transition-equitable* \preceq *final-single-equitable*:

Proof. final-all-equitable \preceq *final-majority-equitable*: lets assume a *final-all-equitable* system. By definition, this means that $\forall q \in F : \mathfrak{E}(q)$, which directly implies that $\#q \in F : \neg \mathfrak{E}(q) = 0$.

Since there are no inequitable final states, we can observe that $\#\{q|q \in F, \neg \mathfrak{E}(q)\} = 0 \leq \#\{q|q \in F, \mathfrak{E}(q)\}$ which is trivially true since a set cannot contain a negative number of elements. Therefore, the system is also *final-majority-equitable*. \square

Proof. transition-equitable \preceq *final-single-equitable*: first of all, lets show that the imposed restriction is actually necessary. This restriction requires that there must be at least one state that must be equitable and non-dead-end.

- If a system has no states it is, by definition *transition-equitable*, since $\forall (q_1, q_2) \in \emptyset : \dots$ is trivially true. However, such a system is not *final-single-equitable* since $\exists q \in \emptyset : \dots$ is false.
- If a system only has dead-end states as a equitable it could be *transition-equitable* but not *final-single-equitable*. We shall illustrate with an example, in which equitable states are marked with blue:

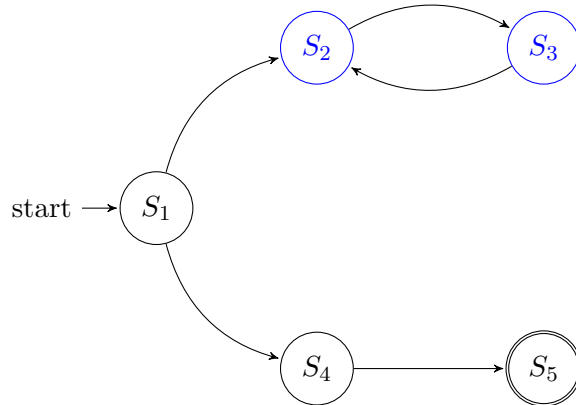


Fig. 5.1: Dead states counter example

³ A dead-end state is one from which it is impossible to get to a final state.

The automaton presented in figure 5.1, is a simple counter example to the *transition-equitable* \preceq *final-single-equitable* subsumption if we relax the restriction about dead-end states. That automaton is *transition-equitable*, since from every equitable state we can only reach other equitable states, but it is not *final-single-equitable*.

From the imposed restriction, we know that there must be at least one non-dead-end equitable state $q_0 \in Q \wedge \mathfrak{E}(q_0)$. Since q_0 is not a dead state, there must exist a path of length n ($q_0, q_1, q_2, \dots, q_n$ for $q_i \in Q$) such that q_n is final ($q_n \in F$).

We want to prove $\mathfrak{E}(q_n)$. Since $q_n \in F$, it will be true that $\exists q \in F : \mathfrak{E}(q)$ making our system *final-single-equitable*. To that end we shall use a simple induction on the length of the path (n).

Let the statement $P_i = \mathfrak{E}(q_i)$.

- **Base case:** $P_0 = \mathfrak{E}(q_0)$. This is true because of the imposed restrictions.
- **Inductive step:** $P_n \Rightarrow P_{n+1}$. By definition this is equivalent to $\mathfrak{E}(q_n) \Rightarrow \mathfrak{E}(q_{n+1})$. Since by construction, there is a transition between q_n and q_{n+1} (ie, $(q_n, q_{n+1}) \in \sigma$), the inductive step is true because we assumed the system was *transition-equitable*.

With this induction we have particularly proven that $P_n = \mathfrak{E}(q_n)$ (q_n is equitable). Since we assumed q_n is final ($q_n \in F$), we have found one final equitable state. Therefore, the system is *final-single-equitable*. □

Semantics

The meaning of these possible definitions depends on the semantic given to the states and transitions of the system in a specific case. For instance, we might consider a system that wants to assert that there is an equitable *end-result*, but it does not matter whether in the process there was equity. Such a system would follow a *final-single-equitable*. An example of this behavior can be observed in the distribution of an inheritance amongst siblings: it does not matter how equitable intermediate situations are, as long as there is an equitable distribution as a final state, the system can be considered equitable.

In other systems for which the end-result is not particularly important (or simply there is no end-result, like in some infinite automata) the measurement for equity for a system could be the proportion of total states that are or not equitable.

5.2.1 Algorithms

In this section we will present the algorithms to compute the state equity of a system according to the different definitions provided in section 5.2. This algorithms will be used to compare the equity characteristics of different conflict resolution strategies in chapter 7, using the tools developed in chapter 6.

We will assume the following data structures:

- User
- Requitall

- Wealth

We will assume the following complexities:

- Let n be the number of users in the system.
- Obtaining the first key of a **Map** structure has a complexity of $O(1)$.
- Searching a **Map** structure of size m has a complexity of $O(\log(m))$.

Note: All pseudocodes presented in this section are intended to expose the idea and complexity of their respective procedures, not the technicalities. We therefore specifically ignore validations, type checks, exceptions and so on.

Algorithm for state equity

Following the formalization presented in sections 3.2 and 5.1.2, we present the following algorithmic approach to determine whether a state is equitable:

Algorithm

```

procedure ISEQUITABLE(Map<user,requital>  $R$ , Map<user,wealth>  $W$ )
   $someUser \leftarrow R.getFirstKey()$ 
   $mensuration \leftarrow \mathfrak{M}(R.getValue(someUser), W.getValue(someUser))$ 
  for each  $user \in R.allKeys()$  do
    if  $mensuration \neq \mathfrak{M}(R.getValue(user), W.getValue(user))$  then
      return false
    end if
  end for
  return true
end procedure

```

Complexity

Let m be the size of the set s . We note the cost of computing the function \mathfrak{M} as merely \mathfrak{M} .

The complexity of this procedure is bounded by: $O(n * \log(n) + n * \mathfrak{M})$ which is:

$$\boxed{O(n * (\log(n) + \mathfrak{M}))} \quad (5.2)$$

Algorithm for *-all-equitable

The following procedure allows to determine whether a system is **-all-equitable*. Depending on the set of states presented as input, the procedure determines *all-equitability*, *final-all-equitability* or *distinguished-all-equitability*.

Algorithm

```

procedure IS*ALLEQUITABLE(Set<Map<user,requital>>  $S$ , Map<user,wealth>
 $W$ )
  for each  $state \in S.allKeys()$  do

```

```

if  $\neg$ isEquitable( $S$ .getValue( $state$ ),  $W$ ) then
  return false
end if
end for
return true
end procedure

```

Complexity

Let m be the size of the set s . We note the cost of computing the function \mathfrak{M} as merely \mathfrak{M} .

The complexity of this procedure is bounded by $O(m * isEquitable(n))$ which is:

$$\boxed{O(m * n * (\log(n) + \mathfrak{M}))} \quad (5.3)$$

Algorithm for *-single-equitable

The following procedure allows to determine whether a system is **-single-equitable*. Depending on the set of states presented as input, the procedure determines *final-single-equitability* or **distinguished-single-equitability**.

Algorithm

```

procedure IS*SINGLEEQUITABLE(Set<Map<user,requital>>  $S$ , Map<user,wealth>
 $W$ )
  for each  $state \in S.allKeys()$  do
    if isEquitable( $S$ .getValue( $state$ ),  $W$ ) then
      return true
    end if
  end for
  return false
end procedure

```

Complexity

Let m be the size of the set s . We note the cost of computing the function \mathfrak{M} as merely \mathfrak{M} .

The complexity of this procedure is bounded by $O(m * isEquitable(n))$ which is:

$$\boxed{O(m * n * (\log(n) + \mathfrak{M}))} \quad (5.4)$$

Algorithm for *-majority-equitable

The following procedure allows to determine whether a system is **-majority-equitable*. Depending on the set of states presented as input, the procedure determines *majority-equitability*, *final-majority-equitability* or *distinguished-majority-equitability*.

Algorithm

```

procedure IS*MAJORITYEQUITABLE(Set<Map<user,requital>>  $S$ , Map<user,wealth>
 $W$ )
   $eq \leftarrow 0$ 
   $noneq \leftarrow 0$ 
  for each  $state \in S.allKeys()$  do
    if  $isEquitable(s.getValue(state), W)$  then
       $eq ++$ 
    else
       $noneq ++$ 
    end if
   $half \leftarrow \lceil \frac{S.size()}{2} \rceil$ 
  if  $eq > half$  then
    return true
  else if  $noneq > half$  then
    return false
  end if
end for
return true
end procedure

```

Note: when there is exactly the same number of equitable and non equitable states, a decision must be made regarding the **majority-equitability** of the system. If such a situation arises, this procedure determines that the system is *majority-equitable* in accordance with the formalism presented in section 5.2. However, this is an arbitrary decision without theoretical ground and can easily be changed.

Complexity

Let m be the size of the set s . We note the cost of computing the function \mathfrak{M} as merely \mathfrak{M} .

The complexity of this procedure is bounded by $O(m * isEquitable(n))$ which is:

$$\boxed{O(m * n * (\log(n) + \mathfrak{M}))} \quad (5.5)$$

Algorithm for transition-equitable

In this section we will present the procedure that allows to determine whether a system is *transition-equitable*.

Algorithm

```

procedure ISTRACEEQUITABLE(Set<Pair<Map<user,requital>, Map<user,requital>>>
 $\sigma$ , Map<user,wealth>  $W$ )
  for each  $statePair \in \sigma$  do
    if  $isEquitable(statePair.first) \&\& \neg isEquitable(statePair.second)$  then
      return false
    end if

```

```

end for
return true
end procedure

```

Complexity

Let m be the size of the set s . We note the cost of computing the function \mathfrak{M} as merely \mathfrak{M} .

The complexity of this procedure is bounded by $O(m * isEquitable(n))$ which is:

$$\boxed{O(m * n * (\log(n) + \mathfrak{M}))} \quad (5.6)$$

This complexity bound holds because, as stated in section 5.1.1, the automaton's underlying graph is a binary tree. Therefore each state will be evaluated at most 3 times.

5.2.2 Measuring equity - the Gini coefficient

So far we have talked about equity as a binary possibility. A state is equitable or it is not.

However it is reasonable to assess a situation's **distance to equity** as a way to measure "how equitable" is a situation. This also expands the possibilities for the definition of equity for a system, for example by proposing modified versions of all previous definition to state that the distance to equity must be less than certain percentile for all states, or a similar notion.

The approach we will take to tackle this issue is to use the Gini coefficient as a measurement for how inequitable a situation is.

The **Gini coefficient** is a standard unit of measurement for income equity in the field of economics. Proposed in 1912 by italian statistician Corrado Gini in his book "Variabilità e mutabilità" [21], this measurement is based on the difference between the Lorenz curve [22] and the line of equity⁴.

The Lorenz curve is a graph formed by the the cumulative distribution function of the wealth of a given population. Colloquially, the curve represents which percentage of the wealth is owned by which percentage of the population, sorted from lowest to greatest. It is a monotonically increasing function that passes by the $(0,0)$ and $(100,100)$ ⁵ points (because 0% of the population, by definition have 0% of the wealth, whereas 100% of the population have 100% of the wealth).

The Gini coefficient is calculated by dividing the area between the Lorenz curve and the line of equity by the total area under the line of equity. If we label said areas as A and B as shown in figure 5.2, then the Gini coefficient is calculated by:

$$\boxed{G = \frac{A}{A + B}} \quad (5.7)$$

⁴ Sometimes called the *egalitarian line* [23, 24]

⁵ Or $(1,1)$, depending on the scale.

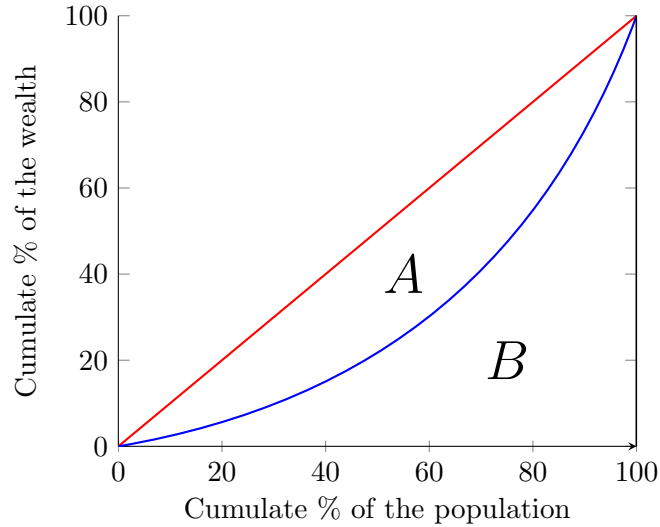


Fig. 5.2: Sample Lorenz curve

Possible values for the Gini coefficient can range from 0 (total inequity, with one party accumulating all the wealth, represented by G_0 in figure 5.3) to 1^6 (total equity, the wealth is equally divided amongst all parties, represented by G_4).

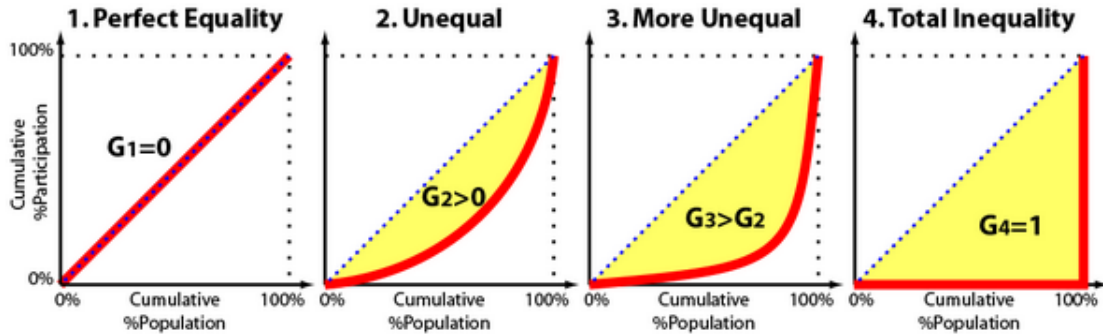


Fig. 5.3: Different values for the Gini coefficient

5.3 Model for Social Networking Systems

In the previous section we discussed a general model for analyzing equity in systems. We now need to discuss the notion of equity specifically tailored for SNSs, returning to the schema started in section 4.2.1 to discuss the definition of conflict.

When a user makes a request to access a certain picture, the system may authorize or deny that request based on, amongst other things, the policies defined by the users tagged in the picture. Therefore, the magnitude *requital* that the system can affect must be related to the aforementioned authorization or denial.

One approach, chosen by Marin et al [7], is to consider a situation equitable if **all the tagged users have seen their policies enforced or violated in the same**

⁶ Assuming non-negative values.

proportion over past interactions. An algorithm was designed so that when a user makes a request, the system will enforce the policy that most improves the equity in a greedy fashion. With this approach the *requital* magnitude is the number of times each user's policy has been granted, and the *wealth* is the proportion.

5.3.1 Automaton representation for SNS

For our study scenario we will use the system representation of automaton described in section 5.1.

We will represent our system with an infinite set of states, with transitions that correspond to a ruling. The requests and the situation previous to it (i.e. the situation that got evaluated to determine the ruling) are embedded within the states.

It is important to notice that the ruling referred to in the previous paragraph is a system ruling. It is the result of the evaluation of the composition function from the rulings of all users involved (i.e. the system's final decision). This representation does not include the process of ruling composition $\mathcal{C}(\{\mathcal{R}(i, c, p, ts)\}_{i \in T_p})$. That, along with the request, is embedded within the state and its result determines the transition.

Modeling parameters

In order to apply our automata model we need to determine the *magnitudes* described in section 3.1.

Wealth

This study scenario is a proper example of the types of system mentioned in section 3.1 where the magnitude *wealth* is not present. Because of the way we defined the measurements, all magnitudes are an artificial construct of the system, with no *wealth* inherent to each party.

Requital

For our study scenario we will measure the *requital* as **the proportion of successful policy enforcements over past interactions**. That is we will consider the formula:

$$requital = \frac{\#enforced}{\#involved}$$

where:

- *#enforced* is the number of times the user's policy has been enforced by the system.
- *#involved* is the total number of times the user's policy has been involved in a request.

Mensuration

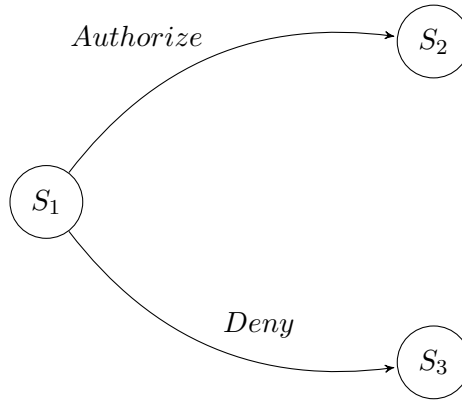
Since in the long run it is highly unlikely for users to have the exact same proportion of policy enforcements, we will envisage a staggered approach, utilizing the \mathfrak{M} function to group proportions into ten-percentages:

$$\mathfrak{M}(\text{requital}) = \lceil \text{requital} \times 10 \rceil \quad (5.8)$$

Since the requital is a number between 0 and 1, this formula will return a natural number between 0 and 10 which we will consider to be our *mesuration*.

Transition example

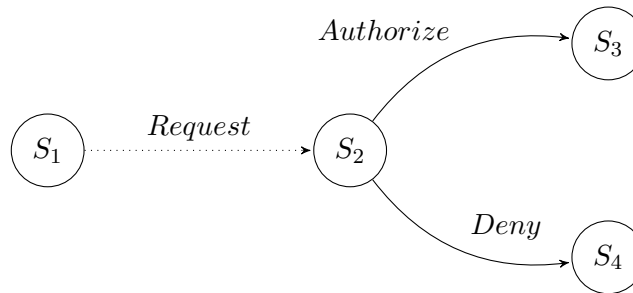
A graphical example of an automaton following this semantic would be:



Current state problem

This particular representation in its current form has the issue that it is not possible to represent a system's "current" state, i.e. without a request.

A possible solution for this problem is to have different types of transitions: one corresponding to the request and another to the ruling. A graphical example:



However we have decided for the time being not to follow this path because this representation is substantially more complex (which hinders legibility and simplicity), but it does not accompany with a corresponding proportional increase in expressivity. It does solve the aforementioned problem, but it also adds several semantic constraints that must be observed for the automaton to be valid, such as:

- If two states p and q are linked by a *request* transition, then $\forall p \in P; R_p^q = R_p^s$.
- Any trace must be a succession of *request* – (*allow*||*deny*) – *request* – (*allow*||*deny*) –

An alternative solution for the aforementioned problem is to have a particular semantic to distinguish the “current” state, that has no request yet from the others. A different notation would not be necessary, since we can simply alter the agreed upon semantic: any state that has no transitions leaving from it is considered a current state and it only represent the system’s situation (without the request).

States in the automaton

In order to properly visualize all the information present in the automaton, we will enhance our notation in the following way:

Let us consider a system with 4 users, each of which had had their respective policies evaluated 10 times and only enforced 5. We will represent a state $q \in Q$ with a node with the following information:

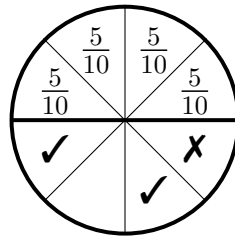


Fig. 5.4: Status representation

The upper half of the node depicted in picture 5.4 represents the four users and for each user, his *requital*. The corresponding lower half, also represents the four users, but notices the user’s involvement in the current request, with the following nomenclature:

- ✓ means that the user’s ruling is *allow*.
- ✗ means that the user’s ruling is *deny*.
- An empty space means the user’s policy is not involved in that request.

Example

In this section we will analyze a particular sequence of requests, with their possible results using the notation previously defined.

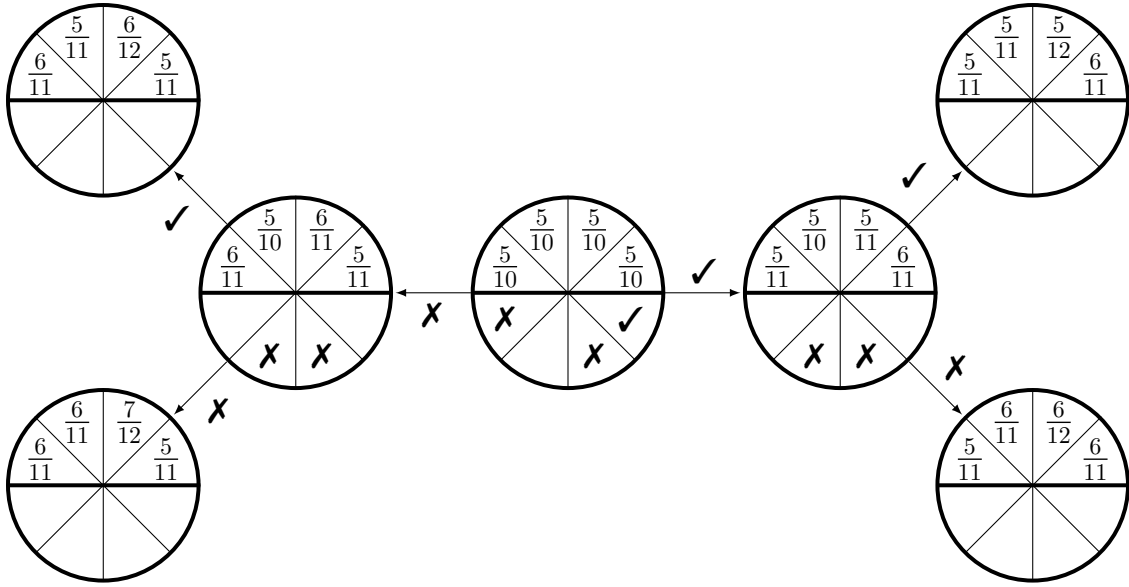


Fig. 5.5: Example of a series of requests with our notation

There are a few interesting things to notice about figure 5.5:

- We are considering a scenario in which if a user is tagged in a picture that he wants to access, he must make a request and determine a ruling like everyone tagged. Then the system will compose all rulings and determine if he is allowed access. This means that a user might be denied access to his own picture.
- It represents all the possible resolutions from a given moment forward, provided a request sequence and a set of response profiles.
- Because of this, it does not show any information regarding the system's chosen strategy: it applies to any and all of the previously defined resolution strategies for the study scenario.
- The system strategy actually determines which will be the trace (i.e. the path from the root to a leaf) that the system's execution will follow.
- It is worth noting that to construct this graph we considered two major hypotheses:
 - We do not allow simultaneous requests;
 - We assume independence between individual rulings and past history. A users' determination is not affected by the previous requests or their respective rulings.

The aftermath of these considerations is that our model has an ordered sequence $\{R_1, R_2, \dots, R_n\}$ of requests with their corresponding rulings by each affected party. For any two nodes such that their distance to the root is equal, the lower half of the circle has to be the equal (i.e. there has to be the same request, with the same replies).

- This model and representation allows us to analyze many things about the system. Depending on which aspect we wish to analyze we can focus on a different facet. For example:
 - We could use it to analyze the mere probability of a state to be equitable (we are not going to continue down this line of work for the time being);
 - If we introduce the notion of distance to equity, we can analyze the traces through which the system traverses to improve equity.
 - In a more general sense, this graph allows us to observe all possible outcomes for systems determinations. This effectively includes all possible strategies taken by the system, each represented by a particular trace. With this information we can compare the results that all strategies would have on a series of requests. We can even compare traces that would not result from any strategy. Some questions we might answer regarding strategies could be:
 - * Which strategy traverses the most equitable states?
 - * Which strategy adds the smallest distance to equity?
 - * Is there any strategy that maintains equity? (i.e. if it reaches an equitable state, all subsequent states it traverses are equitable).
 - * Is there any strategy for which equity always improves?

A posteriori analysis

The nomenclature and analysis developed in the previous section has a major drawback: it only works for post-facts analysis, because it needs the knowledge of all the requests involved. It is indeed a useful mechanism to compare different system strategies, definitions of equity, etc. But unless you have an oracle that knows future requests, this analysis does not provide a mechanism to determine a general-case system strategy for a composition of rulings.

System equity

For this study scenario we are drawn towards utilizing the **majority-equitable** definition for system equity. This is because we are interested in a system that maintains equity as most as possible.

In our study scenario we consider all states equal, therefore, we don't have final or distinguished states that we want to particularly examine.

The **transition-equitable** definition states, in colloquial terms, that a system is equitable if and only if it can maintain equity once it reaches it. In our study scenario this is not true for the general case because the system is required to take a determination that will inevitably have an impact on the *mensuration*, possibly moving a state from an equitable situation to an inequitable one (ie, from an equitable state, either if the system allows or denies the request, it will end up with an inequitable state).

For similar reasons direct **all-equity** is out of consideration; even though it is an ideal case (where every single state is equitable) it is not feasible to have it as a system definition for equity because it is very unrealistic.

6. THE SIMULATOR

In order to analyze how different strategies behave over some fixed set of requests, we made a simulator in python following the scheme presented in section 5.3.1.

The simulator works by maintaining a current status which is updated based on the requests and the system strategy.

- Conceptually, the current status is determined by a map<user,#enforcedPolicy,#involvedPolicy>.
- The requests are loaded from a file that contains m lines, each representing a request. Each line contains n numbers (where n is the number of users in the system) that represent each user's ruling to the request in the following way:
 - 0 means that the user's ruling was to deny the request.
 - 1 means that the user's ruling was to allow the request.
 - 2 means that the user's policy was not involved in the request (i.e. he was not tagged in the picture).

It is important to notice that we're abstracting the entire request to the rulings of the involved users. Since we're taking a system-strategy approach, how they reached that particular ruling is none of our concern for the time being.

The simulator has a number of functions that can be changed to adapt it to the intended simulation:

- **systemStrategy**: it's a function that takes as parameter the current status and the user rulings, and it determines how the system performs the policy composition to choose whether to allow or deny the request.
- **isEquitable**: it's a function that determines whether the current state (taken as parameter) is equitable.
- **store**: as mentioned, the simulator allows to iterate through a trace to evaluate the current strategy and equity formulae. **Store** is a user defined function that, given the current status, determines what must be saved into the trace. For example, this could be the entire status, whether it is equitable or not, etc.

Note: Because of how our simulator takes the input, the set of requests and their respective responses are predetermined. Therefore, **we are assuming independence between the past system's rulings and rulings by individual users.**

6.1 Programmed strategies

The simulator has several built-in strategies:

- **Allow**;
- **Deny**;

- **Majority;**
- **SNS.** This represents the typical strategy taken by SNSs to resolve privacy conflict: one distinguished user (usually the one who provided the information) makes the determination. For the purposes of the simulator, we have chosen to always use the last determining entry (i.e. the last non-2 value) for each request;
- **EIS (Equity Improving Strategy).**

6.1.1 EIS Strategy

The EIS is a proof of concept strategy that we developed to use as a basis for comparison against the traditional strategies.

This heuristic strategy greedily attempts to compose the rulings in the manner that will yield the most equitable result at each point, guided by all involved parties. To that end it will compute the Gini coefficient of the resulting state if it were to approve the request and compare it with the one if it were to deny it. The determination will be the lower of the two.

One important thing to notice about the EIS strategy is that it does not necessarily yield the optimal path (i.e. the path that minimizes inequity, either the sum of all Gini coefficients for the path, or the Gini coefficient of the final state). In order to prove that, let us reconsider our example from section 5.3.1:

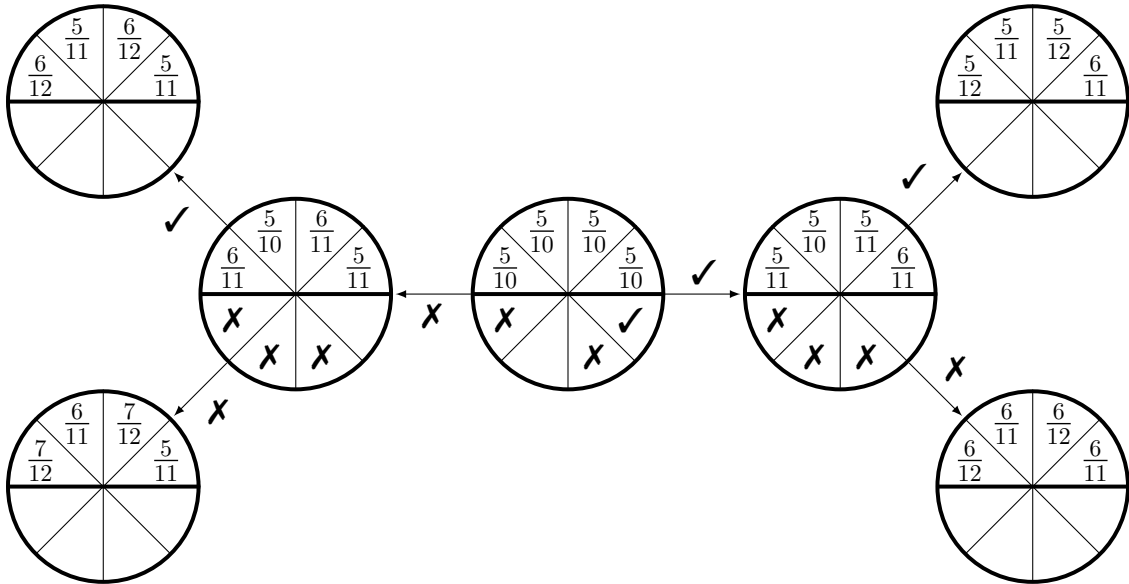


Fig. 6.1: Non optimal path with EIS (requests)

We will replace each node in figure 6.1.1 with the corresponding Gini coefficient for that state:

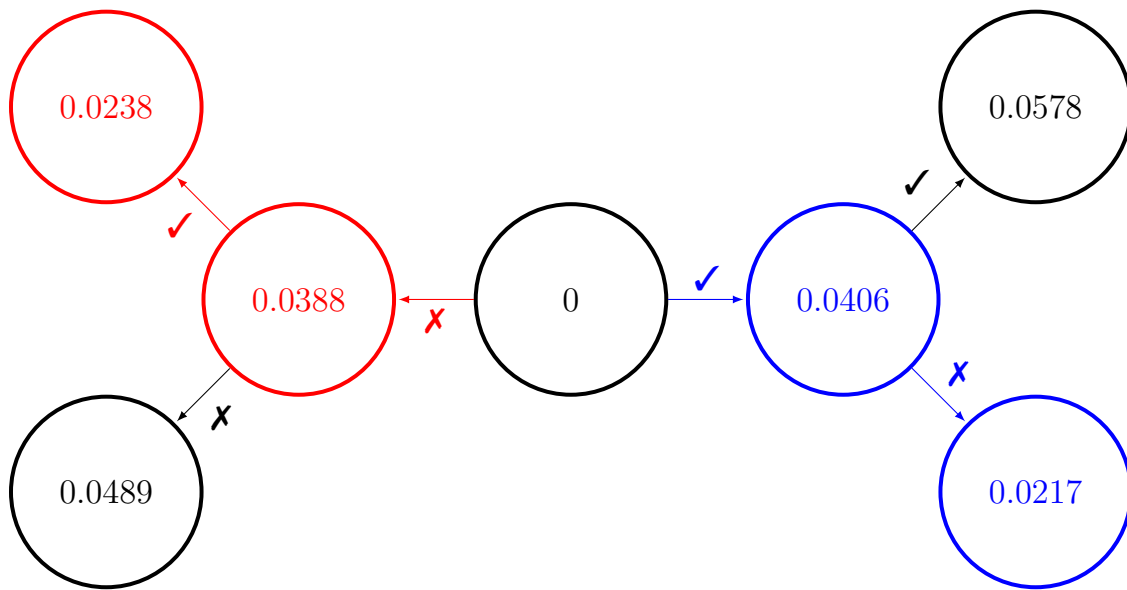


Fig. 6.2: Non optimal path with EIS (Gini coefficients)

As can be seen in figure 6.2, upon the first request, the decision that yields the most equitable situation is to deny the request, since a Gini coefficient for a situation of 0.0388 is preferred over one of 0.0406. Therefore, the EIS strategy will resolve the conflict by denying the aforementioned request. Similarly, when the second request arrives, it will allow it. The EIS strategy will result in the path marked in red.

However, as we can clearly see, out of the 4 possible combinations of determinations the system might, the path in blue offers the best results, both considering the value of the Gini coefficient at the final state or the sum over all states of the trace.

Abuse

EIS strategy is a proof of concept strategy. It is not directly applicable in real life scenarios because, in its current state, it could be manipulated by a malicious user to gain access to a picture under a few assumptions:

- The user can do an arbitrary number of requests on the picture.
- At least one user whose policy involves the desired picture agrees on granting him access.

Since the system attempts to balance the proportion of *enforced* and *involved* requests, if a user wants access to a given piece of information, he only needs to make enough requests that he will get it, assuming no other interactions that compensate.

In order to illustrate this point, let's assume a system with 5 users, with 4 of them tagged in a given picture. The remaining user wants access to that picture and makes a stream of requests to obtain that picture. We assume that 3 of those users want to deny the request and the remaining one wants to allow it.

The system following EIS strategy will allow or deny the request depending on the conditions of the system. Even if we assume initial conditions that benefit the three users that want to deny the request, by the mere fact of making successive requests (and getting

consistently denied), the malicious user can gradually modify the situation to benefit the one user that wants to allow the request.

Lets use our graphical notation to build this example.

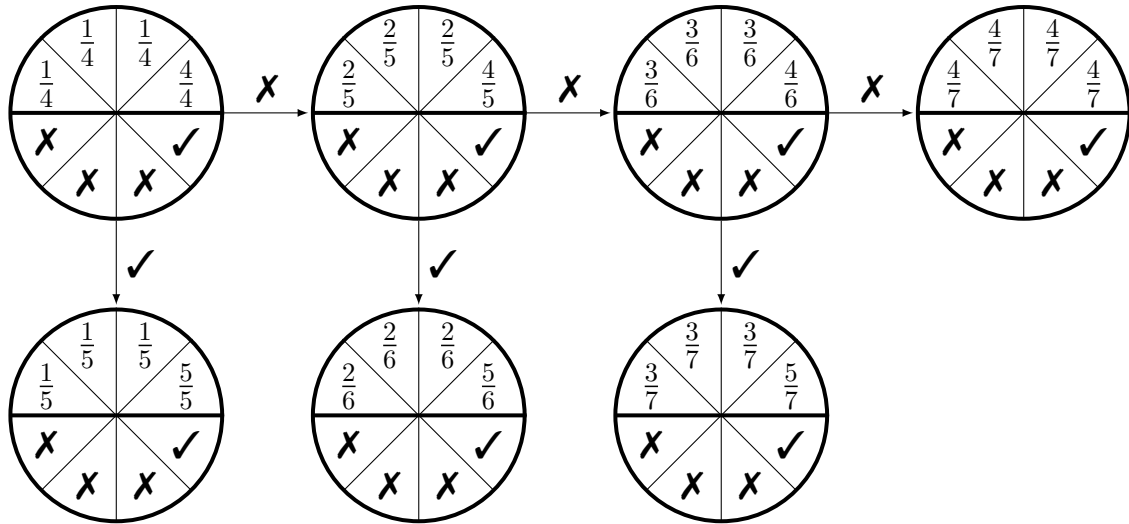


Fig. 6.3: Abusing EIS strategy (requests)

If we compute the Gini coefficient for each state of figure 6.3,

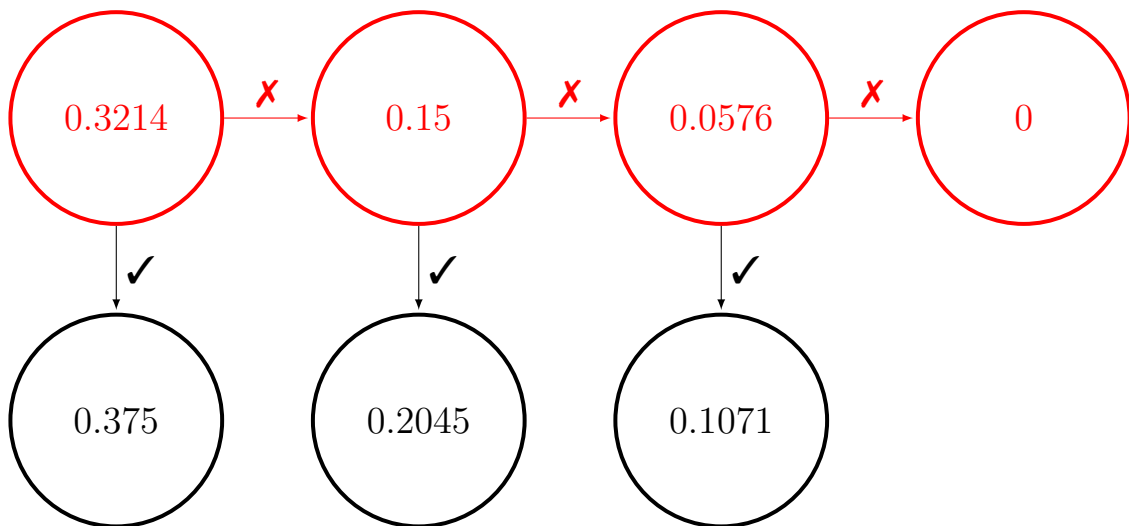


Fig. 6.4: Abusing EIS strategy (Gini coefficients)

In figure 6.4 we can observe that the EIS strategy would follow the path in red, leading to a state of complete equity. From there, regarding how the strategy is configured to respond to a request on perfect a perfectly equitable situation it can either take one or two more requests for the user to obtain access to the file:

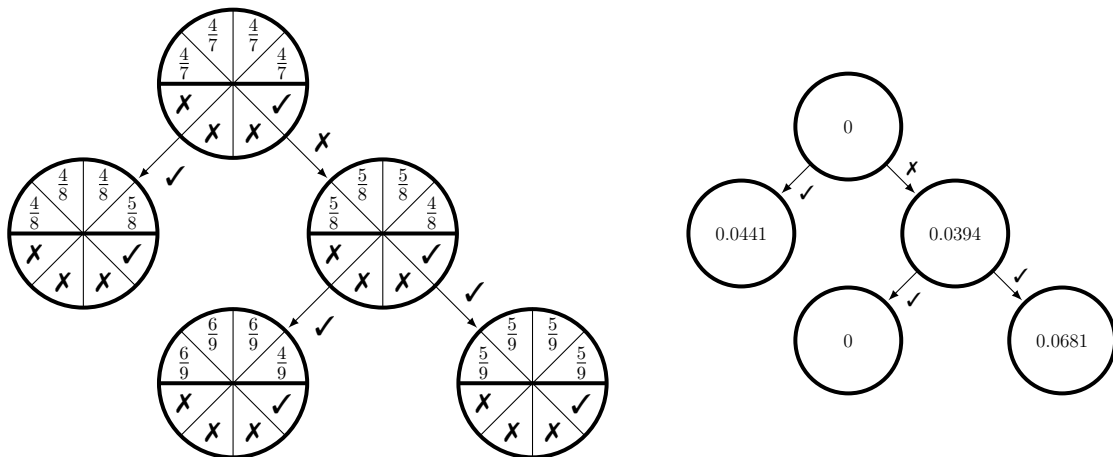


Fig. 6.5: Abusing EIS strategy after equity (requests and Gini coefficients)

As we can see in figure 6.5, even if the *EIS* strategy denies the request on perfect equity (which, as configured in our simulator, it does) it only takes one more request for the user to get access to the picture.

Even though this behavior is not strictly a security breach, but actually how the strategy is intended to work, the fact that users can manipulate the system and gain access to information by a mere flooding of requests makes it unfeasible to be directly applicable to real-life SNSs.

Furthermore, a situation can arise with *EIS* strategy in which all involved users agree on the outcome of a request and the system chooses to do the opposite, which might help with equity but is very counter-intuitive. Lets illustrate the situation with a simple example:

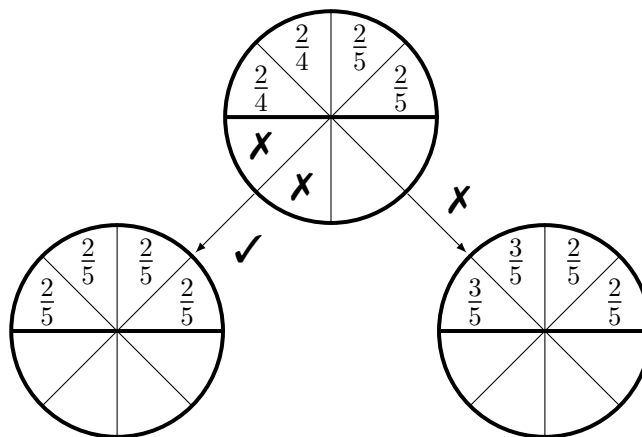


Fig. 6.6: Counter intuitive determination

As we can observe in figure 6.6, a user requests access to a picture affected by the security policies of two users. Both users determine that the request should be denied. However, because of the equity conditions of the system when the request was made, the

ending result would be more equitable to allow the request, even if it violates all involved policies.

6.2 Experimentations and Results

The experimentation for this work was done following two possible lines of behavior by different users: a random approach and a worst-case scenario. Both cases were generated with the tools provided with the simulator (`generateRequests.py`), varying its parameters.

6.2.1 Random

For the full-random approach we assumed a system with 20 users, making a total of **400** requests. Each user's response for each of the requests was fully chosen at random, having 33.333% chance of responding each option. This is obviously a simplification, since in a real world scenario, a user would not *choose* option 2 -not to be involved in a request-, but the system would impose that. However, for the purposes of this experimentation and since in this case the process of who determines the involvement is inconsequential, we abstract this process.

Each run of the 20 user - 400 requests scenario was run 100 times and its results averaged to minimize the possibility of hidden tendencies consequence of one particular random scenario.

These measurements (20 users, 400 requests) were chosen after experimentation. Smaller experiments proved to be unstable (not yielding consistent results between runs) and larger ones did not show significant improvement. We believe the values chosen are an appropriate trade-off between run-time, results and stability.

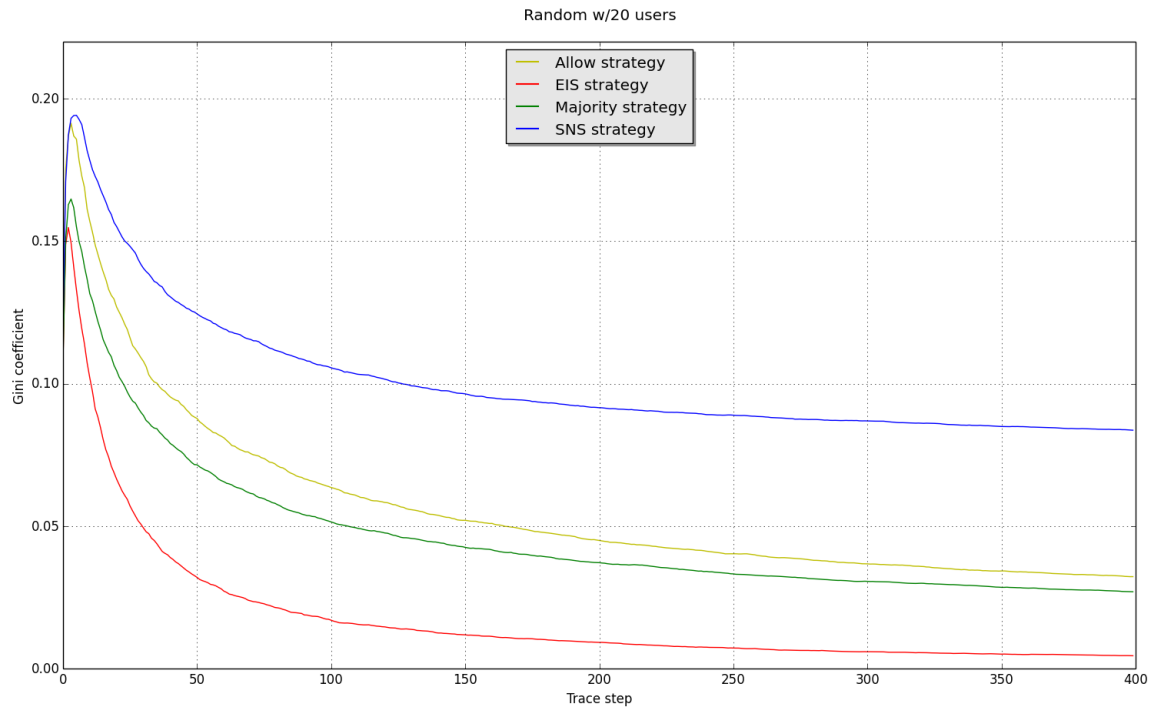


Fig. 6.7: Comparison between strategies in a randomly generated population

One thing we can observe in this randomly generated scenario is that the EIS strategy provides much more equitable decisions than the other strategies.

The average coefficient for the 400 requests in this experiment were:

- EIS strategy: **0.0179**
- Majority strategy: **0.0470**
- Allow strategy: **0.0569**
- SNS strategy: **0.1011**

This means that the EIS strategy was, in average, 2.62 times more equitable than majority, 3.17 more than allow and 5.64 more than SNS.

Another noteworthy information we can obtain from figure 6.7 is that the strategy chosen by most SNSs, to always resolve conflict by following the decision of one *special* user proves the most inequitable. Conceptually, this type of strategy is the most unfair because one user has the monopoly on the decision. However, because of the measure we've chosen to assess equity, if that person's ruling agrees with that of other users, their policies will count as equally enforced, not yielding a 1 value of the Gini coefficient as it should when one party has the monopoly of the decision.

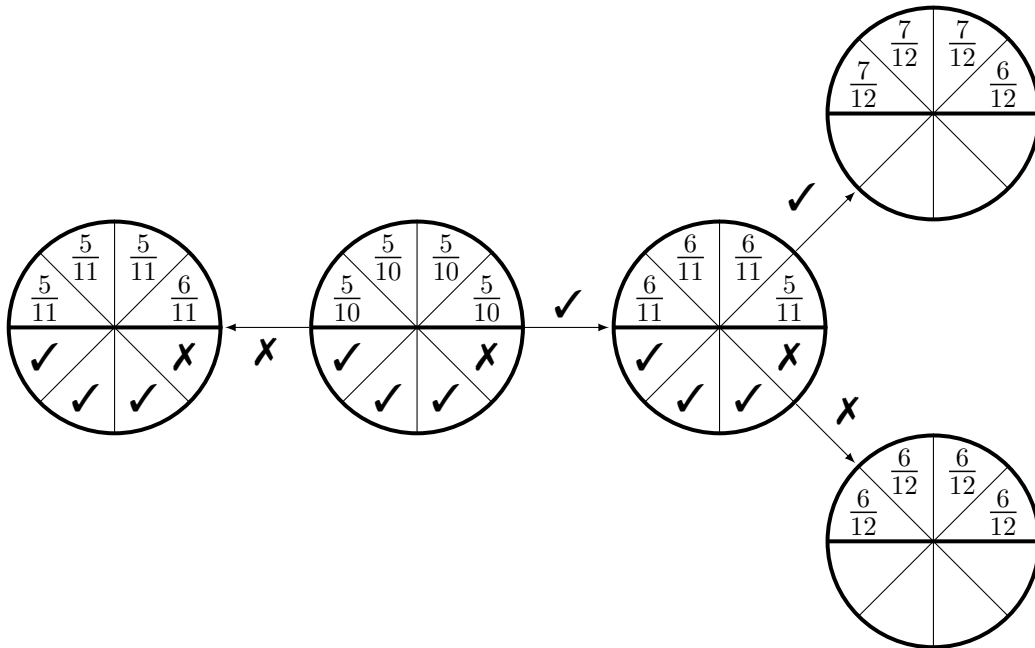
6.2.2 Worst case scenario

To clearly show how much the SNS strategy can affect equity we will construct a worst case scenario. We chose this worst case scenario because since the SNS strategy is the one used in the biggest SNSs, its impact in equity is the most relevant one, as it affects the most people.

In order for this scenario to pose the worst case for the SNS strategy we will have the deciding user's ruling to be the exact opposite of all the other users for each particular request. Repeating this process for a large number of requests will make the deciding user's *mensuration* approach 1 while all other users will see their respective *mensurations* tend to 0.

Since the differences in the EIS and SNS strategies are greatly accentuated in this scenario, and their functions grow quite abruptly, it becomes necessary to use a *logarithmic scale* in the *Y* axis in order to see the graph clearly. However, a complication arises with the logarithmic scale in this scenario, which the reader needs to be aware of.

The EIS strategy for this scenario presents a very interesting behavior. It will do an alternating sequence of *deny/allow*. Let us examine why with a small example:



As we can see, upon the first request, the decision that produces the lowest equity is to *deny* the request, since there are more users that agree in that, which minimizes the Gini coefficient, because it minimizes the relative difference between the *mensuration* for all users. Particularly, since all users start with the same number of *involved* and *enforced* times their policies were enforced, and all users participate in the request, all users will end up with 1 more *enforced* than the deciding user.

At the second step however, the system can compensate the missing *enforced* from the deciding user by enforcing his policy. This restores complete equity to the system, leaving a Gini coefficient of exactly 0 (the case when all users have the exact same *mensuration*).

Since this is a two-step repeatable process, the EIS strategy for this type of requests will always resolve a sequence of *deny - allow* in which after every *deny*, the system

arrives at a state which has a Gini coefficient of 0. These values of 0 pose a problem, since in logarithmic scale they can't be plotted. We have simply decided to ignore all 0 data points in order to plot the Gini curve, but the reader should be aware that half the datapoints are missing from that curve, and therefore the actual average value is much lower (approximately half, depending on the parity of the number of requests considered).

Reconsidering the definitions proposed in section 5.2, we could note that this particularly crafted series of request with EIS strategy makes the system close to *transition-equitable*. It is not directly *transition-equitable* because from every equitable state a non-equitable state is reached (if we define a state as equitable if it has a Gini coefficient of exactly 0). However, if we consider hops of distance 2, the strategy will always transform an equitable into another equitable state which is the notion behind transition equity.

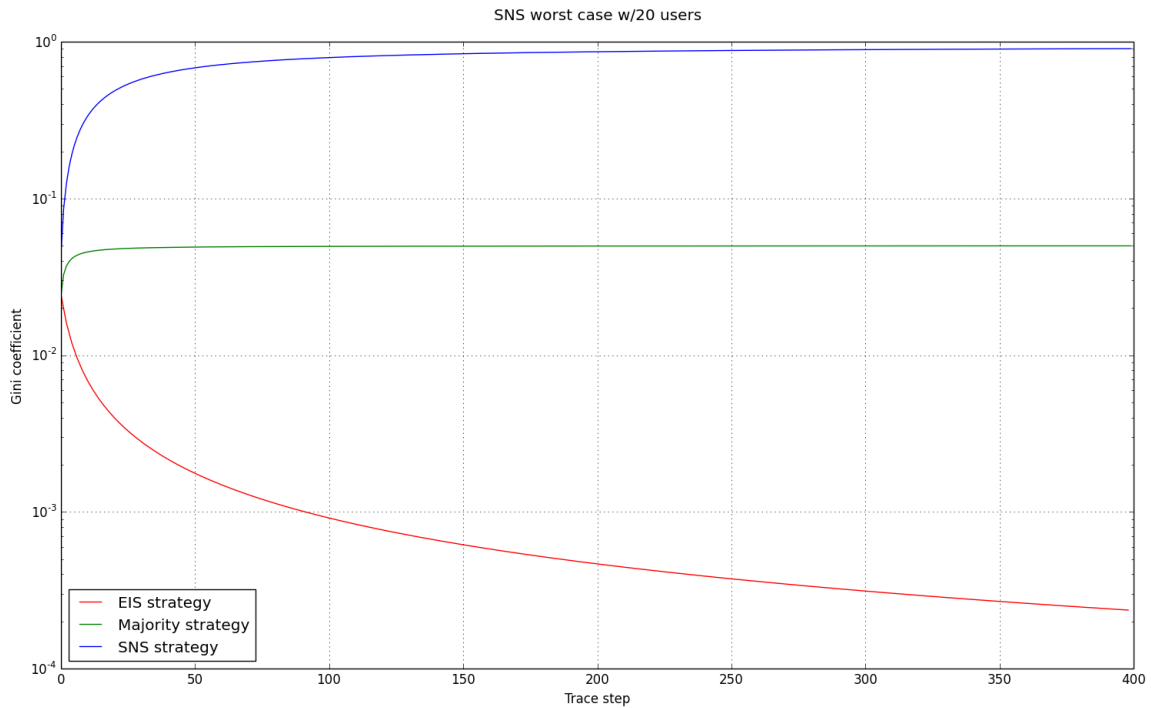


Fig. 6.8: Comparison between strategies in a worst case scenario

Another important thing to take into account from figure 6.8 is that the static strategies are missing. Since for this scenario an *allow* strategy would yield the exact same result as the SNS strategy, we didn't see the value in adding the information to the graph.

Analogously, a *deny* strategy would output the exact same result as the majority strategy.

7. CONCLUSION

7.1 Conclusions and future work

In this thesis we have characterized several aspects of equity from a computer science perspective, providing a formalization for its analysis, based on a brief but relevant debate on how other fields regard the notion of equity.

This jumping point allowed us to design a novel and abstract model based on the formalization developed to analyze the equity properties of systems.

We also presented Social Networking Systems as our case study introducing their relevant aspects concerning equity, establish formal definitions for conflict and reviewed the typical strategies they use to handle conflict in user defined privacy policies.

Last, we applied our proposed model to analyze our case study and were able to observe that the traditional way in which SNS resolve conflict present the most inequitable results.

Sackmann et al. assert that *a highly dynamic system* [such as SNSs] *is only privacy-aware if it enforces formalized and personalized privacy policies* [25]. As we have seen in the experimentation, in SNSs using the traditional conflict resolution strategy, situations can arise in which a user’s policy get consistently not-enforced, therefore making them not privacy-aware.

Even though the results were satisfactory in showing that the strategy current SNS strategy has little to no regard to equity, the scenarios used to test that were artificial. As part of future work we could take advantage of some widely used SNSs (Facebook for instance) to obtain real-life data on how users share information. This could be done in a manner similar than the works by Ravichandran et al [26] or Liu et al [4] in which an application is deployed which collects relevant data directly from users. This would allow for a more realistic study scenario to analyze the equity implications of conflict strategies.

Even though the model is quite abstract and can be applied to many systems, it still has limitations. One possible enhancement would be to include the notion of **obligations** [27] to the system. Whenever the system approves a request, it might add a set of constraints that govern exactly how that authorization is going to work. For example, the system might add some notification or operation restrictions (i.e. “you may see this picture, but you have to notify the subject”, “you may see this picture, but you have to delete the picture afterwards” or “you can see this picture, but not share or tag it”). Sticking to the first example, we could define the *wealth* and *requitat* magnitudes as the amount of time the user is authorized to see the given picture.

The simulator also could benefit from future work. Because of its architecture, it would be feasible to apply concurrent programming techniques to parallelize the algorithm. Even though this wouldn’t improve much for the *build entire tree* feature, since its growth is exponential in the number of requests, for linear executions, could allow much bigger scenarios (i.e. with more requests and/or users) to be processed.

Another improvement that could be done for the simulator, from a more practical approach, is to have it output \LaTeX code that, when compiled it would produce the graph corresponding to the input requests defined in section 5.3.1. This would improve

the user experience of the simulator.

BIBLIOGRAPHY

- [1] O'Reilly. <http://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html>.
- [2] International Telecommunication Union. The world in 2014: Facts and figures. 2014.
- [3] Danah M. Boyd and Nicole B. Ellison. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13(1):210–230, 2007.
- [4] Yabing Liu, Krishna P. Gummadi, Balachander Krishnamurthy, and Alan Mislove. Analyzing facebook privacy settings: User expectations vs. reality. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '11, pages 61–70, New York, NY, USA, 2011. ACM.
- [5] K.E. Himma and H.T. Tavani. *The Handbook of Information and Computer Ethics*, chapter Informational Privacy: Concepts, Theories, and Controversies, pages 131–165. Kingfisher leisure guide. Wiley, 2008.
- [6] Department of Defense. *Department of Defense Trusted Computer System Evaluation Criteria*, December 1985. DOD 5200.28-STD (supersedes CSC-STD-001-83).
- [7] Regina Marin, Guillaume Piolle, and Christophe Bidan. Equity-preserving management of privacy conflicts in social network systems. Avenue de la Boulaie, CS 47601 - 35510 Cesson-Sévigné, France, 2014. CIDre, Supélec.
- [8] Quentin Skinner. *Visions of Politics*, volume II. Cambridge University Press, 3rd edition, 2002.
- [9] Jean-Yves Duclos. Horizontal and vertical equity. In Steven N. Durlauf and Lawrence E. Blume, editors, *The New Palgrave Dictionary of Economics*. Palgrave Macmillan, Basingstoke, 2008.
- [10] Office of Equity and Human Rights, City of Portland. <https://www.portlandoregon.gov/oehr/article/449547>.
- [11] Jason Furman. The concept of neutrality in tax policy. In *Tax: Fundamentals in Advance of Reform*, 2008.
- [12] Carlos F. Balbín. *Manual de Derecho Administrativo. Segunda Edición, actualizada y ampliada*. Editorial La Ley.
- [13] Steven Jan, Kara Hanson, and Lilani Kumarayanake. *Economic Analysis for Management and Policy*. Understanding Public Health. Open University Press, September 2005.
- [14] Loveday Penn-Kekana Veloshnee Govender. Gender biases and discrimination: a review of health care interpersonal interactions. 2007.
- [15] Virginia López Casariego y Érica Almeida. Derecho a la salud sin discriminación. *Instituto Nacional contra la Discriminación, la Xenofobia y el Racismo (INADI)*, 2012.

-
- [16] Michael Backes, Birgit Pfitzmann, and Matthias Schunter. A toolkit for managing enterprise privacy policies. In *In Proc. of ESORICS'03, LNCS 2808*, pages 162–180. Springer, 2003.
- [17] Mainack Mondal, Yabing Liu, Bimal Viswanath, Krishna P. Gummadi, and Alan Mislove. Understanding and specifying social access control lists. In *Symposium On Usable Privacy and Security (SOUPS 2014)*, pages 271–283, Menlo Park, CA, July 2014. USENIX Association.
- [18] Ronald Kainda, Ivan Flechais, and AW Roscoe. Security and usability: Analysis and evaluation. In *Availability, Reliability, and Security, 2010. ARES'10 International Conference on*, pages 275–282. IEEE, 2010.
- [19] Timo Jokela. When good things happen to bad products: Where are the benefits of usability in the consumer appliance market? *interactions*, 11(6):28–35, November 2004.
- [20] J.E. Hopcroft, R. Motwani, and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley series in computer science. Addison-Wesley, 2001.
- [21] C. Gini. *Variabilità e mutabilità: contributo allo studio delle distribuzioni e delle relazioni statistiche*. Number pt. 1. Tipogr. di P. Cuppini, 1912.
- [22] M. O. Lorenz. Methods of Measuring the Concentration of Wealth. *Publications of the American Statistical Association, Volume 9, Number 70, p. 209-219*, 9:209–219, June 1905.
- [23] D. Chotikapanich. *Modeling Income Distributions and Lorenz Curves*. Economic Studies in Inequality, Social Exclusion and Well-Being. Springer New York, 2008.
- [24] D.B. Holsinger and W.J. Jacob. *Inequality in Education: Comparative and International Perspectives*. CERC Studies in Comparative Education. Springer Netherlands, 2009.
- [25] Stefan Sackmann, Jens Strüker, and Rafael Accorsi. Personalization in privacy-aware highly dynamic systems. *Commun. ACM*, 49(9):32–38, September 2006.
- [26] Ramprasad Ravichandran, Michael Benisch, PatrickGage Kelley, and NormanM. Sadeh. Capturing social networking privacy preferences. In Ian Goldberg and MikhailJ. Atallah, editors, *Privacy Enhancing Technologies*, volume 5672 of *Lecture Notes in Computer Science*, pages 1–18. Springer Berlin Heidelberg, 2009.
- [27] Manuel Hilty, David Basin, and Alexander Pretschner. On obligations. In SabrinadeCapitani di Vimercati, Paul Syverson, and Dieter Gollmann, editors, *Computer Security – ESORICS 2005*, volume 3679 of *Lecture Notes in Computer Science*, pages 98–117. Springer Berlin Heidelberg, 2005.