Structured Justifications for Binary Aggregation

MSc Thesis *(Afstudeerscriptie)*

written by

Otto de Jong

under the supervision of **Dr. Ronald de Haan**, and submitted to the Examinations Board in partial fulfillment of the requirements for the degree of

MSc in Logic

at the *Universiteit van Amsterdam*.

Date of the public defense:	**Members of the Thesis Committee:**
25 August 2025	Dr. Maria Aloni (Chair)
	Dr. Ronald de Haan (Supervisor)
	Dr. Gregor Behnke
	Dr. Malvin Gattinger

Institute for Logic, Language and Computation

Abstract

In *binary aggregation (with integrity constraints)*, a group of voters votes on a set of issues. Each person votes by submitting a ballot in the form of a binary vector, stating for each issue whether they accept or reject it. An integrity constraint is a propositional formula over the different issues that states which ballots the voters are allowed to submit. A (consistent) aggregation rule then decides on a set of possible outcomes. Like the ballots submitted, each possible outcome is a binary vector, accepting or rejecting every issue while also taking into account the integrity constraint, i.e. not violating it.

Suppose we are in such a setting, where every voter has submitted a ballot and the aggregation rule decided on one or more possible outcomes. Given a set of normative principles that we call *axioms*, one can try to justify the chosen outcome(s). This can be done by showing that collectively accepting this set of axioms forces the outputted outcome under the inputted ballots, no matter the specific aggregation rule.

In an *unstructured justification*, we provide an explanation in the form of a set of specific instances of these axioms that together justify the outcome. In a *structured justification*, we also provide a structured way to read this explanation.

In the first part of this thesis, we will define unstructured and structured justifications for binary aggregation. First, we propose a new definition of *outcome statements* that are used to describe aggregation rules. Thereafter, we define a tableau-based calculus in which we can structure the justifications. In the second part, we encode the framework of binary aggregation in Python, and we develop an algorithm that allows us to generate justifications with the help of a SAT solver and an MUS extraction tool.

Contents

Introduction

Suppose a municipality wants to decide which urban projects they will realize in the coming period. In order to do so, they let the residents vote on the different projects. Suppose there are three different proposed projects, but there is only enough money to realize two of them. Then the municipality could ask the residents to present the following ballot: for each of the three projects, either accept it or reject it, with the extra constraint that they may not accept all three of the projects.

After the votes are counted, the municipality will have to decide which two projects will be realized. Although this example is rather simple, there are cases where this decision is far from straightforward, e.g. there might be tens of projects with dependencies between the different projects. After such a decision is made, it will be made public. Hopefully, most residents will be happy, or at least not very unhappy, about the outcome.

The decision is made by a mechanism that aggregates the ballots that are cast by the residents and then decides on an outcome, in this case: the election of two of the three projects. If the municipality wanted to offer some transparency concerning their decision, they could try and explain the workings of the mechanism that elects the outcome. But a more direct form of transparency would be to directly justify the outcome by explaining why the outcome elected is the only possible outcome, given the ballots submitted and some collectively shared notions of fairness. Such justifications form the main topic of this thesis.

In this thesis, we will define justifications for the framework of *binary aggregation (with integrity constraints)*. We will provide two kinds of justifications: *unstructured* and *structured justifications*.

The main goal of this thesis is twofold. First, we aim to set up a theoretical model in which we can express and construct justifications for the framework of binary aggregation. Thereafter, we want to encode this framework in a computer and develop an algorithm that allows us to generate explainable justifications. Before we can give a more in-depth description, we provide some theoretical context.

Computational Social Choice

Social choice theory is a field of work that analyses settings of collective decision-making, like the one described above. Generally, such a setting consists of a group of voters and a group of candidates or possible outcomes. Every voter voices their opinion on what the outcome should be by submitting a ballot, and some mechanism should then aggregate these opinions and conclude on a set of outcomes. There is a wide variety of different frameworks in which to express these settings formally. Every framework constitutes a subfield of social choice theory; for an extensive overview of this field, we refer to the work of Arrow, Sen and Suzumura [ASS02].

Voting theory is a prominent subfield of social choice theory. In voting theory, there is a group of voters and a group of candidates. Each voter submits a ballot in the form of a preference order over the candidates. These ballots together form a *profile*. A *social choice function* is a function that takes such a profile as an input and then outputs one or more candidates; this outcome is often called the *winner set*. For a detailed description of this framework, we refer to the chapter written by Zwicker [Zwi16] in the *Handbook of Computational Social Choice* by Brandt, Conitzer, Endriss, Lang and Procaccia [Bra+16].

Social choice functions can satisfy certain normative principles, these we call *axioms*. For example, a social choice function satisfies *strategyproofness* if no voter can lie about their preference in order to get a better outcome for themselves; for a formal definition, we also refer to the work of Zwicker [Zwi16]. Arrow [Arr63] proved an important *impossibility theorem*, in which he showed that there is no social choice function that satisfies a certain set of axioms. This formed an early and important work in the paradigm of social choice theory called the *axiomatic method*.

The axiomatic method focuses on the different axioms, e.g. by checking which sets of axioms are unsatisfiable together (thus formulating impossibility theorems) or by proving whether certain social choice functions satisfy certain axioms. This paradigm is also extensively described in the work of Brandt et al. [Bra+16].

This thesis falls within *computational social choice*. This is the field of work that combines social choice theory with methods from computer science; it also considers the computational aspects of the different concepts within social choice theory. The work of Brandt et al. [Bra+16] offers a broad overview of the field.

As stated above, computational social choice combines social choice theory with tools from computer science. One of these tools are SAT solvers. The SAT problem is a fundamental decision problem in computer science. It takes a (propositional) formula as input and checks whether this formula is satisfiable or not. For an extensive review of the problem of satisfiability in computer science, we refer to the work of Biere, Heule, Van Maaren and Walsh [Bie+09]. A SAT solver is a computer program that "solves" the SAT problem; it takes a formula as input and returns as output whether that formula is satisfiable or not.

In computational social choice, a SAT solver is mostly used to check if there exist social choice functions that satisfy certain sets of axioms. In order to do so, both the social choice function and the axioms need to be encoded as one big propositional formula. The use of a SAT solver in different subfields of computational social choice has been researched by a variety of people [LT09; GE11a; BG16; GP17; End20; Klu+20].

Binary Aggregation

In this thesis, we consider the subfield of computational social choice called *binary aggregation (with integrity constraints)*. In binary aggregation, people vote on a set of *issues* by submitting ballots in the form of binary vectors; accepting or rejecting each of the issues. The *integrity constraint* is a propositional formula over the issues that describes which ballots the voters are allowed to submit. An *aggregation rule* should then decide on a set of possible outcomes, electing one or more binary vectors. Each outcome vector also provides a decision for every issue by either accepting or rejecting it. An aggregation rule is *consistent* if all outcome vectors elected by the rule also satisfy the integrity constraint.

The example described at the beginning of this introduction can easily be expressed within this

framework. Let $\{1, \ldots, 100\}$ be the set of residents that will cast a vote, let $X = \{x_1, x_2, x_3\}$ be the set of issues and let $\Gamma = \neg(x_1 \wedge x_2 \wedge x_3)$ be the integrity constraint (stating that no ballot may accept all issues). Suppose the ballots are cast by the voters as seen in the following profile.

$$\begin{aligned} 30 &: (1, 1, 0) \\ 49 &: (1, 0, 1) \\ 21 &: (0, 1, 1) \end{aligned}$$

It is not evident here what would be the fairest outcome; simply picking the majority for each issue will lead to accepting all issues, which will violate the integrity constraint. In this thesis, we will introduce a variety of different axioms that allow us to discuss settings like the one above.

Binary aggregation is a relatively new subfield of computational social choice. A lot of work in this field has been done by Grandi and Endriss [GE10; GE11b; Gra12a; EG14]. They have based their model on the work of Wilson [Wil75] and Dokow and Holzman [DH10], who propose general frameworks for aggregation procedures.

Binary aggregation is closely related to another subfield of computational social choice called *judgment aggregation*, where people vote on formulas instead of issues. For an early work on judgment aggregation, we refer to a paper by List and Pettit [LP02]. For a clear overview of the framework, we refer to the work of Endriss [End16]. Judgment aggregation is a more general framework than voting theory, which Endriss motivates in the aforementioned work by showing how to express a voting theory setting in the framework of judgment aggregation. Binary aggregation has the same expressive power as judgment aggregation, and one can translate aggregation settings back and forth in both frameworks; this is shown by Grandi [Gra12a].

Another related field is *participatory budgeting*. Participatory budgeting considers more applied settings like the example above, where there is a limited budget and people can vote on how they think that budget should be spent. This framework closely ties to judgment aggregation, as is shown by Rey, Endriss and De Haan [REH20]. For a more extensive introduction to participatory budgeting, we refer to the work of Rey [Rey23].

Contribution

Our contribution can be divided into two parts: the construction of a theoretical model on the one hand and the implementation of that model on the other.

First, we discuss the theoretical part. In that part, we define unstructured and structured justifications for binary aggregation. These concepts serve as tools for the transparency in settings of collective decision-making, as is illustrated by the example at the beginning of this introduction. An unstructured justification consists of two parts: a set of collectively accepted normative principles and a set of instances of those axioms that each explain for a specific scenario how an aggregation rule should behave in that scenario. The union of these instances then forms an explanation for why every aggregation rule that satisfies the set of normative principles should accept the prescribed outcome, given a certain input of ballots. A structured justification extends this definition by also showing how to read this explanation; it structures the explanation into a human-readable proof.

In order to define these justifications, we have developed a language of *outcome statements* that allows us to easily express axioms as sets of formulas, where each formula describes how an aggregation rule should behave in a specific scenario. This descriptive language offers a new way to formally describe aggregation rules.

Consequently, we have developed a tableau-based calculus, based on the semantics of these outcome statements, that allows us to prove the unsatisfiability of certain combinations of outcome statements and axiom instances. In this context, a set of outcome statements and axioms is unsatisfiable if there is no aggregation rule that satisfies all of them. We have also proven termination, soundness and completeness of our calculus. Firstly, this calculus offers a new model for reasoning about aggregation rules in binary aggregation. Secondly, it allows us to structure the explanations in unstructured justifications, and thus is an essential part of the definition of structured justifications.

In the second part of this thesis, we show how to encode the framework of binary aggregation in Python, using the earlier defined outcome statements. In order to do so, we encode both aggregation rules and the different binary aggregation axioms in Python. To our knowledge, the framework of binary aggregation has not been encoded in a computer yet.

Finally, we construct an algorithm that first generates a set that closely resembles an unstructured justification and then structures that set in order to obtain a tree that resembles a structured justification. This is done with the help of a SAT solver. To our knowledge, using a SAT solver as a tool for the framework of binary aggregation has not been done before.

Related Work

For voting theory, Boixel and Endriss [BE20] have already defined unstructured justifications and Boixel, Endriss and De Haan [BEH22] have defined structured justifications. Their work serves as a guideline for our model. The main ideas of the definitions are the same, although the adaptation to the framework of binary aggregation sometimes requires us to adopt different approaches. For example, we have completely diverted from their definition of an outcome statement as their definition does not translate to the framework of binary aggregation nicely. As this definition lies at the fundament of our model, our tableau-based calculus has turned out different as well.

Boixel, Endriss and Nardi [BEN22] have shown how to nicely integrate structured justifications for voting theory into an interactive tool that illustrates the workings of the definition by generating different examples for different inputs. We do not present such a tool, but we have developed an algorithm that generates a structured justification if one exists for the given input.

Nardi, Boixel and Endriss [NBE22] have developed a graph-based algorithm that generates unstructured justifications efficiently; also for voting theory.

Grandi and Endriss [GE13] defined a first order language to express axioms within the field of *preference aggregation*. This is an alternative to the propositional language of outcome statements that we will define. Nardi [Nar21] proposes a similar language for axioms in voting theory by expressing axiom instances as formulas.

Cailloux and Endriss [CE16] developed a system to argue about voting rules. There are some resemblances between our language of outcome statements and the language that they define in their paper.

Outline

We end the introduction with a short outline of the thesis.

Chapter 1: This is a preliminary section. First, we give some specific definitions from propositional logic that are relevant for this thesis. Then, we provide preliminary definitions for graph theory,

as they are necessary for our tableau-based calculus. Consequently, we introduce the framework of binary aggregation with all the relevant definitions. We conclude the chapter by introducing several binary aggregation axioms.

Chapter 2: In this chapter we define the outcome statements and their language. We also provide a definition of an axiom instance and an axiom. Finally, we rewrite the axioms provided in the previous chapter in the language of outcome statements.

Chapter 3: Here we introduce the tableau method. First, we give a general introduction to tableaux. Then we introduce our tableau-based calculus. We conclude the chapter by showing termination, soundness and completeness of the newly introduced calculus.

Chapter 4: In this chapter we introduce the main notions of this thesis: unstructured and structured justifications. We provide several examples for both definitions.

Chapter 5: Now that we have concluded the theoretical section, we start with the encoding. In this chapter we show how to encode the framework of binary aggregation. The encoding of an aggregation rule requires the introduction of several extra axioms. We end the chapter by encoding the axioms introduced in the first chapter and rewritten in the second chapter.

Chapter 6: In the final chapter of the thesis we show the algorithm that we have constructed. It consists of two parts: the first part generates a set that resembles an unstructured justification. The second part structures this set by generating a tree that resembles a structured justification. We show the workings of both parts of the algorithm through several examples.

Part I

Theoretical Part

Chapter 1

Preliminaries

In this chapter we will provide the necessary context for this thesis. First, we consider some relevant definitions from propositional logic. Thereafter, we provide some basic definitions of graph theory. Finally, we will introduce the framework of binary aggregation and state some relevant axioms within that framework.

1.1 Propositional Logic

We will assume that the reader is familiar with propositional logic; if not, we refer to the work of Mendelson [Men09]. Still, we provide several specific definitions.

For a set of *propositional variables* X, let $\mathcal{L}(X)$ denote the set of *propositional formulas* over X, i.e. propositional formulas that contain only variables from the set X.

A propositional formula that is a disjunction of literals is called a *clause*. A propositional formula φ is in *conjunctive normal form (CNF)* if it is the conjunction over a set of clauses; in this case, we call φ a CNF-formula. A formula φ is in *disjunctive normal form (DNF)* if it is a disjunction over a set of formulas that are themselves conjunctions of literals. Such a formula, we call a DNF-formula. We present a lemma concerning these types of formulas.

Lemma 1.1. *For every propositional formula φ, we can find a DNF-formula that is equivalent to φ. The same holds for CNF-formulas.*

For the proof of this theorem, we refer to the work of Howson [How05].

Definition 1.1. *We define the **length** $L(\varphi)$ of a propositional formula φ recursively:*

- $L(\bot) = L(\top) := 0$*;*
- $L(x) := 1$*, for a propositional variable x;*
- $L(\neg\varphi) := 1 + L(\varphi)$*, for a propositional formula φ;*
- $L(\varphi \otimes \psi) := 1 + L(\varphi) + L(\psi)$*, for two propositional formulas φ and ψ and any operator $\otimes \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.*

1.2 Graph Theory

In Chapter 3, we will define a tableau-based calculus with which we construct structured justifications for binary aggregation. As graph theory offers the underlying structure for our tableaux, we provide the necessary definitions in the following section. For a more thorough discussion of graph theory, we refer to the work of Tutte [Tut01].

A *graph* is a tuple $G = (V, E)$ where V is a set of *nodes* or *vertices* and E is a binary relation on V. For two nodes $v, w \in V$, an element $(v, w) \in E$ is called an *edge*. For two vertices $v, w \in V$, the edge $(v, w) \in E$ is an *outgoing edge* for v and an *incoming edge* for w. The *in-degree* of a vertex is the amount of incoming edges for that vertex and the *out-degree* is the amount of outgoing edges.

A *path* in a graph $G = (V, E)$ is a sequence of vertices $(v_1, v_2, v_3, \ldots)$ such that $(v_i, v_{i+1}) \in E$ for every index i. A *cycle* is a path $(v_1, \ldots, v_n)$ with $v_1 = v_n$. We call a graph *acyclic* if there are no cycles. We call a graph *connected* if there is a path between every two vertices in V.

A *(directed) tree* is a connected acyclic graph. In a tree, a *root node* is a vertex that has an in-degree of 0. A node with an out-degree of 0 is called a *leaf node*, and if we have that $(v, w), (v, w') \in E$, we call w and w' the *children* of v. Every path from a root node to a leaf node $v \in V$ is called a *branch*; we then say that the branch *ends* in v. A tree is infinite if it contains infinitely many nodes.

To conclude the section, we make the following remark, which will be helpful later on.

Remark 1.1. *A tree where every node only has finitely many children is infinite if and only if it contains an infinite branch.*

1.3 Binary Aggregation

In this section, we will provide the basic definitions for the framework of binary aggregation (with integrity constraints). The following definitions are based on the work of Grandi and Endriss [GE11b; Gra12b].

Let $\mathcal{I} = \{1, \ldots, m\}$ be a finite set of *issues* that need to be decided on and let $X = \{x_1, \ldots, x_m\}$ denote the set of propositional variables corresponding to the issues in $\mathcal{I}$. Let $X^* = X \cup \{\neg x \mid x \in X\}$ denote the set of *literals* on X. A literal $\ell \in X^*$ is *positive* if $\ell = x$ for some $x \in X$ and *negative* if $l = \neg x$ for some $x \in X$.

A vector $B \in \{0, 1\}^m$ is called a *ballot*, and it provides a binary decision for each issue: yes (denoted by a 1) or no (denoted by a 0). Such a ballot can also be viewed as a truth assignment on the propositional variables in X, making x_j true if there is a 1 on the j-th position of B, and false if it is a 0. This can naturally be extended to any propositional formula $\varphi \in \mathcal{L}(X)$: we say that a ballot B *satisfies* such a propositional formula φ if the truth assignment induced by the ballot makes φ true. We then write $B \vDash \varphi$.

Example 1.1. *For $m = 3$, we say that the ballot $B = (1, 0, 1)$ makes x_1 true, while it makes x_2 false. We may also write $B \vDash \neg(x_1 \wedge x_2 \wedge x_3)$, as the variable x_2 is false under the truth assignment induced by the ballot B.*

An *integrity constraint* $\Gamma \in \mathcal{L}(X)$ is a satisfiable propositional formula with variables from X that describes which ballots are allowed. Let $\mathrm{Mod}(\Gamma) \subseteq \{0, 1\}^m$ be the set of ballots that satisfy Γ; we call those the *rational* ballots.

Example 1.2. *For $m = 3$ and $\Gamma = (x_1 \to x_2) \wedge \neg x_3$, we get that $Mod(\Gamma) = \{(1,1,0),(0,1,0),(0,0,0)\}$.*

Let n be the number of *voters* that vote on the issues in $\mathcal{I}$ (or X). A multiset $\mathbf{B}$ with $|\mathbf{B}| = k$ consisting of rational ballots is called a *profile* of size k. We denote the set of all possible profiles of size k by $\mathrm{Mod}(\Gamma)^k$ and the set of all possible profiles (with at most n voters) by $\mathrm{Mod}(\Gamma)^+ := \bigcup_{1 \leq k \leq n} \mathrm{Mod}(\Gamma)^k$.

Example 1.3. *Again, let $m = 3$ and $\Gamma = (x_1 \to x_2) \wedge \neg x_3$. Then for $\mathbf{B} = \{(1,1,0),(0,1,0)\}$, we have that $\mathbf{B} \in Mod(\Gamma)^2$. We also have that $\mathbf{B} \in Mod(\Gamma)^+$.*

Remark 1.2. *Note here that our framework is anonymous, i.e. we do not consider which voter submitted which ballot, we only consider what ballots are submitted. This is due to the fact that anonymity is a prerequisite condition for almost every application of binary aggregation (and democratic elections in general). Additionally, it simplifies our framework because we can consider profiles as multisets instead of ordered sequences of ballots.*

An *aggregation rule* is a function $F : \mathrm{Mod}(\Gamma)^+ \to 2^{\{0,1\}^m} \setminus \{\varnothing\}$, mapping any possible profile of rational ballots to a nonempty subset of *outcome vectors*; or simply *outcomes*.

Notation 1.2. *Sometimes we denote an aggregation rule by F, where it is assumed that $F : Mod(\Gamma)^+ \to 2^{\{0,1\}^m} \setminus \{\varnothing\}$.*

Note here that a profile is a multiset, while the set of outcomes, sometimes called the *outcome set*, is a set (without duplicates); this set describes which outcomes are elected by the aggregation rule. An aggregation rule F is called *consistent* if $F(\mathbf{B}) \subseteq \mathrm{Mod}(\Gamma)$ for any $\mathbf{B} \in \mathrm{Mod}(\Gamma)^+$, i.e. all outcome vectors in $F(\mathbf{B})$ satisfy the constraint Γ, where we note that we may consider any outcome in $F(\mathbf{B})$ as a ballot (a binary vector with dimension m). An outcome vector $v \in \{0,1\}^m$ is called *consistent* if it satisfies the integrity constraint, i.e. $v \vDash \Gamma$. In that case, we have that $v \in \mathrm{Mod}(\Gamma)$.

Example 1.4. *Let $m = 3$, $\Gamma = x_1 \vee x_2$ and let F be an aggregation rule such that there is a profile $\mathbf{B} \in Mod(\Gamma)^+$ with $F(\mathbf{B}) = \{(0,0,1),(1,1,1)\}$. Then F is not consistent as $(0,0,1) \nvDash x_1 \vee x_2$. The rule F' that maps any profile $\mathbf{B} \in Mod(\Gamma)^+$ to the singleton $\{(1,0,0)\}$ is consistent.*

Convention 1. *For the rest of this thesis, let n be the number of voters, let $\mathcal{I}$ be the set of issues with $|\mathcal{I}| = m$, let X be the corresponding set of variables with literal set X^* and let $\Gamma \in \mathcal{L}(X)$ be the integrity constraint. Whenever it is not specified, we assume that $\Gamma = \top$.*

1.4 Binary Aggregation Axioms

In binary aggregation, axioms are properties that can be satisfied or violated by aggregation rules. The axioms presented in the following section are primarily based on notions of fairness and should all be easy to accept as reasonable principles. The following axiom definitions are based on the work of Lang, Pigozzi, Slavkovik, Van der Torre and Vesic [Lan+17] and the work of Grossi and Pigozzi [GP22]. Both have defined axioms for judgment aggregation. We define adaptations of these axioms for binary aggregation.

Faithfulness

Whenever we consider a profile $\mathbf{B} \in \mathrm{Mod}(\Gamma)^1$, we know it is of the form $\mathbf{B} = \{B\}$, where B is some rational ballot in $\mathrm{Mod}(\Gamma)$. The first axiom is based on the idea that in a situation with a single

voter, the aggregation rule should adopt the ballot submitted by this single voter as the unique outcome.

Definition 1.2 (Faithfulness)**.** *We say that an aggregation rule F is **faithful** if for every profile $\mathbf{B} \in Mod(\Gamma)^1$, we have that $F(\mathbf{B}) = \{B\}$, where $\mathbf{B} = \{B\}$.*

Homogeneity

The next axiom considers the scenario where a profile consists of multiple copies of some other profile. Before we can state the axiom, we first need to define addition for profiles.

Definition 1.3. *For two profiles $\mathbf{B}, \mathbf{B}' \in Mod(\Gamma)^+$, we define the **sum** of the profiles as $\mathbf{B} + \mathbf{B}' := \mathbf{B} \uplus \mathbf{B}'$, where $\uplus$ is the multiset sum, which adds all the elements of $\mathbf{B}$ together with all the elements of $\mathbf{B}'$ in the natural way. For an integer $k \geq 1$ and a profile $\mathbf{B}$, we define $k\mathbf{B} := \underbrace{\mathbf{B} + \cdots + \mathbf{B}}_{k \text{ times}}$.*

Example 1.5. *For $m = 2$, $\mathbf{B} = \{(1,1),(1,1),(1,0),(0,0)\}, \mathbf{B}' = \{(1,1),(1,0)\}$ and $\mathbf{B}'' = \{(1,1),(0,0)\}$, we have that $\mathbf{B} = \mathbf{B}' + \mathbf{B}''$. For a profile $\mathbf{B}''' = \{(1,1),(1,1),(0,0),(0,0)\}$, we have that $\mathbf{B}''' = 2\mathbf{B}''$. Note here that profiles are multisets, so the order does not matter.*

Definition 1.4 (Homogeneity)**.** *We say that an aggregation rule F is **homogeneous** if for any profile $\mathbf{B} \in Mod(\Gamma)^+$ and any integer $k \geq 1$ such that $k\mathbf{B} \in Mod(\Gamma)^+$, we have that $F(\mathbf{B}) = F(k\mathbf{B})$.*

Example 1.6. *Let $n = 4$, $m = 2$ and let F be a homogeneous aggregation rule, then we should have that $F(\{(1,1),(0,0)\}) = F(\{(1,1),(1,1),(0,0),(0,0)\})$.*

Unanimity

The following two axioms consider the situation in which every ballot in the profile agrees on an issue, either by accepting it or rejecting it unanimously.

Definition 1.5 (Weak Unanimity)**.** *We say that an aggregation rule F is **weakly unanimous** if for every profile $\mathbf{B} \in Mod(\Gamma)^+$ and any literal $\ell \in X^*$, if we have that for all $B \in \mathbf{B}$ it holds that $B \vDash \ell$, then there is an outcome $v \in F(\mathbf{B})$ such that $v \vDash \ell$.*

Definition 1.6 (Strong Unanimity)**.** *We say that an aggregation rule F is **strongly unanimous** if for every profile $\mathbf{B} \in Mod(\Gamma)^+$ and any literal $\ell \in X^*$, if we have that for all $B \in \mathbf{B}$ it holds that $B \vDash \ell$, then for every outcome $v \in F(\mathbf{B})$ we have that $v \vDash \ell$.*

Example 1.7. *For $m = 2$, some aggregation rule F and a profile $\mathbf{B} = \{(1,0),(1,1)\}$, if $F(\mathbf{B}) = \{(1,1),(0,1)\}$, it is not strongly unanimous, as not every outcome vector satisfies x_1, while every ballot in $\mathbf{B}$ does satisfy x_1. It could however still be weakly unanimous, as there is an outcome in $F(\mathbf{B})$ that satisfies x_1; this depends on what F elects as outcome sets for the other profiles in $Mod(\Gamma)^+$.*

Reinforcement

Reinforcement is a slightly more intricate axiom. Like homogeneity, it is concerned with the composition of profiles.

Definition 1.7 (Reinforcement)**.** *We say that an aggregation rule F satisfies **reinforcement** if for every three profiles $\mathbf{B}, \mathbf{B}', \mathbf{B}'' \in Mod(\Gamma)^+$ where $\mathbf{B} = \mathbf{B}' + \mathbf{B}''$ and $F(\mathbf{B}') \cap F(\mathbf{B}'') \neq \varnothing$, we have that $F(\mathbf{B}) = F(\mathbf{B}') \cap F(\mathbf{B}'')$.*

Example 1.8. *For* $m = 2$, $\mathbf{B} = \{(1,1),(1,1),(1,0),(0,0)\}, \mathbf{B}' = \{(1,1),(1,0)\}$ *and* $\mathbf{B}'' = \{(1,1),(0,0)\}$, *we have that* $\mathbf{B} = \mathbf{B}' + \mathbf{B}''$, *as seen in Example 1.5. Now suppose that for some aggregation rule* F, *we have that* $F(\mathbf{B}') = \{(1,1),(1,0)\}$ *and* $F(\mathbf{B}'') = \{(1,1)\}$, *then if* F *satisfies reinforcement, it should hold that* $F(\mathbf{B}) = \{(1,1)\}$, *as that is the intersection of the two outcome sets* $F(\mathbf{B}')$ *and* $F(\mathbf{B}'')$.

Monotonicity

The following axiom formalizes the idea that if some literal is elected under some profile, then it should also be elected under a profile in which even more ballots satisfy that literal. Before we state it, we need to properly define this scenario.

Definition 1.8. *For two profiles* $\mathbf{B}, \mathbf{B}' \in Mod(\Gamma)^+$, *we say that* $\mathbf{B}'$ *is an* ℓ***-improvement*** *of* $\mathbf{B}$ *if there are ballots* $B \in \mathbf{B}$ *and* $B' \in \mathbf{B}'$ *such that* $\mathbf{B} \setminus \{B\} = \mathbf{B}' \setminus \{B'\}$ *where* B *and* B' *are the same ballot except for the fact that* $B' \vDash \ell$ *while* $B \nvDash \ell$.

Example 1.9. *For* $m = 2$, *let* $\mathbf{B} = \{(1,0),(0,1),(1,1)\}$ *and let* $\mathbf{B}' = \{(0,0),(0,1),(1,1)\}$. *Then* $\mathbf{B}'$ *is an* $\neg x_1$*-improvement of* $\mathbf{B}$, *as for* $B = (1,0)$ *and* $B' = (0,0)$, *we have that* $\mathbf{B} \setminus \{B\} = \mathbf{B}' \setminus \{B'\}$ *where* B *and* B' *are the same ballot except for the fact that* $B' \vDash \neg x_1$ *while* $B \vDash x_1$, *so* $B \nvDash \neg x_1$.

Now we are ready to state the axiom.

Definition 1.9 (Monotonicity)**.** *We say that an aggregation rule* F *is* ***monotonic*** *if for every two profiles* $\mathbf{B}, \mathbf{B}' \in Mod(\Gamma)^+$ *such that* $\mathbf{B}'$ *is an* ℓ*-improvement of* $\mathbf{B}$, *we have that if for every outcome* $v \in F(\mathbf{B})$ *it holds that* $v \vDash \ell$, *we also have that for every outcome* $v' \in F(\mathbf{B}')$ *it holds that* $v' \vDash \ell$.

Example 1.10. *In Example 1.9 above, the profile* $\mathbf{B}'$ *is an* $\neg x_1$*-improvement of* $\mathbf{B}$. *For an aggregation rule* F *that satisfies monotonicity, suppose that* $F(\mathbf{B}) = \{(0,1)\}$, *then it should also hold that every outcome vector in* $F(\mathbf{B}')$ *should reject* x_1 *(have a 0 on the first coordinate), so* $F(\mathbf{B}') = \{(0,1)\}$ *would do, or any other outcome set in which every outcome rejects the first issue.*

Majority-preservation

The next axiom states the idea that an aggregation rule should elect the outcomes that are agreed upon by the majority of the ballots in the profile, e.g. if a majority of the ballots accepts the first issue, then all outcomes should also accept the first issue. In order to state it formally, we require some prerequisite definitions.

Definition 1.10. *For a profile* $\mathbf{B} \in Mod(\Gamma)^+$ *and a formula* $\varphi \in \mathcal{L}(X)$, *we define* $N(\mathbf{B}, \varphi)$ *to be the number of ballots in* $\mathbf{B}$ *that make* φ *true. Formally, we get* $N(\mathbf{B}, \varphi) := |\{B \in \mathbf{B} \mid B \vDash \varphi\}|$.

Example 1.11. *Let* $n = 4$, $m = 3$ *and consider the profile* $\mathbf{B} = \{(1,0),(1,1),(0,1),(0,0)\}$. *Then we find that* $N(\mathbf{B}, x_1) = 2$, *while* $N(\mathbf{B}, x_1 \vee x_2) = 3$.

Definition 1.11. *We define the* ***majority set*** *to be the set of all literals in* X^* *that are satisfied by a strict majority of the ballot:* $m(\mathbf{B}) := \{\ell \in X^* \mid N(\mathbf{B}, \ell) > \frac{|\mathbf{B}|}{2}\}$.

A consistent outcome vector $v \in Mod(\Gamma)$ *is a* ***majority extension*** *of* $m(\mathbf{B})$ *if it satisfies all literals in* $m(\mathbf{B})$. *The set of majority extensions of* $m(\mathbf{B})$ *is then defined as follows:*

$$ext(m(\mathbf{B})) := \{v \in \{0,1\}^m \mid v \vDash \Gamma \text{ and } \forall \ell \in m(\mathbf{B}) : v \vDash \ell\}.$$

We say that $\mathbf{B}$ *is* ***majority-consistent*** *if* $ext(m(\mathbf{B})) \neq \varnothing$ *, i.e. there is a vector satisfying both the literals in* $ext(m(\mathbf{B}))$ *and* Γ*.*

Example 1.12. *Let* $n = 4$*,* $m = 3$ *and let* $\mathbf{B} = \{(1,1,1),(1,0,1),(1,0,0),(0,0,0)\}$*. Then* $m(\mathbf{B}) = \{x_1, \neg x_2\}$ *as both literals are satisfied by three ballots in* $\mathbf{B}$*; a strict majority. Then we find that* $ext(m(\mathbf{B})) = \{(1,0,0),(1,0,1)\}$ *(assuming that* $\Gamma \equiv \top$*).*

Example 1.13. *Again, consider the example above but now let* $\Gamma = x_2 \vee x_3$*. Then the vector* $(1,0,0)$ *is no longer a consistent outcome, so we would have that* $ext(m(\mathbf{B})) = \{(1,0,1)\}$*.*

Finally, we show an example of a profile that is not majority-consistent. This is an adaptation of a famous paradox in judgment aggregation. The scenario in the example below is referred to as the *doctrinal paradox*; it is described in the work of Kornhauser and Sager [KS93].

Example 1.14. *Let* $n = 3$*,* $m = 3$*,* $\Gamma = \neg(x_1 \wedge x_2 \wedge x_3)$ *and let* $\mathbf{B} = \{(1,1,0),(1,0,1),(0,1,1)\}$*. We then find that* $m(\mathbf{B}) = \{x_1, x_2, x_3\}$ *as all issues are accepted by two of the three ballots. However, there is no outcome that satisfies both* Γ *and all the literals in* $m(\mathbf{B})$*, so* $ext(m(\mathbf{B})) = \varnothing$*. We conclude that* $\mathbf{B}$ *is not majority-consistent.*

Now we are ready to state the axiom.

Definition 1.12 (Majority-preservation)**.** *We say that an aggregation rule* F *is* ***majority-preserving*** *if* $F(\mathbf{B}) = ext(m(\mathbf{B}))$ *whenever* $\mathbf{B} \in Mod(\Gamma)^+$ *is majority-consistent.*

Note here that an aggregation rule that satisfies majority-preservation will never output an inconsistent outcome vector whenever the input profile is majority-consistent. We have defined it in this way as we will only consider consistent aggregation rules in our final model.

Example 1.15. *Let* $n = 4$*,* $m = 3$*,* $\Gamma = \top$ *and* F *be an aggregation rule that is majority-preserving. Then for* $\mathbf{B} = \{(1,1,1),(1,0,1),(1,0,0),(0,0,0)\}$*, it should hold that* $F(\mathbf{B}) = \{(1,0,0),(1,0,1)\}$*.*

Cancellation

The final axiom we introduce is cancellation. It considers the scenario where every issue gets the exact same votes in some profile. In that scenario, the cancellation axiom proposes that the outcome set should contain every consistent outcome vector.

Definition 1.13 (Cancellation)**.** *An aggregation rule* F *satisfies cancellation if* $Mod(\Gamma) \subseteq F(\mathbf{B})$ *for every profile* $\mathbf{B} \in Mod(\Gamma)^+$ *such that* $N(\mathbf{B}, x) = N(\mathbf{B}, \neg x)$ *for every* $x \in X$*.*

Note here that $N(\mathbf{B}, x) = N(\mathbf{B}, \neg x)$ implies that $N(\mathbf{B}, x) = N(\mathbf{B}, \neg x) = \frac{1}{2}n$. Then we find that $m(\mathbf{B}) = \varnothing$ so $\text{ext}(m(\mathbf{B})) = \text{Mod}(\Gamma)$.

Example 1.16. *Let* $n = 4$*,* $m = 3$*,* $\Gamma = x_1 \vee x_2$ *and* F *be an aggregation rule that satisfies cancellation. Then for* $\mathbf{B} = \{(0,1,1),(1,0,1),(1,0,0),(0,1,0)\}$*, it should hold that* $F(\mathbf{B}) \supseteq \{(0,1,0),(1,0,0),(1,1,0),(0,1,1),(1,0,1),(1,1,1)\}$*.*

Remark 1.3. *It is important to note here that cancellation is implied by majority-preservation. This follows from the fact that whenever we have a profile* $\mathbf{B}$ *as described above,* $ext(m(\mathbf{B})) = Mod(\Gamma)$*.*

However, the majority-preservation axiom is very strong, and it not always desirable to adopt it as a property of an aggregation rule. Sometimes, it might be better to offer a weaker alternative axiom like cancellation, as will become clear later in this thesis.

Chapter 2

Expressing Axioms in $\mathcal{L}(O)$

In Section 1.4, we have introduced several binary aggregation axioms. All of them are denoted as conditions in a somewhat formal language; using universal and existential quantifiers to describe what the outcome of an aggregation rule should look like. In this chapter, we introduce a propositional language that allows us to rewrite the axioms using propositional formulas over a set of *outcome statements* as propositional variables. In order to do so, we introduce *axiom instances*. Every *axiom* is then simply a set of axiom instances, each of them describing what the aggregation rule should behave like in a specific scenario. In the final section of this chapter, we define a set of these formulas (each corresponding to an axiom instance) for each axiom, and show that an aggregation rule satisfies this set of formulas if and only if it satisfies the axiom.

2.1 Outcome Statements

In the previous chapter, we were describing the possible outcome sets of an aggregation rule through the satisfaction of axioms. In the following section, we define outcome statements, which will help us reason more directly about the outcome sets of aggregation rules.

The idea of using outcome statements is based on the work of Boixel, Endriss and De Haan [BEH22]. They define outcome statements for voting theory that narrow down the set of possible outcomes given a certain input profile. In this thesis, we introduce a different type of outcome statement, which considers what formulas the outcomes should satisfy rather than what the possible outcome should be. This is a consequence of the fact that for binary aggregation, there are $2^{2^m} - 1$ different possible (nonempty) outcome sets (where m is the number of issues): every binary vector is either in the outcome or not. As a solution, we introduce an outcome statement that does not directly describe the set of possible outcomes, but rather states what formulas are satisfied by the set of outcomes, which allows us to write more compact outcome statements.

Definition 2.1. *Let $\mathbf{B} \in Mod(\Gamma)^+$ be a profile, let $\varphi \in \mathcal{L}(X)$ be a formula and let F be an aggregation rule. A **universal outcome statement** is a tuple of the form $[\![\mathbf{B}, \varphi]\!]$. We say that an aggregation rule F **satisfies** an outcome statement $[\![\mathbf{B}, \varphi]\!]$ if every outcome in $F(\mathbf{B})$ satisfies the formula φ, where we recall that an outcome can be seen as a valuation on X. Formally, we define the following:*

$$F \vDash [\![\mathbf{B}, \varphi]\!] \iff \forall v \in F(\mathbf{B}) : v \vDash \varphi.$$

Example 2.1. *Let $m = 3$, let $\mathbf{B} \in Mod(\Gamma)^+$ and let F be some aggregation rule such that $F(\mathbf{B}) =$*

$\{(1,0,1),(1,1,0),(1,0,0)\}$. *Then it holds that* $F \vDash [\![\mathbf{B}, x_1]\!]$ *as every outcome* $v \in F(\mathbf{B})$ *satisfies* x_1. *It also holds that* $F \vDash [\![\mathbf{B}, x_1 \vee x_2 \vee x_3]\!]$, *but we have that* $F \nvDash [\![\mathbf{B}, x_2 \vee x_3]\!]$, *as* $(1,0,0) \in F(\mathbf{B})$ *while* $(1,0,0) \nvDash x_2 \vee x_3$.

Definition 2.2. *Building on Definition 2.1, we construct the set of all universal outcome statements. There is a universal outcome statement for every profile in* $Mod(\Gamma)^+$ *and formula in* $\mathcal{L}(X)$*:*

$$O := \{[\![\mathbf{B}, \varphi]\!] \mid \mathbf{B} \in Mod(\Gamma)^+ \text{ and } \varphi \in \mathcal{L}(X)\}.$$

Now we define the propositional language $\mathcal{L}(O)$ over the set of universal outcome statements O.

Definition 2.3. *Let* $[\![\mathbf{B}, \varphi]\!]$ *be a universal outcome statement with* $\mathbf{B} \in Mod(\Gamma)^+$ *and* $\varphi \in \mathcal{L}(X)$*; then we have that* $[\![\mathbf{B}, \varphi]\!] \in O$. *Let* ψ *and* ρ *be propositional formulas with variables from* O, *so* $\psi, \rho \in \mathcal{L}(O)$. *We define the following semantics.*

- $F \vDash [\![\mathbf{B}, \varphi]\!]$ *is defined in Definition 2.1,*
- $F \vDash \neg\psi$ *if and only if* $F \nvDash \psi$,
- $F \vDash \psi \wedge \rho$ *if and only if* $F \vDash \psi$ *and* $F \vDash \rho$,
- $F \vDash \psi \vee \rho$ *if and only if* $F \vDash \psi$ *or* $F \vDash \rho$,
- $F \vDash \psi \to \rho$ *if and only if* $F \nvDash \psi$ *or* $F \vDash \rho$,
- $F \vDash \top$ *always,*
- $F \vDash \bot$ *never.*

Note here that for every universal outcome statement of the form $[\![\mathbf{B}, \bot]\!] \in O$, we have that $[\![\mathbf{B}, \bot]\!] \equiv \bot$, as there is no aggregation rule such that every vector in $F(\mathbf{B})$ satisfies $\bot$.

Notation 2.1. *From now on, we will use tuples of the form* $[\![\mathbf{B}, \varphi]\!]$ *to denote both universal outcome statements in* O *and propositional variables in* $\mathcal{L}(O)$.

Example 2.2. *For* $\mathbf{B}, \mathbf{B}' \in Mod(\Gamma)^+$ *and* $\varphi, \psi \in \mathcal{L}(X)$, *the formula* $[\![\mathbf{B}, \varphi]\!] \to [\![\mathbf{B}', \psi]\!]$ *is in* $\mathcal{L}(O)$. *For an aggregation rule* F *with* $F \vDash [\![\mathbf{B}, \varphi]\!] \to [\![\mathbf{B}', \psi]\!]$, *the following holds:*

$$(\exists v \in F(\mathbf{B}) : v \nvDash \varphi) \text{ or } (\forall v' \in F(\mathbf{B}') : v' \vDash \psi).$$

This is equivalent to the following statement:

$$\text{if } (\forall v \in F(\mathbf{B}) : v \vDash \varphi) \text{ then } (\forall v' \in F(\mathbf{B}') : v' \vDash \psi);$$

if all outcomes under input $\mathbf{B}$ *satisfy* φ, *then all outcomes under input* $\mathbf{B}'$ *should satisfy* ψ.

Upon closer inspection of the universal outcome statement and its negation, we define a dual notion which will simplify our model later on.

Definition 2.4. *Let* $[\![\mathbf{B}, \varphi]\!]$ *be some universal outcome statement. We define an* ***existential outcome statement*** *as follows:*

$$\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle := \neg[\![\mathbf{B}, \neg\varphi]\!].$$

The meaning of the existential outcome statement becomes clear in the following lemma.

Lemma 2.2. *Let $\mathbf{B} \in Mod(\Gamma)^+$, $\varphi \in \mathcal{L}(X)$ and F be an aggregation rule, then we find the following:*

$$F \vDash \langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle \iff \exists v \in F(\mathbf{B}) : v \vDash \varphi.$$

Proof. First, suppose that $F \vDash \langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle$. This means that $F \vDash \neg[\![\mathbf{B}, \neg\varphi]\!]$, so $F \nvDash [\![\mathbf{B}, \neg\varphi]\!]$. Then we find that there exists an outcome vector $v \in F(\mathbf{B})$ such that $v \nvDash \neg\varphi$, but this is equivalent to saying that $v \vDash \varphi$, so we find that there exists an outcome vector $v \in F(\mathbf{B})$ for which $v \vDash \varphi$.

Conversely, suppose that there is an outcome vector $v \in F(\mathbf{B})$ such that $v \vDash \varphi$, then we find that $v \nvDash \neg\varphi$, so not every outcome vector in $F(\mathbf{B})$ makes $\neg\varphi$ true, so by definition we find that $F \nvDash [\![\mathbf{B}, \neg\varphi]\!]$, so $F \vDash \neg[\![\mathbf{B}, \neg\varphi]\!]$. We conclude that $F \vDash \langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle$. □

Example 2.3. *Let $m = 2$, $\mathbf{B} \in Mod(\Gamma)^+$ and F be some aggregation rule such that $F(\mathbf{B}) = \{(1,0),(0,1)\}$. Then we have that $F \vDash \langle\!\langle \mathbf{B}, x_1 \rangle\!\rangle$ and $F \vDash \langle\!\langle \mathbf{B}, \neg x_1 \rangle\!\rangle$ while $F \nvDash [\![\mathbf{B}, x_1]\!]$ and $F \nvDash [\![\mathbf{B}, \neg x_1]\!]$.*

Because an aggregation rule can never output an empty set by definition, we find the following lemma, which shows that the universal outcome statement implies the existential one.

Lemma 2.3. *Let $\mathbf{B} \in Mod(\Gamma)^+$, $\varphi \in \mathcal{L}(X)$ and F be some aggregation rule, then the following holds:*

$$\text{if } F \vDash [\![\mathbf{B}, \varphi]\!] \text{ then } F \vDash \langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle.$$

Proof. Because $F(\mathbf{B}) \neq \varnothing$ by definition, if all outcomes in $F(\mathbf{B})$ satisfy a formula, then there is at least one outcome in $F(\mathbf{B})$ that satisfies it. □

We conclude the section with an observation about the set of all existential and universal outcome statements, which will become useful later on.

Definition 2.5. *We define the set of all outcome statements as follows:*

$$\mathbb{S} := O \cup \{\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle \mid \mathbf{B} \in Mod(\Gamma)^+ \text{ and } \varphi \in \mathcal{L}(X)\}.$$

Elements of this set are either universal or existential outcome statements; they are denoted by $s \in \mathbb{S}$. From now on, we simply refer to them as ***outcome statements****. We denote a set of outcome statements by S.*

Definition 2.6. *Let $L(O) = O \cup \{\neg[\![\mathbf{B}, \varphi]\!] \mid [\![\mathbf{B}, \varphi]\!] \in O\}$ be the set of literals in the language $\mathcal{L}(O)$.*

In the following lemma, we show the relation between the literals in $L(O)$ and outcome statements.

Lemma 2.4. *Every outcome statement is in $L(O)$. Additionally, for every element in $L(O)$, there exists an outcome statement that is equivalent to this element.*

Proof. It is immediately clear that any universal outcome statement is in $L(O)$. Now consider an existential outcome statement $\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle$. By definition, it holds that $\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle = \neg[\![\mathbf{B}, \neg\varphi]\!]$. As we have that $\neg[\![\mathbf{B}, \neg\varphi]\!] \in L(O)$ by definition of $L(O)$, we are done.

Now we prove the second claim. For every universal outcome statement in $L(O)$ it is immediately clear that there exists an equivalent outcome statement, namely the universal outcome statement itself. Now consider a negative literal $\neg[\![\mathbf{B}, \varphi]\!] \in L(O)$. Then we find that $\neg[\![\mathbf{B}, \varphi]\!] \equiv \langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle$, because $\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle = \neg[\![\mathbf{B}, \neg\neg\varphi]\!]$. This concludes the proof of the second claim. □

2.2 Axioms and Axiom Instances in $\mathcal{L}(O)$

The following definitions are inspired by Section 2.2 of the paper by Boixel and Endriss [BE20]. Building on the three general requirements they offer for axiom instances, we provide a definition of an axiom instance for binary aggregation.

Grandi and Endriss [GE13] have defined a first order language to express axioms within preference aggregation. In this section, we will define a propositional language to express the binary aggregation axioms.

Intuitively, an axiom is a normative principle describing a specific property of an aggregation rule. We have introduced several examples of such axioms in Section 1.4. The propositional language in Section 2.1 allows us to easily encode these axioms as sets of formulas. In this section, we describe this alternative way to denote axioms. We view an axiom as a set of axiom instances that describe how an aggregation rule should behave in a specific scenario, i.e. for one or more specific profiles. The axiom itself is then simply the union of all these local descriptions, which gives us a full description of the behavior of an aggregation rule. This idea is encapsulated in the following definition.

Definition 2.7. *An **axiom instance** A is a propositional formula with variables from O, so $A \in \mathcal{L}(O)$. An **axiom** $\mathcal{A}$ is a set of axiom instances. For an aggregation rule F and an axiom instance $A \in \mathcal{L}(O)$, we say that F **satisfies** A if $F \vDash A$. Consequently, we say that F **satisfies** an axiom $\mathcal{A}$ if $F \vDash A$ for all $A \in \mathcal{A}$; by abuse of notation, we also write $F \vDash \mathcal{A}$.*

As discussed earlier, an axiom instance A of an axiom $\mathcal{A}$ is a specific instantiation of that axiom, describing for a particular set of profiles (related through the axiom) how the aggregation rule should elect the outcome.

Example 2.4. *As an example of a very basic axiom, suppose we have the axiom that states that for every aggregation rule, every outcome under any profile should accept the first issue, x_1. Then an axiom instance of this axiom would be that for some specific profile, every outcome of the aggregation rule under that input should accept x_1. We might encode this axiom as the following set:*

$$\mathcal{A} = \{[\![\mathbf{B}, x_1]\!] \mid \mathbf{B} \in Mod(\Gamma)^+\}.$$

Then we find that an aggregation rule F satisfies the axiom described above if and only if $F \vDash \mathcal{A}$.

Definition 2.8. *The **interpretation** of an axiom instance A is the set of aggregation rules that satisfy A, we denote it as follows:*

$$\mathbb{I}(A) \coloneqq \{F : Mod(\Gamma)^+ \to 2^{\{0,1\}^m} \setminus \{\varnothing\} \mid F \vDash A\}.$$

*The **interpretation** of an axiom $\mathcal{A}$ is the set of aggregation rules that satisfy $\mathcal{A}$, we denote it similarly:*

$$\mathbb{I}(\mathcal{A}) \coloneqq \{F : Mod(\Gamma)^+ \to 2^{\{0,1\}^m} \setminus \{\varnothing\} \mid F \vDash \mathcal{A}\}.$$

*For an integrity constraint Γ, we define the **consistent interpretations** as follows:*

$$\mathbb{I}_\Gamma(A) \coloneqq \mathbb{I}(A) \cap \{F \mid F : Mod(\Gamma)^+ \to 2^{Mod(\Gamma)} \setminus \{\varnothing\}\} \text{ and } \mathbb{I}_\Gamma(\mathcal{A}) \coloneqq \mathbb{I}(\mathcal{A}) \cap \{F \mid F : Mod(\Gamma)^+ \to 2^{Mod(\Gamma)} \setminus \{\varnothing\}\}.$$

These sets denote the subsets of satisfying aggregation rules that are also consistent.

Example 2.5. *If we reconsider Example 2.4, $\mathbb{I}(\mathcal{A})$ is exactly the set of all aggregation rules that satisfy the described axiom. For an integrity constraint Γ, $\mathbb{I}_\Gamma(\mathcal{A})$ is the set of consistent aggregation rules that satisfy the described axiom.*

We note the following about the relation between the interpretation of an axiom and that of the axiom instances.

Remark 2.1. *For an axiom $\mathcal{A}$ consisting of axiom instances A, note that $\mathbb{I}(\mathcal{A}) = \cap_{A \in \mathcal{A}} \mathbb{I}(A)$, as $\mathbb{I}(\mathcal{A})$ is exactly the set of aggregation rules that satisfy every axiom instance $A \in \mathcal{A}$. From this observation it also immediately follows that if $A \in \mathcal{A}$, it holds that $\mathbb{I}(\mathcal{A}) \subseteq \mathbb{I}(A)$, as any aggregation rule that satisfies $\mathcal{A}$ also satisfies A.*

We make another remark concerning the generality of the definition of an axiom.

Remark 2.2. *Every set S of outcome statements is in fact an axiom, as it is a set consisting of axiom instances, namely formulas in $\mathcal{L}(O)$. More specifically, S contains only literals from $\mathcal{L}(O)$. This follows from the fact that every outcome statement is in $L(O)$ (see Lemma 2.4). Because of this, we may also use $\mathbb{I}(S)$ to denote the interpretation of the set S.*

In this thesis, we will often consider different sets of axioms; hence the following definition.

Definition 2.9. *For a set of axioms $\mathbb{A}$, we may generalize Definition 2.8 and define the **interpretation** of this set:*

$$\mathbb{I}(\mathbb{A}) := \bigcap_{\mathcal{A} \in \mathbb{A}} \mathbb{I}(\mathcal{A}).$$

The consistent interpretation is then defined as follows:

$$\mathbb{I}_\Gamma(\mathbb{A}) := \mathbb{I}(\mathbb{A}) \cap \{F \mid F : Mod(\Gamma)^+ \to 2^{Mod(\Gamma)} \setminus \{\varnothing\}\}.$$

When justifying an outcome statement in Chapter 4 of this thesis, we will never use a complete axiom. Instead, we use one or more axiom instances from each axiom in some set $\mathbb{A}$. The following definition comes in handy. Note here that we denote a set of axiom instances by $\mathcal{A}$, which we have earlier used to denote an axiom. Syntactically, every set of axiom instances can be seen as an axiom. However, in the following definition, we do not wish to see $\mathcal{A}$ as an axiom, but rather as a collection of axiom instances.

Definition 2.10. *For a set of axiom instances $\mathcal{A}$ and a set of axioms $\mathbb{A}$, we write $\mathcal{A} \lhd \mathbb{A}$ if every axiom instance in $\mathcal{A}$ is an instance of some axiom in $\mathbb{A}$, i.e. for every axiom instance $A \in \mathcal{A}$ there is an axiom $\mathcal{A}' \in \mathbb{A}$ such that $A \in \mathcal{A}'$.*

Example 2.6. *Suppose we have three axioms $\mathcal{A}^1, \mathcal{A}^2$ and $\mathcal{A}^3$. Then for any three axiom instances $A^1 \in \mathcal{A}^1$, $A^2 \in \mathcal{A}^2$ and $A^3 \in \mathcal{A}^3$, we have that for $\mathcal{A} = \{A^1, A^2, A^3\}$ and $\mathbb{A} = \{\mathcal{A}^1, \mathcal{A}^2, \mathcal{A}^3\}$, it holds that $\mathcal{A} \lhd \mathbb{A}$. It also holds that $\{A^1, A^2\} \lhd \mathbb{A}$.*

We conclude the section by defining several cases of entailment.

Definition 2.11. *Let $\mathcal{A}$ be a set of axiom instances and let $s \in \mathbb{S}$ be an outcome statement. We write $\mathcal{A} \vDash s$ if every aggregation rule that satisfies $\mathcal{A}$ also satisfies s:*

$$\mathcal{A} \vDash s \iff \mathbb{I}(\mathcal{A}) \subseteq \mathbb{I}(s).$$

For an integrity constraint Γ, we define the following:

$$\mathcal{A} \models_\Gamma s \iff \mathbb{I}_\Gamma(\mathcal{A}) \subseteq \mathbb{I}_\Gamma(s).$$

Example 2.7. *Let $m = 3$, $\mathbf{B} \in Mod(\Gamma)^+$, $\Gamma = x_1$, $\mathcal{A} = \{[\![\mathbf{B}, x_2]\!], [\![\mathbf{B}, (x_1 \wedge x_2) \to x_3]\!]\}$ and $s = [\![\mathbf{B}, x_3]\!]$. Then it does not hold that $\mathcal{A} \models s$, as for an aggregation rule F with $F(\mathbf{B}) = \{(0,1,0)\}$, we find that $F \in \mathbb{I}(\mathcal{A})$ while $F \notin \mathbb{I}(s)$, so $\mathbb{I}(\mathcal{A}) \not\subseteq \mathbb{I}(s)$. However, it is the case that $\mathcal{A} \models_\Gamma s$: let $F \in \mathbb{I}_\Gamma(\mathcal{A})$ and let $v \in F(\mathbf{B})$, then $v \models x_1$ as F is consistent and $v \models x_2$ as $F \models [\![\mathbf{B}, x_2]\!]$. Now we find that $v \models x_1 \wedge x_2$ and because $F \models [\![\mathbf{B}, (x_1 \wedge x_2) \to x_3]\!]$, we conclude that $v \models x_3$. As we took an arbitrary outcome $v \in F(\mathbf{B})$, we may now conclude that $F \models [\![\mathbf{B}, x_3]\!]$, so $F \in \mathbb{I}_\Gamma(s)$.*

In Section 3.2, we will consider the situation where an outcome statement follows from the combination of an axiom instance and a set of other outcome statements. We already define it here.

Definition 2.12. *Let S be a set of outcome statements (see Remark 2.2), let $A \in \mathcal{L}(O)$ be an axiom instance and let $s \in \mathbb{S}$ be an outcome statement. Then we write $S, A \models s$ if every aggregation rule that satisfies both S and A also satisfies s, i.e.*

$$S, A \models s \iff \mathbb{I}(S) \cap \mathbb{I}(A) \subseteq \mathbb{I}(s).$$

Example 2.8. *Let $S = \{[\![\mathbf{B}, \varphi]\!]\}$ and let $A = [\![\mathbf{B}, \varphi]\!] \to [\![\mathbf{B}', \psi]\!]$ for $\mathbf{B}, \mathbf{B}' \in Mod(\Gamma)^+$ and $\varphi, \psi \in \mathcal{L}(X)$. Then, for $s = [\![\mathbf{B}', \psi]\!]$, it holds that $S, A \models s$: let F be an aggregation rule such that $F \in \mathbb{I}(S) \cap \mathbb{I}(A)$, then $F \models [\![\mathbf{B}, \varphi]\!]$ and $F \models [\![\mathbf{B}, \varphi]\!] \to [\![\mathbf{B}', \psi]\!]$, so clearly we also have that $F \models [\![\mathbf{B}', \psi]\!]$. We conclude that $F \in \mathbb{I}(\{[\![\mathbf{B}', \psi]\!]\})$.*

2.3 Rewriting the Binary Aggregation Axioms in $\mathcal{L}(O)$

In the following section, we show how to rewrite the axioms from Section 1.4 as axioms in the language of $\mathcal{L}(O)$. Every axiom is a set of axiom instances, which are propositional formulas in $\mathcal{L}(O)$, as discussed in the previous section. First, we provide one more definition which will help us to express more specific statements in $\mathcal{L}(O)$.

Definition 2.13. *For an aggregation rule F, a profile $\mathbf{B} \in Mod(\Gamma)^+$ and an outcome vector $v \in F(\mathbf{B})$, we define the literal set*

$$L_v := \{x \mid x \in X \text{ and } v \models x\} \cup \{\neg x \mid x \in X \text{ and } v \models \neg x\} = \{\ell \mid \ell \in X^* \text{ and } v \models \ell\}.$$

Now let the formula

$$\varphi_v := \bigwedge_{\ell \in L_v} \ell$$

describe the conjunction of every choice made in the outcome v. For every $x \in X$, v satisfies either x or $\neg x$, so φ_v is a formula that precisely describes the outcome vector v. Consequently, let

$$\varphi_{F(\mathbf{B})} := \bigvee_{v \in F(\mathbf{B})} \varphi_v$$

describe the disjunction of the outcome vectors in $F(\mathbf{B})$.

Example 2.9. *Let $m = 3$ and consider some aggregation rule F and profile $\mathbf{B}$ such that $F(\mathbf{B}) =$*

$\{(1,1,0),(1,1,1),(0,0,0)\}$. *Then for* $v = (1,1,0)$*, we would have that* $\varphi_v = x_1 \wedge x_2 \wedge \neg x_3$*. Additionally, we find that* $\varphi_{F(\mathbf{B})} = (x_1 \wedge x_2 \wedge \neg x_3) \vee (x_1 \wedge x_2 \wedge x_3) \vee (\neg x_1 \wedge \neg x_2 \wedge \neg x_3)$.

Note that $F \vDash [\![\mathbf{B}, \varphi_{F(\mathbf{B})}]\!]$ by definition, as any outcome vector in $F(\mathbf{B})$ satisfies one of the formulas in the disjunction $\varphi_{F(\mathbf{B})}$. Also note that for any $v \in F(\mathbf{B})$ we have that $F \vDash \langle\!\langle \mathbf{B}, \varphi_v \rangle\!\rangle$, as there is an outcome vector in $F(\mathbf{B})$ that satisfies φ_v. Finally, note that we may also consider the formulas of the form φ_B for a ballot $B \in \mathrm{Mod}(\Gamma)$, using the set L_B instead of L_v, where we note that B is also a binary vector of size m. We state two lemmas concerning these formulas.

Lemma 2.5. *For two outcome vectors* $v, w \in \{0,1\}^m$*, if* $v \vDash \varphi_w$*, then* $v = w$.

Proof. Suppose $v \vDash \varphi_w$ but $v \neq w$ for contradiction. That means that the vectors disagree on some issue $x \in X$. Suppose without loss of generality that $w \vDash x$ while $v \vDash \neg x$; the other way around is proven analogously. Then we find that $x \in L_w$, which implies that $v \vDash x$, as $v \vDash \varphi_w$. This gives us that $v \vDash x \wedge \neg x$, which contradicts the fact that any vector satisfies either x or $\neg x$, and not both. We conclude that $v = w$. □

Lemma 2.6. *Let* $\mathbf{B}, \mathbf{B}' \in \mathit{Mod}(\Gamma)^+$ *and let* F *be an aggregation rule. Then* $F \vDash [\![\mathbf{B}, \varphi_{F(\mathbf{B}')}]\!]$ *implies that* $F(\mathbf{B}) \subseteq F(\mathbf{B}')$.

Proof. Suppose that $F \vDash [\![\mathbf{B}, \varphi_{F(\mathbf{B}')}]\!]$. This means that any outcome vector in $F(\mathbf{B})$ satisfies $\varphi_{F(\mathbf{B}')}$. Now take any outcome vector $v \in F(\mathbf{B})$. Then it holds that $v \vDash \varphi_{F(\mathbf{B}')}$, which means that there is a formula $\varphi_{v'}$ with $v' \in F(\mathbf{B}')$ such that $v \vDash \varphi_{v'}$. But now we may apply Lemma 2.5 and find that $v = v'$, so $v \in F(\mathbf{B}')$. We conclude that $F(\mathbf{B}) \subseteq F(\mathbf{B}')$. □

Now we are ready to rewrite the axioms from Section 1.4 in the language $\mathcal{L}(O)$. For every axiom we rewrite, we immediately show that the original axiom and the rewritten version are equivalent.

Faithfulness

The faithfulness axiom in $\mathcal{L}(O)$ is denoted as follows:

$$\mathcal{A}^{Fai} := \{[\![\mathbf{B}, \varphi_B]\!] \mid \mathbf{B} \in \mathrm{Mod}(\Gamma)^1 \text{ and } \mathbf{B} = \{B\} \text{ for some } B \in \mathrm{Mod}(\Gamma)\}.$$

Note here that for any profile of size 1, $\mathcal{A}^{Fai}$ contains a formula ensuring that the outcome should be exactly the single ballot submitted in the profile. Now we show that the axiom above is equivalent to the faithfulness axiom introduced in Section 1.4.

Lemma 2.7 (Faithfulness)**.** *For any aggregation rule* F*, it holds that* $F \vDash \mathcal{A}^{Fai}$ *if and only if* F *is faithful.*

Proof. For the left to right implication, suppose that $F \vDash \mathcal{A}^{Fai}$. Now let $\mathbf{B} \in \mathrm{Mod}(\Gamma)^1$ with $\mathbf{B} = \{B\}$. Then it holds that $F \vDash [\![\mathbf{B}, \varphi_B]\!]$, so for any vector $v \in F(\mathbf{B})$ it should hold that $v \vDash \varphi_B$. Now, note that by Lemma 2.5, we get that $v = B$ for any outcome $v \in F(\mathbf{B})$. Since $F(\mathbf{B})$ cannot be empty, we may conclude that $F(\mathbf{B}) = \{B\}$, which shows that F is indeed faithful.

For the other direction, suppose that F is faithful, then it immediately follows that for any $\mathbf{B} \in \mathrm{Mod}(\Gamma)^1$ with $\mathbf{B} = \{B\}$ we have that $F \vDash [\![\mathbf{B}, \varphi_B]\!]$, as $F(\mathbf{B}) = \{B\}$. This concludes our proof. □

Homogeneity

For homogeneity, we find the following axiom:

$$\mathcal{A}^{Hom} := \{[\![\mathbf{B}, \varphi]\!] \leftrightarrow [\![k\mathbf{B}, \varphi]\!] \mid k \in \mathbb{N}_{>0}, \varphi \in \mathcal{L}(X) \text{ and } \mathbf{B}, k\mathbf{B} \in \mathrm{Mod}(\Gamma)^+\}.$$

Lemma 2.8 (Homogeneity)**.** *For any aggregation rule F, it holds that $F \vDash \mathcal{A}^{Hom}$ if and only if F is homogeneous.*

Proof. First, suppose that $F \vDash \mathcal{A}^{Hom}$. Now let $\mathbf{B} \in \mathrm{Mod}(\Gamma)^+$ and consider the profile $k\mathbf{B} \in \mathrm{Mod}(\Gamma)^+$ for an arbitrary $k \in \mathbb{N}_{>0}$ (for which $k\mathbf{B} \in \mathrm{Mod}(\Gamma)^+$). As we trivially have that $F \vDash [\![\mathbf{B}, \varphi_{F(\mathbf{B})}]\!]$ and $F \vDash [\![k\mathbf{B}, \varphi_{F(k\mathbf{B})}]\!]$ and we have that $F \vDash \mathcal{A}^{Hom}$ by assumption, we may conclude that $F \vDash [\![\mathbf{B}, \varphi_{F(k\mathbf{B})}]\!]$ and $F \vDash [\![k\mathbf{B}, \varphi_{F(\mathbf{B})}]\!]$ too. Then Lemma 2.6 gives us both $F(\mathbf{B}) \subseteq F(k\mathbf{B})$ and $F(k\mathbf{B}) \subseteq F(\mathbf{B})$ respectively, which proves that $F(\mathbf{B}) = F(k\mathbf{B})$. As we had taken an arbitrary profile $\mathbf{B}$ and integer k, we have now proven that F is homogeneous.

Now suppose that F is homogeneous and consider $\mathbf{B} \in \mathrm{Mod}(\Gamma)^+$, $k \in \mathbb{N}_{>0}$ and $k\mathbf{B} \in \mathrm{Mod}(\Gamma)^+$. As F is homogeneous, we find that $F(\mathbf{B}) = F(k\mathbf{B})$, so it is immediately clear that for any formula $\varphi \in \mathcal{L}(X)$, we have that $F \vDash [\![\mathbf{B}, \varphi]\!]$ if and only if $F \vDash [\![k\mathbf{B}, \varphi]\!]$. We conclude that $F \vDash \mathcal{A}^{Hom}$. □

Unanimity

Now we define both weak and strong unanimity. Recall that weak unanimity states that whenever every ballot in a profile agrees on an issue, then there should be at least one outcome that also agrees on this issue. This is compactly expressible with the existential outcome statement we defined earlier:

$$\mathcal{A}^{Weak\text{-}Un} := \{\langle\!\langle \mathbf{B}, \ell \rangle\!\rangle \mid \mathbf{B} \in \mathrm{Mod}(\Gamma)^+, \ell \in X^* \text{ and } \forall B \in \mathbf{B} : B \vDash \ell\}.$$

The axiom above contains an existential outcome statement for every profile in which everyone agrees on accepting or rejecting some issue, forcing the fact that there should be at least one outcome that agrees with the ballots in the profile (on that particular issue). We show that both definitions of weak unanimity are equivalent.

Lemma 2.9 (Weak Unanimity)**.** *For any aggregation rule F, it holds that $F \vDash \mathcal{A}^{Weak\text{-}Un}$ if and only if F is weakly unanimous.*

Proof. Suppose that $F \vDash \mathcal{A}^{Weak\text{-}Un}$, now consider any profile $\mathbf{B}$ and literal $\ell \in X^*$ such that for all $B \in \mathbf{B}$ we have that $B \vDash \ell$. Then our assumption gives us that $F \vDash \langle\!\langle \mathbf{B}, \ell \rangle\!\rangle$, so there is at least one outcome vector in $F(\mathbf{B})$ that satisfies ℓ. We conclude that F is weakly unanimous.

For the other direction, if F is weakly unanimous, it is immediately clear that $F \vDash \langle\!\langle \mathbf{B}, \ell \rangle\!\rangle$ for every profile $\mathbf{B} \in \mathrm{Mod}(\Gamma)^+$ and literal $\ell \in X^*$ such that every ballot $B \in \mathbf{B}$ satisfies ℓ. This implies that $F \vDash \mathcal{A}^{Weak\text{-}Un}$, which concludes our proof. □

Strong unanimity can be written as the dual of weak unanimity, where we use the universal outcome statement instead of the existential one:

$$\mathcal{A}^{Strong\text{-}Un} := \{[\![\mathbf{B}, \ell]\!] \mid \mathbf{B} \in \mathrm{Mod}(\Gamma)^+, \ell \in X^* \text{ and } \forall B \in \mathbf{B} : B \vDash \ell\}.$$

Lemma 2.10 (Strong Unanimity)**.** *For any aggregation rule F, it holds that $F \vDash \mathcal{A}^{Strong\text{-}Un}$ if and only if F is strongly unanimous.*

Proof. As for weak unanimity, we find that $\mathcal{A}^{Strong\text{-}Un}$ exactly encapsulates the condition that states that whenever every ballot in a profile agrees on one literal, then all outcomes should also agree (with the ballots) on that literal. We conclude that the proof in both directions is immediate. □

Reinforcement

As reinforcement is an axiom of a more intricate nature, it is not surprising that the equivalent axiom that we will define in this section is a bit more complicated than some of the other axioms. However, there is still a clear correspondence between the axiom as stated in Section 1.4 and the axiom introduced below:

$$\begin{aligned}\mathcal{A}^{Rei} := \{&\big(\langle\!\langle \mathbf{B}', \varphi_v \rangle\!\rangle \wedge \langle\!\langle \mathbf{B}'', \varphi_v \rangle\!\rangle\big) \rightarrow \\ &\big(((\llbracket \mathbf{B}', \varphi \rrbracket \vee \llbracket \mathbf{B}'', \varphi \rrbracket) \rightarrow \llbracket \mathbf{B}, \varphi \rrbracket) \wedge \\ &((\langle\!\langle \mathbf{B}', \varphi_w \rangle\!\rangle \wedge \langle\!\langle \mathbf{B}'', \varphi_w \rangle\!\rangle) \rightarrow \langle\!\langle \mathbf{B}, \varphi_w \rangle\!\rangle)\big) \\ &\mid \mathbf{B}, \mathbf{B}', \mathbf{B}'' \in \mathrm{Mod}(\Gamma)^+, \mathbf{B} = \mathbf{B}' + \mathbf{B}'', \varphi \in \mathcal{L}(X) \text{ and } v, w \in \{0,1\}^m\}.\end{aligned}$$

Before we prove the equivalence, we provide some intuition. The axiom above can be read as follows: whenever the formula on the first line is satisfied, the conjunction of the formulas on the second and third line should also be satisfied. For an aggregation rule F, the formula on the first line is satisfied if there is an outcome $v \in \{0,1\}^m$ that is in both $F(\mathbf{B}')$ and $F(\mathbf{B}'')$, which is equivalent to the reinforcement condition that states that $F(\mathbf{B}') \cap F(\mathbf{B}'') \neq \varnothing$. If this condition is satisfied, reinforcement states that $F(\mathbf{B}) = F(\mathbf{B}') \cap F(\mathbf{B}'')$. In $\mathcal{A}^{Rei}$, we express both inclusions separately. The implication on the second line states that $F(\mathbf{B}) \subseteq F(\mathbf{B}') \cap F(\mathbf{B}'')$; this will follow from Lemma 2.6. The actual equivalence will become clear in the proof of the lemma below. Finally, the formula on the third line describes the fact that $F(\mathbf{B}') \cap F(\mathbf{B}'') \subseteq F(\mathbf{B})$ by stating that whenever some vector is in both $F(\mathbf{B}')$ and $F(\mathbf{B}'')$, it should also be in $F(\mathbf{B})$.

Now, we provide the proof of the equivalence of the two reinforcement definitions.

Lemma 2.11 (Reinforcement)**.** *For any aggregation rule F, it holds that $F \vDash \mathcal{A}^{Rei}$ if and only if F satisfies reinforcement.*

Proof. First, suppose that $F \vDash \mathcal{A}^{Rei}$ and consider the profiles $\mathbf{B}, \mathbf{B}', \mathbf{B}'' \in \mathrm{Mod}(\Gamma)^+$ such that $\mathbf{B} = \mathbf{B}' + \mathbf{B}''$ and $F(\mathbf{B}') \cap F(\mathbf{B}'') \neq \varnothing$. We need to show that $F(\mathbf{B}') \cap F(\mathbf{B}'') = F(\mathbf{B})$ as that would prove that F satisfies reinforcement. Note that we have that $F \vDash \langle\!\langle \mathbf{B}', \varphi_v \rangle\!\rangle \wedge \langle\!\langle \mathbf{B}'', \varphi_v \rangle\!\rangle$ for some $v \in \{0,1\}^m$ as $F(\mathbf{B}') \cap F(\mathbf{B}'') \neq \varnothing$ by assumption, so there is an outcome vector $v \in F(\mathbf{B}') \cap F(\mathbf{B}'')$ and for that v it holds that $F \vDash \langle\!\langle \mathbf{B}', \varphi_v \rangle\!\rangle \wedge \langle\!\langle \mathbf{B}'', \varphi_v \rangle\!\rangle$. Then, as $F \vDash \mathcal{A}^{Rei}$, we first find that $F \vDash (\llbracket \mathbf{B}', \varphi \rrbracket \vee \llbracket \mathbf{B}'', \varphi \rrbracket) \rightarrow \llbracket \mathbf{B}, \varphi \rrbracket$ for any $\varphi \in \mathcal{L}(X)$. If we let $\varphi = \varphi_{F(\mathbf{B}')}$, we find that $F \vDash \llbracket \mathbf{B}', \varphi_{F(\mathbf{B}')} \rrbracket \vee \llbracket \mathbf{B}'', \varphi_{F(\mathbf{B}')} \rrbracket$ (as $F \vDash \llbracket \mathbf{B}', \varphi_{F(\mathbf{B}')} \rrbracket$ by definition) so we then also have that $F \vDash \llbracket \mathbf{B}, \varphi_{F(\mathbf{B}')} \rrbracket$. Analogously, we find that $F \vDash \llbracket \mathbf{B}, \varphi_{F(\mathbf{B}'')} \rrbracket$. Now Lemma 2.6 gives us that both $F(\mathbf{B}) \subseteq F(\mathbf{B}')$ and $F(\mathbf{B}) \subseteq F(\mathbf{B}'')$, so $F(\mathbf{B}) \subseteq F(\mathbf{B}') \cap F(\mathbf{B}'')$.

For the inclusion in the other direction, suppose that $w \in F(\mathbf{B}') \cap F(\mathbf{B}'')$. Then we have that $F \vDash \langle\!\langle \mathbf{B}', \varphi_w \rangle\!\rangle \wedge \langle\!\langle \mathbf{B}'', \varphi_w \rangle\!\rangle$ and we may use the second part of the consequent in $\mathcal{A}^{Rei}$ (the formula on the third line) and find that $F \vDash \langle\!\langle \mathbf{B}, \varphi_w \rangle\!\rangle$. This means that there is an outcome $w' \in F(\mathbf{B})$ such that $w' \vDash w$; by Lemma 2.5 we then get that $w' = w$, so $w \in F(\mathbf{B})$. Now we may conclude that $F(\mathbf{B}') \cap F(\mathbf{B}'') \subseteq F(\mathbf{B})$ so $F(\mathbf{B}') \cap F(\mathbf{B}'') = F(\mathbf{B})$, which proves that F satisfies reinforcement.

For the other direction, suppose that F satisfies reinforcement and consider any profiles $\mathbf{B}, \mathbf{B}', \mathbf{B}'' \in \mathrm{Mod}(\Gamma)^+$ for which $\mathbf{B} = \mathbf{B}' + \mathbf{B}''$, any vectors $v, w \in \{0,1\}^m$ and any formula $\varphi \in \mathcal{L}(X)$. Suppose that $F \vDash \langle\!\langle \mathbf{B}', \varphi_v \rangle\!\rangle \wedge \langle\!\langle \mathbf{B}'', \varphi_v \rangle\!\rangle$, then we need to prove that $F \vDash ((\llbracket \mathbf{B}', \varphi \rrbracket \vee \llbracket \mathbf{B}'', \varphi \rrbracket) \rightarrow \llbracket \mathbf{B}, \varphi \rrbracket) \wedge ((\langle\!\langle \mathbf{B}', \varphi_w \rangle\!\rangle \wedge \langle\!\langle \mathbf{B}'', \varphi_w \rangle\!\rangle) \rightarrow \langle\!\langle \mathbf{B}, \varphi_w \rangle\!\rangle)$. As $F \vDash \langle\!\langle \mathbf{B}', \varphi_v \rangle\!\rangle \wedge \langle\!\langle \mathbf{B}'', \varphi_v \rangle\!\rangle$, we know that $v \in F(\mathbf{B}')$ and $v \in F(\mathbf{B}'')$ (using Lemma 2.5), so $F(\mathbf{B}') \cap F(\mathbf{B}'') \neq \varnothing$. As F satisfies reinforcement, we may conclude that $F(\mathbf{B}') \cap F(\mathbf{B}'') = F(\mathbf{B})$. Now suppose that $F \vDash \llbracket \mathbf{B}', \varphi \rrbracket \vee \llbracket \mathbf{B}'', \varphi \rrbracket$, then either every vector in $F(\mathbf{B}')$ satisfies φ, or every vector in $F(\mathbf{B}'')$ does. As $F(\mathbf{B}) \subseteq F(\mathbf{B}')$ and $F(\mathbf{B}) \subseteq F(\mathbf{B}'')$ (because $F(\mathbf{B}) = F(\mathbf{B}') \cap F(\mathbf{B}'')$), both cases immediately give us that $F \vDash \llbracket \mathbf{B}, \varphi \rrbracket$, so it holds that $F \vDash (\llbracket \mathbf{B}', \varphi \rrbracket \vee \llbracket \mathbf{B}'', \varphi \rrbracket) \rightarrow \llbracket \mathbf{B}, \varphi \rrbracket$. Now suppose that $F \vDash \langle\!\langle \mathbf{B}', \varphi_w \rangle\!\rangle \wedge \langle\!\langle \mathbf{B}'', \varphi_w \rangle\!\rangle$, then we find that $w \in F(\mathbf{B}') \cap F(\mathbf{B}'')$ (again by using Lemma 2.5). Now as $F(\mathbf{B}') \cap F(\mathbf{B}'') = F(\mathbf{B})$, we find that $w \in F(\mathbf{B})$, so $F \vDash \langle\!\langle \mathbf{B}, \varphi_w \rangle\!\rangle$ and we are done. We conclude that $F \vDash \mathcal{A}^{Rei}$. □

Monotonicity

As seen in Section 1.4, the essential part of the monotonicity axiom is the definition of an ℓ-improvement. As we also use that definition in our encoding, the $\mathcal{A}^{Mon}$ axiom is relatively simple.

$$\mathcal{A}^{Mon} := \{\llbracket \mathbf{B}, \ell \rrbracket \rightarrow \llbracket \mathbf{B}', \ell \rrbracket \mid \mathbf{B}, \mathbf{B}' \in \mathrm{Mod}(\Gamma)^+, \ell \in X^* \text{ and } \mathbf{B}' \text{ is an } \ell\text{-improvement of } \mathbf{B}\}.$$

As the axiom above is almost equivalent to the axiom introduced in Section 1.4, the proof is very short.

Lemma 2.12 (Monotonicity)**.** *For any aggregation rule F, it holds that $F \vDash \mathcal{A}^{Mon}$ if and only if F is monotonous.*

Proof. Monotonicity states that, for any $\ell \in X^*$ and any ℓ-improvement $\mathbf{B}'$ of $\mathbf{B}$, if every outcome in $F(\mathbf{B})$ satisfies ℓ, then every outcome in $F(\mathbf{B}')$ satisfies ℓ. Note that this is exactly what is written in $\mathcal{A}^{Mon}$, so the proof is immediate. □

Majority-preservation

For majority-preservation, we want to model the fact that for any majority-consistent profile $\mathbf{B}$, we should have that $F(\mathbf{B}) = \mathrm{ext}(m(\mathbf{B}))$. Before we state the axiom, we define the following formula, analogous to the definition of $\varphi_{F(\mathbf{B})}$:

$$\varphi_{\mathrm{ext}(m(\mathbf{B}))} := \bigvee_{v \in \mathrm{ext}(m(\mathbf{B}))} \varphi_v,$$

where we recall that φ_v is the conjunction of all literals that v satisfies (see Definition 2.13). Now we are ready to rewrite the axiom:

$$\begin{aligned}\mathcal{A}^{Maj} := &\{\langle\!\langle \mathbf{B}, \varphi_v \rangle\!\rangle \mid \mathbf{B} \in \mathrm{Mod}(\Gamma)^+, v \in \mathrm{ext}(m(\mathbf{B}))\} \\ &\cup \{\llbracket \mathbf{B}, \varphi_{\mathrm{ext}(m(\mathbf{B}))} \rrbracket \mid \mathbf{B} \in \mathrm{Mod}(\Gamma)^+ \text{ and } \mathrm{ext}(m(\mathbf{B})) \neq \varnothing\}.\end{aligned}$$

Note that the first part ensures that any majority extension will be in the outcome set (if there is any), while the second part ensures that the outcome set is a subset of the set of majority extensions. Also note that if a profile $\mathbf{B}$ is not majority-consistent, there are no formulas of the form $\langle\!\langle \mathbf{B}, \varphi_v \rangle\!\rangle$ or $\llbracket \mathbf{B}, \varphi_{\mathrm{ext}(m(\mathbf{B}))} \rrbracket$ in $\mathcal{A}^{Maj}$. Now we prove that the axiom is correctly rewritten.

Lemma 2.13 (Majority-preservation)**.** *For any aggregation rule F, it holds that $F \vDash \mathcal{A}^{Maj}$ if and only if F is majority-preserving.*

Proof. First suppose that $F \vDash \mathcal{A}^{Maj}$ and let $\mathbf{B} \in \mathrm{Mod}(\Gamma)^+$ be majority-consistent. We want to prove that $F(\mathbf{B}) = \mathrm{ext}(m(\mathbf{B}))$, because then we have shown that F is majority-preserving. First note that for every $v \in \mathrm{ext}(m(\mathbf{B}))$, we have that $F \vDash \langle\!\langle \mathbf{B}, \varphi_v \rangle\!\rangle$, and for the outcome vector $w \in F(\mathbf{B})$ for which $w \vDash \varphi_v$, Lemma 2.5 gives us that $w = v$, so $v \in F(\mathbf{B})$. This gives us that $\mathrm{ext}(m(\mathbf{B})) \subseteq F(\mathbf{B})$. Now note that $F \vDash [\![\mathbf{B}, \varphi_{\mathrm{ext}(m(\mathbf{B}))}]\!]$ by the second part of $\mathcal{A}^{Maj}$. Now if we follow the proof of Lemma 2.6 but substitute $\varphi_{\mathrm{ext}(m(\mathbf{B}))}$ for $\varphi_{F(\mathbf{B}')}$, we find that $F(\mathbf{B}) \subseteq \mathrm{ext}(m(\mathbf{B}))$. We conclude that $F(\mathbf{B}) = \mathrm{ext}(m(\mathbf{B}))$, so we have proven that F is majority-preserving.

For the implication from right to left, suppose that F is majority-preserving. Recall that for any profile $\mathbf{B}$ that is not majority-consistent, there will not be any formulas of the form $\langle\!\langle \mathbf{B}, \varphi_v \rangle\!\rangle$ or $[\![\mathbf{B}, \varphi_{\mathrm{ext}(m(\mathbf{B}))}]\!]$ in $\mathcal{A}^{Maj}$. Now suppose $\mathbf{B}$ is majority-consistent, then as F is majority-preserving, we know that $F(\mathbf{B}) = \mathrm{ext}(m(\mathbf{B}))$. Then it is immediately clear that for any $v \in \mathrm{ext}(m(\mathbf{B}))$ we have that $F \vDash \langle\!\langle \mathbf{B}, \varphi_v \rangle\!\rangle$, as $v \in \mathrm{ext}(m(\mathbf{B}))$ implies that $v \in F(\mathbf{B})$. It also evidently holds that $F \vDash [\![\mathbf{B}, \varphi_{\mathrm{ext}(m(\mathbf{B}))}]\!]$, as $\varphi_{\mathrm{ext}(m(\mathbf{B}))} = \varphi_{F(\mathbf{B})}$. We conclude that $F \vDash \mathcal{A}^{Maj}$ and we are done. □

Cancellation

Finally, we rewrite the cancellation axiom and show that it is equivalent to the original definition. We define it as follows:

$$\mathcal{A}^{Can} := \{\langle\!\langle \mathbf{B}, \varphi_v \rangle\!\rangle \mid \mathbf{B} \in \mathrm{Mod}(\Gamma)^+, v \in \mathrm{Mod}(\Gamma) \text{ and } \forall x \in X : N(\mathbf{B}, x) = N(\mathbf{B}, \neg x)\}.$$

Now we provide the final equivalence.

Lemma 2.14 (Cancellation)**.** *For any aggregation rule F, it holds that $F \vDash \mathcal{A}^{Can}$ if and only if F satisfies cancellation.*

Proof. First suppose that $F \vDash \mathcal{A}^{Can}$ and let $\mathbf{B} \in \mathrm{Mod}(\Gamma)^+$ such that $N(\mathbf{B}, x) = N(\mathbf{B}, \neg x)$ for every $x \in X$. Then as $F \vDash \mathcal{A}^{Can}$, we find that $F \vDash \langle\!\langle \mathbf{B}, \varphi_v \rangle\!\rangle$ for every $v \in \mathrm{Mod}(\Gamma)$. From Lemma 2.5 we may then conclude that $v \in F(\mathbf{B})$ for every $v \in \mathrm{Mod}(\Gamma)$, so $\mathrm{Mod}(\Gamma) \subseteq F(\mathbf{B})$ which proves that F satisfies cancellation.

Now suppose that F satisfies cancellation and consider a profile $\mathbf{B} \in \mathrm{Mod}(\Gamma)^+$ such that $N(\mathbf{B}, x) = N(\mathbf{B}, \neg x)$ for every $x \in X$. Then we find that $\mathrm{Mod}(\Gamma) \subseteq F(\mathbf{B})$, so for every $v \in \mathrm{Mod}(\Gamma)$ it holds that $F \vDash \langle\!\langle \mathbf{B}, \varphi_v \rangle\!\rangle$. We conclude that $F \vDash \mathcal{A}^{Can}$. □

2.4 Discussion

In this chapter, we have defined outcome statements for binary aggregation. They offer an alternative for the outcome statements defined by Boixel et al. [BEH22], who define an outcome statement (for voting theory) as a tuple $\langle \succ_N, \mathcal{O} \rangle$, where $\succ_N$ is a profile (of preferences) and $\mathcal{O}$ is a set of possible outcome sets; a possible outcome set here is a subset of the candidates. The idea is then that whenever a voting rule satisfies such a statement, the outcome set of that rule under input profile $\succ_N$ should be one of the possible outcome sets in $\mathcal{O}$.

As already noted in the introduction of this chapter, there are $2^{2^m} - 1$ possible outcomes (without taking the integrity constraint into account). Because of this, the direct translation of

their definition of outcome statements to binary aggregation is an inconvenient way of describing aggregation rules. For example, if we want to describe the set of outcome sets where every outcome vector satisfies x_1, we should consider all possible subsets of binary vectors that start with a 1; we would need to state the $2^{2^{m-1}} - 1$ possible outcome sets. We conclude that the direct translation of the definition of outcome statements for voting theory by Boixel et al. [BEH22] would not suffice.

In Definition 2.1, we offered an alternative definition. Instead of stating what the outcome set should be, our definition describes what the outcome set should look like, i.e. what formulas the outcome vectors should satisfy. This allows for a more readable and intuitive way of describing the behavior of an aggregation rule. As is seen in Section 2.3, these outcome statements contain all the expressive power that we need, and the duality of the existential and universal outcome statements allows us to compactly denote the binary aggregation axioms in an intuitive manner.

It is important to note here that we could still express the direct translation of the voting theory outcome statements within our own adapted definition. For an aggregation rule F, a profile $\mathbf{B} \in \mathrm{Mod}(\Gamma)^+$ and a set of possible outcome sets $\mathcal{O} \subseteq 2^{\{0,1\}^m} \setminus \{\varnothing\}$, we find the following:

$$F \text{ satisfies } \langle \mathbf{B}, \mathcal{O} \rangle \iff F \vDash \bigvee_{O \in \mathcal{O}} \big((\bigwedge_{v \in O} \langle\!\langle \mathbf{B}, \varphi_v \rangle\!\rangle) \wedge [\![\mathbf{B}, \varphi_O]\!] \big),$$

where we make use of Definition 2.13. Here we note that $F \vDash (\bigwedge_{v \in O} \langle\!\langle \mathbf{B}, \varphi_v \rangle\!\rangle) \wedge [\![\mathbf{B}, \varphi_O]\!]$ if and only if $F(\mathbf{B}) = O$. We conclude that our definition is a generalization of the definition proposed by Boixel et al. [BEH22].

Chapter 3

The Tableau Method

In Chapter 4, we will define both unstructured and structured justifications. In this chapter, we introduce a tableau-based calculus, which will serve as a means of structuring an unstructured justification through the use of a graph, or more specifically: a tree. The specific tree that we will define in this chapter, we call a *tableau*. Every node in the tableau represents a set of outcome statements and each edge represents some sort of inference in which we add or alter some outcome statement. In the construction of such a tableau, we start off with a single root node corresponding to a set of outcome statements and then gradually expand the tableau by adding new nodes. This chapter is based on the work of Boixel, Endriss and De Haan [BEH22], who have defined such a calculus to obtain structured justifications for voting theory. However, we introduce a different set of expansion rules and our notion of an outcome statement is different from theirs. Other than that, the general structure of the method remains the same.

3.1 Introduction to Tableaux

The tableau method is a proof procedure that can be used to show the unsatisfiability of one or more logical formulas through the use of a set of inference rules; see the work of D'Agostino, Gabbay, Hähnle and Posegga [DAg+13] for an extensive description of this method. When using the tableau method, one is concerned with the construction of a tableau, which is a dynamical mathematical object. In principle, a tableau is a graph with some extra properties, which will be set out later on in the formal definition. Every node in the graph corresponds to a set of formulas, and every edge in the graph corresponds to an *expansion rule*. A tableau is constructed gradually by applying expansion rules and thereby *expanding* the tableau. In a tableau for classical propositional logic, as defined by D'Agostino [DAg13], an example of such an expansion rule could be: if we are in some node which is labeled with a set of formulas that contains the formula $p \wedge q$, we may expand the tableau by drawing an edge from this node to a new node containing the separate formulas p and q instead of $p \wedge q$, see Figure 3.1. Another example is the expansion rule for a disjunction of formulas, see Example 3.2. This rule illustrates a case distinction: if $p \vee q$ holds, then either p holds or q holds.

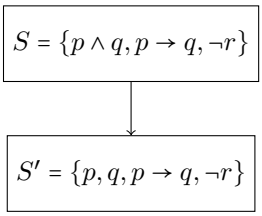

Figure 3.1: Expansion rule for conjunction

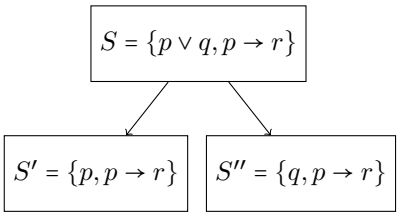

Figure 3.2: Expansion rule for disjunction

In general, we instantiate such a tableau with a single node that corresponds to an initial set of formulas. Consequently, we start expanding the tableau by gradually applying expansion rules.

In this thesis, the tableau method will be used to prove that there is no consistent aggregation rule that satisfies a certain set of outcome statements in combination with a set of axiom instances. We initialize a tableau by defining a single node that is labeled with this set of outcome statements. This node we call the *root* node. Then, we apply expansion rules in order to add new nodes to the tableau. Every node in the tableau is labeled with a set of outcome statements. In any node S, we consider the set $\mathbb{I}(S)$ of aggregation rules that satisfy the set of outcome statements S. The goal is to end up with a tree such that all the leaf nodes correspond to sets of outcome statements that are clearly inconsistent, which means that all of these sets should contain a statement of the form $[\![\mathbf{B}, \bot]\!]$ or $\langle\!\langle \mathbf{B}, \bot \rangle\!\rangle$. This we call a *closed* tableau. If we can construct such a tableau, we may then conclude that there is no consistent aggregation rule that satisfies the set that we used to instantiate the tableau (in combination with the axiom instances that appear in the tableau).

Formally, a tableau is defined as follows.

Definition 3.1. *A **tableau** $\mathcal{T}$ is a tree with exactly one root node, while all other nodes have an in-degree of 1. Every node is labeled with a set S of outcome statements, and every edge is labeled with an **expansion rule**. We label the root node of the tableau with the set $\mathcal{S}$. The root node is always labeled with a finite set.*

3.2 The Expansion Rules

In this section, we define the set of expansion rules which we will be using in our tableaux. We define our expansion rules given a set of axiom instances. We distinguish seven different rules. Some of them will simply expand the tableau by adding an extra outcome statement to the set corresponding to some leaf node, while others will actually reorganize the set of outcome statements, e.g. by rewriting some formula in some outcome statement. The expansion rules are based the work of Boixel et al. [BEH22].

Definition 3.2. *Let $\mathcal{A}$ be a set of axiom instances. We define the ways to expand a tableau with new nodes and edges using **expansion rules**. Every rule will **expand** a branch in the tableau that ends in a leaf node S by introducing one or more child nodes (with corresponding directed edges from S to the newly introduced nodes). If it is clear or irrelevant which branch is expanded, we might also simply speak of expanding the tableau (instead of a specific branch).*

Let S be a leaf node of some tableau $\mathcal{T}$. We define the following expansion rules:

- ***Axiom-driven expansion rule:** For any instance $A \in \mathcal{A}$, profile $\mathbf{B} \in Mod(\Gamma)^+$ and formula*

$\varphi \in \mathcal{L}(X)$ *such that* $S, A \vDash [\![\mathbf{B}, \varphi]\!]$ *(see Definition 2.12), we may expand the branch ending in* S *with a node* $S' = S \cup \{[\![\mathbf{B}, \varphi]\!]\}$*, given that* $\mathbb{I}(S') \subsetneq \mathbb{I}(S)$*.*

Analogously, if $S, A \vDash \langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle$*, we may expand the branch ending in* S *with a node* $S' = S \cup \{\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle\}$*, given that* $\mathbb{I}(S') \subsetneq \mathbb{I}(S)$*.*

- ***Constraint-driven expansion rule:*** *For any propositional formula* $\varphi \in \mathcal{L}(X)$ *such that* $\Gamma \vDash \varphi$ *and for any profile* $\mathbf{B} \in Mod(\Gamma)^+$*, we may expand the branch ending in* S *with a node* $S' = S \cup \{[\![\mathbf{B}, \varphi]\!]\}$*, given that* $\mathbb{I}(S') \subsetneq \mathbb{I}(S)$*.*

- ***Branching rule:*** *For any profile* $\mathbf{B} \in Mod(\Gamma)^+$ *and formula* $\varphi \in \mathcal{L}(X)$*, we may expand the branch ending in* S *with two new nodes* $S' = S \cup \{[\![\mathbf{B}, \varphi]\!]\}$ *and* $S'' = S \cup \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}$ ***or*** $S' = S \cup \{[\![\mathbf{B}, \neg\varphi]\!]\}$ *and* $S'' = S \cup \{\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle\}$*, given that in either case, both* $\mathbb{I}(S') \subsetneq \mathbb{I}(S)$ *and* $\mathbb{I}(S'') \subsetneq \mathbb{I}(S)$*.*

- ***Simplification rule:*** *For any* $[\![\mathbf{B}, \varphi]\!], [\![\mathbf{B}, \psi]\!] \in S$*, we may expand the branch ending in* S *by adding a node* $S' = \big(S \setminus \{[\![\mathbf{B}, \varphi]\!], [\![\mathbf{B}, \psi]\!]\}\big) \cup \{[\![\mathbf{B}, \varphi \wedge \psi]\!]\}$*.*

- ***Rewrite rule:*** *For any* $[\![\mathbf{B}, \varphi]\!] \in S$ *and formula* $\psi \in \mathcal{L}(X)$ *such that* $\psi \equiv \varphi$ *and* $L(\psi) < L(\varphi)$*, we may expand the branch ending in* S *by adding a node* $S' = \big(S \setminus \{[\![\mathbf{B}, \varphi]\!]\}\big) \cup \{[\![\mathbf{B}, \psi]\!]\}$*.*

  We may do the same for outcome statements of the form $\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle$*.*

- ***Witness rule:*** *For any outcome statement* $[\![\mathbf{B}, \varphi]\!] \in S$*, we may expand the branch ending in* S *by adding a node* $S' = S \cup \{\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle\}$*, given that there is no statement in* S *of the form* $\langle\!\langle \mathbf{B}, \psi \rangle\!\rangle$ *with* $\psi \equiv \varphi$*.*

- ***Refinement rule:*** *For any* $\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle, [\![\mathbf{B}, \psi]\!] \in S$*, we may expand the branch ending in* S *by adding a node* $S' = S \cup \{\langle\!\langle \mathbf{B}, \varphi \wedge \psi \rangle\!\rangle\}$*, given that there is no statement in* S *of the form* $\langle\!\langle \mathbf{B}, \rho \rangle\!\rangle$ *with* $\rho \equiv \varphi \wedge \psi$*.*

The definition above is rather elaborate, so we will extensively illustrate its workings with examples.

Axiom-driven Expansion Rule

The first expansion rule is the axiom-driven expansion rule. The idea of this rule is that we can add outcome statements that are a consequence of the axiom instances in $\mathcal{A}$. The condition at the end ensures that we will only add outcome statements that actually narrow down the number of aggregation rules that satisfy the set of outcome statements; otherwise we could infinitely add trivial outcome statements. We provide two examples of this rule. In the first example, we can simply add the axiom instance as it is also an outcome statement, i.e. it is an atomic formula in $\mathcal{L}(O)$. It is immediately clear that $S, A \vDash A$

Example 3.1. *Suppose there is some axiom instance* $[\![\mathbf{B}, \varphi]\!] \in \mathcal{A}$ *and let* $\mathcal{T}$ *be some tableau containing a branch ending in the leaf node* S*. Then we may expand this branch through the use of the axiom-driven expansion rule by simply adding a new leaf node* $S' = S \cup \{[\![\mathbf{B}, \varphi]\!]\}$ *and a corresponding directed edge, given that* $\mathbb{I}(S') \subsetneq \mathbb{I}(S)$*. This is illustrated in Figure 3.3.*

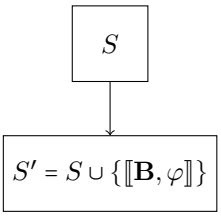

Figure 3.3: The axiom-driven expansion rule described in Example 3.1

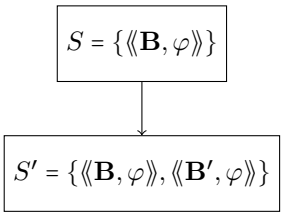

Figure 3.4: The axiom-driven expansion rule described in Example 3.2

In the second example, our axiom instance is an implication, so we cannot simply add it, as it is not an outcome statement. Rather, we add an outcome statement s for which it holds that $S, A \vDash s$.

Example 3.2. *Suppose we have some axiom instance $A = \langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle \to \langle\!\langle \mathbf{B}', \varphi \rangle\!\rangle$ with $A \in \mathcal{A}$ and suppose we are in some tableau $\mathcal{T}$ that contains a branch ending in the leaf node $S = \{\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle\}$. Then it holds that $S, A \vDash \langle\!\langle \mathbf{B}', \varphi \rangle\!\rangle$, so we may apply the axiom-driven expansion rule to the branch ending in S by adding the node $S' = S \cup \{\langle\!\langle \mathbf{B}', \varphi \rangle\!\rangle\}$, again with a corresponding edge from S to S', see Figure 3.4. Note here that it evidently holds that $\mathbb{I}(S') \subsetneq \mathbb{I}(S)$.*

Constraint-driven Expansion Rule

Consequently, we consider the constraint-driven expansion rule. This expansion rule allows us to add a universal outcome statement for any profile and any logical consequence of the integrity constraint. This rule is motivated by the fact that we only want to consider outcomes that actually satisfy the constraint. Sometimes, if the constraint is large, it might be better for the readability of the inference step to only introduce a consequence of the constraint rather than the formula itself. Again, we require the newly introduced outcome statement to actually narrow down the set of satisfying aggregation rules (to prevent infinite application of the rule). We illustrate the rule with two examples.

Example 3.3. *Let Γ be an integrity constraint, let $\mathbf{B} \in Mod(\Gamma)^+$ and let $\mathcal{T}$ be some tableau containing a branch ending in the leaf node S. Then we may expand this branch through the use of the constraint-driven expansion rule by simply adding a new leaf node $S' = S \cup \{[\![\mathbf{B}, \Gamma]\!]\}$ with a corresponding edge, given that $\mathbb{I}(S') \subsetneq \mathbb{I}(S)$. See Figure 3.5.*

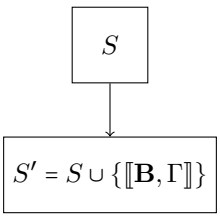

Figure 3.5: The constraint-driven expansion rule described in Example 3.3

As described in the definition, we can also add any logical consequence of the constraint.

Example 3.4. *Let $\Gamma = (x_1 \to x_2) \wedge (x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \to x_3)$, let $\mathbf{B} \in Mod(\Gamma)^+$ and let $\mathcal{T}$ be some tableau containing a branch ending in the leaf node $S = \{[\![\mathbf{B}, \neg x_1 \wedge \neg x_3 \wedge \neg x_4]\!]\}$. Then we may*

expand this branch through the use of the constraint-driven expansion rule by adding a new leaf node $S' = S \cup \{[\![\mathbf{B}, x_1 \vee x_3 \vee x_4]\!]\}$ *and an edge* (S, S')*. Here it is clear that* $\mathbb{I}(S') \subsetneq \mathbb{I}(S)$*, in fact, we even have that* $\mathbb{I}(S') = \varnothing$*. See Figure 3.6.*

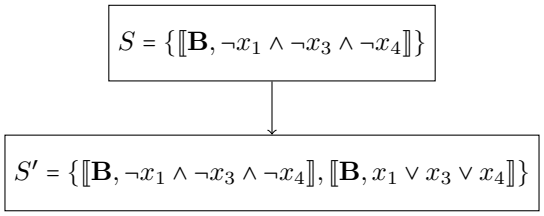

Figure 3.6: The constraint-driven expansion rule described in Example 3.4

Branching Rule

The branching rule is the only rule that introduces more than one new leaf node. It parallels the idea of the following case distinction: for every profile and every formula, it should either hold that all outcomes satisfy the formula, or there is an outcome that does not satisfy it. The same holds for the negation of this formula, which is why we allow the second option in the definition. Again, we require the set of aggregation rules to actually get smaller in both newly introduced leaf nodes.

Example 3.5. *Suppose we are in some tableau* $\mathcal{T}$ *that contains a branch ending in the leaf node* S *and let* $\mathbf{B} \in \mathit{Mod}(\Gamma)^+$ *and* $\varphi \in \mathcal{L}(X)$*. Then we may apply the branching rule to the branch ending in* S *by adding the nodes* $S' = S \cup \{[\![\mathbf{B}, \varphi]\!]\}$ *and* $S'' = S \cup \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}$*, together with the edges* (S, S') *and* (S, S'')*, given that both* $\mathbb{I}(S') \subsetneq \mathbb{I}(S)$ *and* $\mathbb{I}(S'') \subsetneq \mathbb{I}(S)$*. This is illustrated in Figure 3.7.*

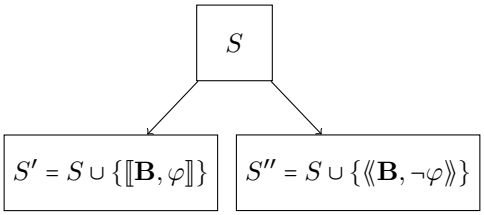

Figure 3.7: The branching rule described in Example 3.5

Example 3.6. *If we reconsider the scenario in Example 3.5, note that we can also apply the branching rule to the branch ending in* S *by adding the nodes* $S' = S \cup \{[\![\mathbf{B}, \neg\varphi]\!]\}$ *and* $S'' = S \cup \{\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle\}$*, together with the edges* (S, S') *and* (S, S'')*, given that both* $\mathbb{I}(S') \subsetneq \mathbb{I}(S)$ *and* $\mathbb{I}(S'') \subsetneq \mathbb{I}(S)$*.*

Simplification Rule

The simplification rule allows us to make the following inference: if every outcome under input $\mathbf{B}$ should satisfy φ and every outcome under input $\mathbf{B}$ should satisfy ψ, then every outcome under input $\mathbf{B}$ should satisfy $\varphi \wedge \psi$. This is the first rule that actually deletes an outcome statement when applied. We illustrate it with an example.

Example 3.7. *For a tableau* $\mathcal{T}$ *that contains a branch ending in a leaf node* $S = \{[\![\mathbf{B}, \varphi]\!], [\![\mathbf{B}, \psi]\!]\}$*, we may expand this branch through the application of the simplification rule by adding a node* $S' = \{[\![\mathbf{B}, \varphi \wedge \psi]\!]\}$ *and an edge* (S, S')*, see Figure 3.8.*

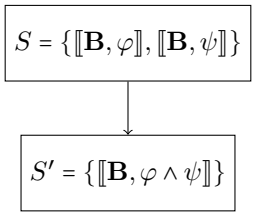

Figure 3.8: The simplification rule described in Example 3.7

Rewrite Rule

The rewrite rule allows us to rewrite the formulas in the outcome statements into shorter (equivalent) formulas. An important example is the deletion of double negations, see Example 3.8.

Example 3.8. *For a tableau $\mathcal{T}$ that contains a branch ending in a leaf node $S = \{[\![\mathbf{B}, \neg\neg\varphi]\!]\}$, we may expand this branch through the application of the rewrite rule by adding a node $S' = \{[\![\mathbf{B}, \varphi]\!]\}$ and an edge (S, S'), see Figure 3.9.*

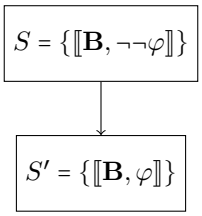

Figure 3.9: The rewrite rule described in Example 3.8

Witness Rule

The witness rule mirrors the following inference: if every outcome in some set $F(\mathbf{B})$ satisfies φ, then there is an outcome in $F(\mathbf{B})$ that satisfies φ. This follows from the fact that $F(\mathbf{B}) \neq \varnothing$ by definition. In the application of the witness rule, we simply add the existential outcome statement, without deleting the corresponding universal statement. We do require that the leaf node of the branch that we expand does not contain an equivalent statement already.

Example 3.9. *For a tableau $\mathcal{T}$ that contains a branch ending in a leaf node $S = \{[\![\mathbf{B}, \varphi]\!]\}$, we may expand this branch through the application of the witness rule by adding a node $S' = \{[\![\mathbf{B}, \varphi]\!], \langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle\}$ and an edge (S, S'), see Figure 3.10.*

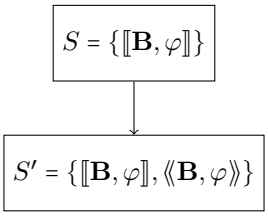

Figure 3.10: The witness rule described in Example 3.9

Refinement Rule

Our final rule is the refinement rule. It concerns the following situation: if there is an outcome in some set $F(\mathbf{B})$ that satisfies φ and every outcome in $F(\mathbf{B})$ satisfies ψ, then there is an outcome that satisfies $\varphi \wedge \psi$ (namely the outcome that already satisfies φ). As with the witness rule, we require that the leaf node of the branch that we expand does not already contain an equivalent statement. We provide one more example.

Example 3.10. *For a tableau $\mathcal{T}$ that contains a branch ending in a leaf node $S = \{\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle, [\![\mathbf{B}, \psi]\!]\}$, we may expand this branch through the application of the refinement rule by adding a node $S' = \{\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle, [\![\mathbf{B}, \psi]\!], \langle\!\langle \mathbf{B}, \varphi \wedge \psi \rangle\!\rangle\}$ and an edge (S, S'), see Figure 3.13.*

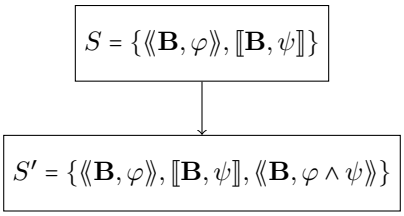

Figure 3.11: The refinement rule described in Example 3.10

3.3 Correctness of the Calculus

Now that we have illustrated the inner workings of our tableau-based model, we want to prove that it actually behaves desirably. This comes down to first showing that the process of applying expansion rules will eventually terminate and then proving soundness and completeness of our tableau-based method. Before we are able to formally state these three properties, some further definitions are required. The following definitions are also based on the work of Boixel et al. [BEH22]. In this section, we generally follow the structure of Section 3.3 from their paper.

Earlier, we discussed the fact that we use the tableau method to prove that there is no consistent aggregation rule that satisfies a set of outcome statements $\mathcal{S}$ in combination with a set of axiom instances $\mathcal{A}$. We do so by constructing a tableau, rooted in some node labeled with the set $\mathcal{S}$, such that all the leaf nodes in the constructed tableau contain inconsistencies. Then we conclude that there is no consistent aggregation rule F that satisfies both the statements in the root node $\mathcal{S}$ and the axiom instances in the set $\mathcal{A}$. We define the following concepts.

Definition 3.3. *In a tableau $\mathcal{T}$, we call a node S **inconsistent** if $\mathbb{I}(S) = \varnothing$, and **consistent** otherwise. An outcome statement is called **inconsistent** if it is of the form $[\![\mathbf{B}, \bot]\!]$ or $\langle\!\langle \mathbf{B}, \bot \rangle\!\rangle$, for some profile $\mathbf{B}$. We say that $\mathcal{T}$ is **closed** if every leaf node contains an inconsistent outcome statements; otherwise it is **open**.*

Note here that any node containing an inconsistent outcome statement is itself inconsistent, but not any inconsistent node contains an inconsistent outcome statement, as it could for example contain two statements of the form $[\![\mathbf{B}, \varphi]\!]$ and $[\![\mathbf{B}, \neg\varphi]\!]$. We illustrate these definitions with an example.

Example 3.11. *Consider the tableau $\mathcal{T}$ in Figure 3.12. We do not consider the expansion rules as the example serves the purpose of dissecting the definitions in Definition 3.3. Note here that node*

S_2 contains an inconsistent outcome statement. Also note that the node S_4 is inconsistent, as we can never satisfy the outcome statements in S_4. However, S_4 does not contain any inconsistent outcome statement. We conclude that $\mathcal{T}$ is not closed. We could however easily close it by applying the simplification rule to the branch ending in S_4 and then applying the rewrite rule, where we note that $(x_1 \wedge x_2) \wedge (x_1 \to \neg x_2) \equiv \bot$.

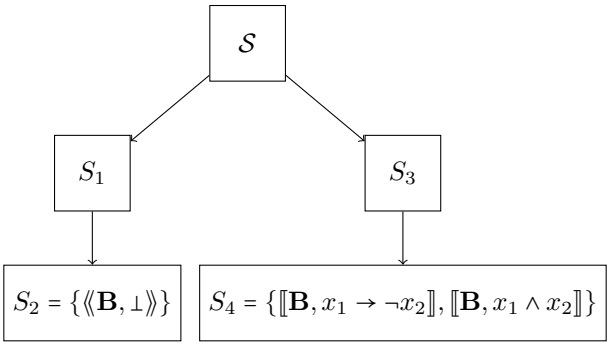

Figure 3.12: An example illustrating Definition 3.3

For Definition 3.2 we require a set of axiom instances $\mathcal{A}$. This is the set of axiom instances that can be applied in the axiom-driven expansion rules of the tableau. We define the following.

Definition 3.4. *We say that a tableau is **licensed by** a set of axiom instances $\mathcal{A}$ if every axiom instance A that is used in an application of the axiom-driven expansion rule in the tableau is in the set $\mathcal{A}$.*

If there actually is a consistent aggregation rule that satisfies the outcome statements in the root node $\mathcal{S}$ and the axiom instances in some set $\mathcal{A}$, we will never be able to construct a closed tableau that is rooted in $\mathcal{S}$ and licensed by $\mathcal{A}$. We still want the process of applying expansion rules to eventually stop. The following definition captures this situation.

Definition 3.5. *A tableau $\mathcal{T}$ is **saturated** if no branch in $\mathcal{T}$ can be expanded further by the application of one of the expansion rules.*

A saturated tableau $\mathcal{T}$ cannot be expanded further, so there is no branch in $\mathcal{T}$ ending in a leaf node S that can be expanded. For this to be the case, the outcome set S should satisfy certain properties. In the next example, we illustrate the situation. First, we establish some notation concerning the depiction of tableaux in the rest of this thesis.

Notation 3.1. *From now on, when depicting a tableau, we do not denote the outcome sets in the nodes. Instead, we only explicitly write down the outcome statements when introducing them. The edges in the tableau are labeled with the expansion rule that is applied. If the axiom-driven expansion rule is applied, we instead label the edge with the relevant axiom. This method of depicting tableaux is taken from the paper by Boixel et al. [BEH22].*

Example 3.12. *For simplicity, let $n = 1$, $m = 1$ and $\Gamma = \top$. Then we find that $Mod(\Gamma)^+ = \{\{(0)\}, \{(1)\}\}$. Now consider the trivial set of outcome statements $\mathcal{S} = \{[\![\{(0)\}, x_1 \vee \neg x_1]\!]\}$ and let $\mathcal{A} = \mathcal{A}^{Fai}$; note here that for $m = 1$, we find that $\mathcal{A}^{Fai} = \{[\![\{(0)\}, \neg x_1]\!], [\![\{(1)\}, x_1]\!]\}$.*

The tableau in Figure 3.13 is a saturated tableau that is licensed by $\mathcal{A} = \mathcal{A}^{Fai}$ and rooted in $\mathcal{S}$. To conclude that the tableau is saturated, we observe the following. Any axiom instance in $\mathcal{A}$ is

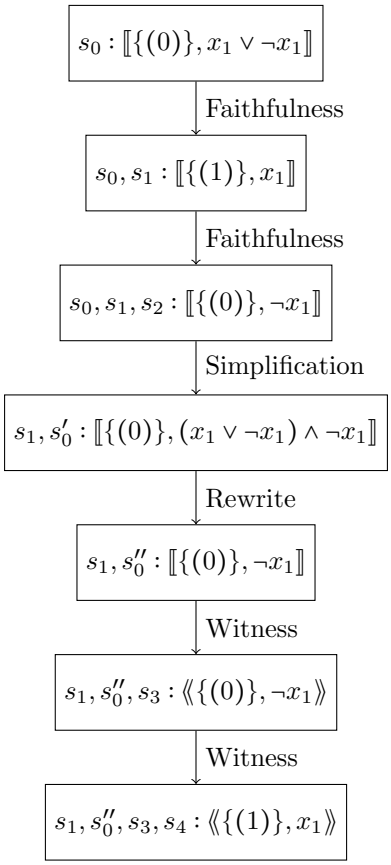

Figure 3.13: The saturated tableau described in Example 3.12

already introduced through the application of the axiom-driven expansion rule, so no application of the axiom-driven expansion rule will further narrow down the set of satisfying aggregation rules. As $\Gamma = \top$*, no application of the constraint-driven expansion rule will narrow down the set of satisfying aggregation rules. As the leaf node in the tableau already completely describes a specific aggregation rule, we will never be able to satisfy the precondition of the branching rule, as we cannot make any case distinction about the properties of the aggregation rules in* $\mathbb{I}(S)$ *(where* S *denotes the leaf node). It is immediately evident that we can no longer apply the simplification rule, the rewrite rule, the witness rule or the refinement rule.*

3.3.1 Termination

Now we are ready to prove that the construction of any tableau eventually terminates. It is formalized as follows. Note that because every node has at most two children, a tableau is infinite (contains infinitely many nodes) if and only if it has an infinite branch; see Remark 1.1.

Boixel et al. [BEH22] also show the termination of their calculus for the framework of voting theory. Our proof is a bit more extensive, as our set of expansion rules is bigger than theirs.

Theorem 3.2 (Termination)**.** *When using the tableau expansion rules from Definition 3.2, we can never construct a tableau that contains an infinite branch. As a consequence, any gradual expansion of a tableau (through the application of the expansion rules) will lead to the construction of a saturated tableau within finitely many steps.*

Proof. We show that no expansion rule can be applied infinitely many times within a branch, which proves that we can never construct an infinite branch. Let $\mathcal{T}$ be a tableau and let $\mathcal{S}$ be the root node of $\mathcal{T}$. As there are only finitely many profiles and finitely many outcome vectors, there are only finitely many aggregation rules. This follows from the fact that every aggregation rule consists of the election of a nonempty subset of outcome vectors for every profile. We conclude that $|\mathbb{I}(\mathcal{S})|$ is finite.

First, note that the axiom-driven expansion rule, the constraint-driven expansion rule and the branching rule can only be applied under the condition that $\mathbb{I}(S') \subsetneq \mathbb{I}(S)$, where a branch ending in the leaf node S is expanded with the node S'. As $|\mathbb{I}(\mathcal{S})|$ is finite and for every such application of one of these rules, we have that $|\mathbb{I}(S')| < |\mathbb{I}(S)|$, we conclude that none of the three may be used infinitely many times in a branch.

For a leaf node S in $\mathcal{T}$, we define the set of all universal outcome statements in S as follows:

$$[\![S]\!] := \{s \in S \mid s = [\![\mathbf{B}, \varphi]\!] \text{ for some } \mathbf{B} \in \mathrm{Mod}(\Gamma)^+ \text{ and } \varphi \in \mathcal{L}(X)\}.$$

Now we make the following remarks.

- $|[\![\mathcal{S}]\!]|$ is finite, as $|\mathcal{S}|$ is finite by definition.
- For any expansion of the branch ending in S via the simplification rule (by adding the node S'), we have that $|[\![S']\!]| = |[\![S]\!]| - 1$. This follows from the simple fact that we collapse two universal outcome statements into one when applying this rule.
- For any application of the rewrite rule, the witness rule or the refinement rule, we have that $|[\![S']\!]| = |[\![S]\!]|$, as the first rule only rewrites a formula in some outcome statement in S and the other two rules only alter the subset of existential outcome statements.
- For any application of the axiom-driven expansion rule, the constraint-driven expansion rule or the branching rule we do get that $|[\![S']\!]| = |[\![S]\!]| + 1$, but they can only be applied finitely many times, as explained above.

If we tie all of the above together, we conclude that in order to apply the simplification rule infinitely many times in a branch, we would need to apply one or more of the axiom-driven expansion rule, constraint-driven expansion rule or branching rule infinitely many times, which is not possible, as shown before. We conclude that we can never have infinitely many applications of the simplification rule in a branch.

Now we have proven that the first four rules can only have finitely many occurrences in any branch. It is left to show that the same holds for the other three. We make the following observation: because the first four rules can only produce finitely many universal outcome statements and any formula can only be rewritten to a shorter equivalent formula finitely many times, the rewrite rule can only introduce finitely many more universal outcome statements by rewriting the formulas in the universal outcome statements introduced by the first four rules. Note here that the witness rule and the refinement rule cannot introduce any new universal outcome statements.

For the witness rule, observe that we may only apply it at most once for every universal outcome statement that is introduced in the branch: for $[\![\mathbf{B}, \varphi]\!]$ and $[\![\mathbf{B}, \psi]\!]$ with $\varphi \equiv \psi$, we may only apply it once, and no existential outcome statement that is introduced can be deleted; it can be rewritten however, but the precondition of the witness rule makes sure that we cannot introduce any equivalent existential outcome statements.

Now we note that the only way to introduce new universal outcome statements is through any of the first five rules, but we have just proven that all of them together can only introduce finitely many universal outcome statements in a branch. We conclude that we will never be able to apply the witness rules infinitely many times in a branch, as it would require an infinite amount of universal outcome statements to be introduced in that branch.

Now we show that in any branch, we may only apply the refinement rule finitely many times. Define the set of existential outcome statements in S, denoted by $\langle\!\langle S \rangle\!\rangle$, analogous to the set of universal outcome statements in S. Now we partition this set into equivalence classes with the following equivalence relation:

$$\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle \sim \langle\!\langle \mathbf{B}', \psi \rangle\!\rangle \iff \mathbf{B} = \mathbf{B}' \text{ and } \varphi \equiv \psi.$$

The fact that this is indeed an equivalence relation follows immediately from the fact that '$\equiv$' is an equivalence relation for the formulas in $\mathcal{L}(X)$. We then define equivalence classes and the set of all equivalence classes:

$$[\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle] := \{s \in \mathbb{S} \mid s \sim \langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle\} \text{ and } \langle\!\langle S \rangle\!\rangle_\sim := \{[\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle] \mid \exists s \in S \text{ such that } s \sim \langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle\},$$

where we note that for $\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle \sim \langle\!\langle \mathbf{B}', \psi \rangle\!\rangle$, we have that $[\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle] = [\langle\!\langle \mathbf{B}', \psi \rangle\!\rangle]$. Now we observe the following.

- The domain $\mathrm{Mod}(\Gamma)^+$ is finite and there are finitely many formulas in $\mathcal{L}(X)$ up to equivalence; the latter follows from the fact that there are finitely many valuations on X and every formula $\varphi \in \mathcal{L}(X)$ can be represented by the subset of valuations on X under which φ is true. As a consequence, the set $\langle\!\langle \mathbb{S} \rangle\!\rangle_\sim$ is finite.

- For any expansion of the branch ending in S with the node S', we find that $\langle\!\langle S \rangle\!\rangle_\sim \subseteq \langle\!\langle S' \rangle\!\rangle_\sim$. This follows from the fact that the only rule that actually deletes an existential outcome statement is the rewrite rule, but that rule also introduces an equivalent existential statement, so for the application of the rewrite rule we find that $\langle\!\langle S \rangle\!\rangle_\sim = \langle\!\langle S' \rangle\!\rangle_\sim$.

- For any expansion of the branch ending in S with the node S' through the application of the refinement rule, we have that $\langle\!\langle S \rangle\!\rangle_\sim \subsetneq \langle\!\langle S' \rangle\!\rangle_\sim$. This is a direct consequence of the precondition of the rule, that states that we cannot introduce an existential outcome statement that is equivalent to an existential outcome statement in S.

We summarize the observations above. Firstly, the set $\langle\!\langle \mathbb{S} \rangle\!\rangle_\sim$ is finite. Secondly, no expansion rule will reduce the set of "existential equivalence classes". Finally, through the application of the refinement rule, we properly extend the set $\langle\!\langle S \rangle\!\rangle_\sim$, i.e. $\langle\!\langle S \rangle\!\rangle_\sim \subsetneq \langle\!\langle S' \rangle\!\rangle_\sim$. Now we note that for any node S, we can only properly extend the set $\langle\!\langle S \rangle\!\rangle_\sim$ finitely many times, as $\langle\!\langle S \rangle\!\rangle_\sim \subseteq \langle\!\langle \mathbb{S} \rangle\!\rangle_\sim$ and $\langle\!\langle \mathbb{S} \rangle\!\rangle_\sim$ is finite. We conclude that in any branch, we may only apply the refinement rule finitely many times.

Finally, we note that in any branch, all the other rules (except the rewrite rule) can only be applied finitely many times, so they can only introduce finitely many new outcome statements in that branch. Since any formula in an outcome statement can only be rewritten finitely many times to a shorter equivalent formula, we find that the rewrite rule can also only be applied finitely many times in that branch, which concludes our proof. □

3.3.2 Soundness

Now that we have shown termination, we conclude the chapter by proving soundness and completeness.

Recall that the tableau method described in this chapter offers a way to prove that there is no consistent aggregation rule that satisfies a set of outcome statements (in combination with a set of axiom instances). This is done by taking this set of outcome statements as a root node and constructing a tableau that is closed, which shows that there is no consistent aggregation rule that satisfies our initial set of outcome statements in combination with the axiom instances used for the applications of the axiom-driven expansion rule. Every branch can be seen as a sequence of argumentation steps, where the branching rule acts as a case distinction. A closed tableau then corresponds to finding a contradiction for every case in the case distinction. In this context, soundness should then say the following: if we can construct such a closed tableau for a set $\mathcal{S}$, then there should be no consistent aggregation rule satisfying this set (and the relevant set of axiom instances). Now we formalize and prove this.

Theorem 3.3 (Soundness)**.** *If we can find a closed tableau $\mathcal{T}$ that is rooted in $\mathcal{S}$ and licensed by $\mathcal{A}$, then there is no consistent aggregation rule that satisfies the outcome statements in $\mathcal{S}$ and the axiom instances in $\mathcal{A}$.*

Proof. We follow the proof idea of Boixel et al. [BEH22].

We prove the contrapositive. Suppose there is a consistent aggregation rule F that satisfies the outcome statements in $\mathcal{S}$ and the axiom instances in $\mathcal{A}$. Then we need to show that we cannot construct a closed tableau that is licensed by $\mathcal{A}$ and rooted in $\mathcal{S}$.

First, observe that F satisfies the outcome statements in $\mathcal{S}$ by assumption, so $F \in \mathbb{I}(\mathcal{S})$. Consequently, we show that every possible expansion rule applied to a node S with $F \in \mathbb{I}(S)$ will introduce at least one other node S' such that $F \in \mathbb{I}(S')$. This implies that any tableau rooted in $\mathcal{S}$ and licensed by $\mathcal{A}$ will have at least one branch ending in a leaf node S'' such that $F \in \mathbb{I}(S'')$, so this leaf node cannot contain any inconsistent outcome statements, as we would then have that $\mathbb{I}(S'') = \varnothing$. This proves that we will never be able to construct a closed tableau that is rooted in $\mathcal{S}$ and licensed by $\mathcal{A}$, as that would require all the leaf nodes in the tableau to contain at least one inconsistent outcome statement.

Suppose we are in a node S in the tableau with $F \in \mathbb{I}(S)$; we consider the seven possible expansion rules and show that each of them introduces at least one new node S' such that $F \in \mathbb{I}(S')$.

- **Axiom-driven expansion rule:** take any axiom instance $A \in \mathcal{A}$. Because F satisfies both the outcome statements in S and the axiom instances in $\mathcal{A}$ by assumption, we know that $F \in \mathbb{I}(S) \cap \mathbb{I}(A)$ (as $A \in \mathcal{A}$). But for any outcome statement s that we would add through the application of the axiom-driven expansion rule, we require that $\mathbb{I}(S) \cap \mathbb{I}(A) \subseteq \mathbb{I}(s)$, so for any such statement we find that $F \in \mathbb{I}(s)$. This implies that for the newly added node $S' = S \cup \{s\}$, we have that $F \in \mathbb{I}(S')$.

- **Constraint-driven expansion rule:** consider a propositional formula $\varphi \in \mathcal{L}(X)$ such that $\Gamma \vDash \varphi$. As F is consistent, we know that for any profile $\mathbf{B} \in \mathrm{Mod}(\Gamma)^+$, we have that $F \vDash [\![\mathbf{B}, \Gamma]\!]$. But as Γ (classically) entails φ, we immediately find that $F \vDash [\![\mathbf{B}, \varphi]\!]$ holds as well; note here that any outcome vector in $F(\mathbf{B})$ satisfies Γ when seen as a valuation, and as $\Gamma \vDash \varphi$, this vector also satisfies φ. We conclude that for any outcome statement $[\![\mathbf{B}, \varphi]\!]$ that is added

through the application of the constraint-driven expansion rule, we find that $F \in \mathbb{I}(S')$, with $S' = S \cup \{[\![\mathbf{B}, \varphi]\!]\}$.

- **Branching rule:** for any profile $\mathbf{B} \in \mathrm{Mod}(\Gamma)^+$ and formula $\varphi \in \mathcal{L}(X)$, we either have that $F \vDash [\![\mathbf{B}, \varphi]\!]$ or $F \vDash \langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle$ (the latter being equivalent to $F \nvDash [\![\mathbf{B}, \varphi]\!]$). That means that for $S' = S \cup \{[\![\mathbf{B}, \varphi]\!]\}$ and $S'' = S \cup \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}$, we either have that $F \in \mathbb{I}(S')$ or $F \in \mathbb{I}(S'')$, where $F \in \mathbb{I}(S)$ already holds by assumption. For the other possible application of the branching rule the proof is analogous.

- **Simplification rule:** for any application of the simplification rule it holds that $\mathbb{I}(S) = \mathbb{I}(S')$, where S' is the node that is added to the branch ending in the leaf node S through the application of the simplification rule. This follows from the fact that for any aggregation rule F and any outcome statements $[\![\mathbf{B}, \varphi]\!], [\![\mathbf{B}, \psi]\!]$ and $[\![\mathbf{B}, \varphi \wedge \psi]\!]$, it holds that $F \vDash [\![\mathbf{B}, \varphi]\!]$ and $F \vDash [\![\mathbf{B}, \psi]\!]$ if and only if $F \vDash [\![\mathbf{B}, \varphi \wedge \psi]\!]$. We conclude that $F \in \mathbb{I}(S')$.

- **Rewrite rule:** it is immediately evident that for any application of the rewrite rule we have that $\mathbb{I}(S) = \mathbb{I}(S')$, where S' is the newly added node with one formula in one outcome statement rewritten. We again conclude that $F \in \mathbb{I}(S')$.

- **Witness rule:** suppose that $S' = S \cup \{\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle\}$ with $[\![\mathbf{B}, \varphi]\!] \in S$, where S' is the node added through the application of the witness rule to the branch ending in S. Then as we have that $F \in \mathbb{I}(S)$, every outcome vector in $F(\mathbf{B})$ satisfies φ. Since $F(\mathbf{B}) \neq \varnothing$ by definition, we conclude that there exists an outcome vector in $F(\mathbf{B})$ that satisfies φ, so $F \in \mathbb{I}(\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle)$. Then, as $F \in \mathbb{I}(S)$ holds by assumption, we conclude that $F \in \mathbb{I}(S \cup \{\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle\}) = \mathbb{I}(S')$.

- **Refinement rule:** let $S' = S \cup \{\langle\!\langle \mathbf{B}, \varphi \wedge \psi \rangle\!\rangle\}$, with $\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle, [\![\mathbf{B}, \psi]\!] \in S$, where S' is added through the application of the refinement rule. Because it holds that $F \in \mathbb{I}(S)$, every outcome vector in $F(\mathbf{B})$ satisfies ψ and there exists an outcome vector in $F(\mathbf{B})$ that satisfies φ. Then it immediately follows that there exists an outcome vector in $F(\mathbf{B})$ that satisfies $\varphi \wedge \psi$, namely the outcome vector that already satisfies φ. We conclude that $F \in \mathbb{I}(S')$.

We conclude the following: for the root node $\mathcal{S}$ we have that $F \in \mathbb{I}(\mathcal{S})$ and for any application of an expansion rule (with axiom instances from $\mathcal{A}$) expanding a branch ending in a leaf node S such that $F \in \mathbb{I}(S)$, we find that there exists at least one newly added node S' such that $F \in \mathbb{I}(S')$. This implies that for any tableau rooted in $\mathcal{S}$ and licensed by $\mathcal{A}$, there is at least one branch ending in a leaf node S'' such that $F \in \mathbb{I}(S'')$, which means that this leaf node does not contain any inconsistent outcome statements, so we will never be able to construct a closed tableau. This concludes our proof. □

3.3.3 Completeness

Before we prove completeness, we prove two auxiliary lemmas. The first lemma concerns the idea that the set of satisfying aggregation rules narrows down as you follow a branch from the root node to the leaf node.

Lemma 3.4. *In a tableau $\mathcal{T}$, if S' is a child node of S (so there is an edge from S to S'), then we have that $\mathbb{I}(S') \subseteq \mathbb{I}(S)$.*

Proof. We show that for any application of one of the rules, we find that $\mathbb{I}(S') \subseteq \mathbb{I}(S)$.

- If S' is added through the application of any of the first three rules, we immediately find that $\mathbb{I}(S') \subsetneq \mathbb{I}(S)$, as this is a precondition of those rules.
- For the simplification rule, we make the same observation as in the soundness proof: for any profile $\mathbf{B} \in \mathrm{Mod}(\Gamma)^+$, any aggregation rule F and any two formulas $\varphi, \psi \in \mathcal{L}(X)$, it holds that $F \vDash [\![\mathbf{B}, \varphi]\!]$ and $F \vDash [\![\mathbf{B}, \psi]\!]$ if and only if $F \vDash [\![\mathbf{B}, \varphi \wedge \psi]\!]$. We conclude that for $S' = \big(S \setminus \{[\![\mathbf{B}, \varphi]\!], [\![\mathbf{B}, \psi]\!]\}\big) \cup \{[\![\mathbf{B}, \varphi \wedge \psi]\!]\}$, we find that $\mathbb{I}(S') = \mathbb{I}(S)$.
- For the rewrite rule, it is immediately clear that $\mathbb{I}(S') = \mathbb{I}(S)$.
- Note that for the witness rule, any aggregation rule that satisfies the outcome statements in the child node S' will also satisfy the statements in S, as $S \subsetneq S'$. We conclude that $\mathbb{I}(S') \subseteq \mathbb{I}(S)$.
- Finally, for the refinement rule, we analogously find that $\mathbb{I}(S') \subseteq \mathbb{I}(S)$ as a consequence of the fact that $S \subsetneq S'$.

This concludes our proof. □

Before we get to the proof of completeness, we prove another auxiliary lemma. This lemma proves an important property of saturated tableaux: any inconsistent leaf node in a saturated tableau should contain an inconsistent outcome statement. Earlier, we discussed the fact that an inconsistent node need not contain an inconsistent outcome statement; take the node $S = \{[\![\mathbf{B}, x_1]\!], \langle\!\langle \mathbf{B}, \neg x_1 \rangle\!\rangle\}$ for example. The idea of this lemma is that we can always "trace down" the inconsistency through the application of the expansion rules. For example, in the case of S we could apply refinement and find the statement $\langle\!\langle \mathbf{B}, \neg x_1 \wedge x_1 \rangle\!\rangle$, then rewriting would grant us the inconsistent outcome statement $\langle\!\langle \mathbf{B}, \bot \rangle\!\rangle$. In a saturated tableau, we cannot apply any more rules, so this "tracing down" should then already have happened.

Lemma 3.5. *In a saturated tableau $\mathcal{T}$, any inconsistent leaf node contains an inconsistent outcome statement.*

Proof. Let $\mathcal{T}$ be a saturated tableau and suppose for contradiction that there is a branch ending in a leaf node S such that $\mathbb{I}(S) = \varnothing$ while S contains no inconsistent outcome statement.

Note that this means that S contains no outcome statements of the form $\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle$ or $[\![\mathbf{B}, \varphi]\!]$ with $\varphi \equiv \bot$, otherwise we could apply the rewrite rule to S, contradicting the fact that $\mathcal{T}$ is saturated.

Let $S_{\mathbf{B}}$ be the set of outcome statements in S that make a statement concerning profile $\mathbf{B} \in \mathrm{Mod}(\Gamma)^+$, so $S_{\mathbf{B}} = \{s \in S \mid s = \langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle \text{ or } s = [\![\mathbf{B}, \varphi]\!] \text{ for some } \varphi \in \mathcal{L}(X)\}$. Now note that there is at least one profile $\mathbf{B}$ such that $\mathbb{I}(S_{\mathbf{B}}) = \varnothing$. Suppose it is not the case, then we can simply construct an aggregation rule F as follows: for any profile $\mathbf{B} \in \mathrm{Mod}(\Gamma)^+$, take some $F' \in \mathbb{I}(S_{\mathbf{B}})$ (which exists by assumption) and let $F(\mathbf{B}) := F'(\mathbf{B})$. Then as $S = \cup_{\mathbf{B} \in \mathrm{Mod}(\Gamma)^+} S_{\mathbf{B}}$, we conclude that $F \in \mathbb{I}(S)$, which contradicts the fact that $\mathbb{I}(S) = \varnothing$. We conclude that there exists a profile $\mathbf{B} \in \mathrm{Mod}(\Gamma)^+$ such that $\mathbb{I}(S_{\mathbf{B}}) = \varnothing$.

Consequently, we show that $S_{\mathbf{B}}$ contains exactly one universal outcome statement. If it contains more than one universal outcome statement, we could apply the simplification rule, contradicting the fact that $\mathcal{T}$ is saturated and S is a leaf node in $\mathcal{T}$.

If $S_{\mathbf{B}}$ does not contain any universal outcome statements, we can construct an aggregation rule F that satisfies the statements in $S_{\mathbf{B}}$ as follows. For any profile $\mathbf{B}' \in \mathrm{Mod}(\Gamma)^+$ other than $\mathbf{B}$: pick an arbitrary outcome $F(\mathbf{B}') \in 2^{\{0,1\}^m} \setminus \{\varnothing\}$. For $\mathbf{B}$: let $F(\mathbf{B})$ be a set of outcome vectors such

that for every (existential) outcome statement in $S_{\mathbf{B}}$ of the form $\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle$, there is a $v \in F(\mathbf{B})$ such that $v \vDash \varphi$; recall here that we have that $\varphi \not\equiv \bot$. Then it is immediately clear that $F \in \mathbb{I}(S_{\mathbf{B}})$, which contradicts the fact that $\mathbb{I}(S_{\mathbf{B}}) = \varnothing$. We conclude that in addition, $S_{\mathbf{B}}$ must contain at least one universal outcome statement, so $S_{\mathbf{B}}$ should contain exactly one universal outcome statement.

Now we have found that $\mathbb{I}(S_{\mathbf{B}}) = \varnothing$ while $S_{\mathbf{B}}$ contains exactly one universal outcome statement, and no inconsistent outcome statements. Let $S'_{\mathbf{B}}$ be the set of the existential outcome statements in $S_{\mathbf{B}}$, then we can prove that $\mathbb{I}(S'_{\mathbf{B}}) \neq \varnothing$, by again constructing an aggregation rule F such that $F(\mathbf{B})$ contains a vector satisfying φ for each statement $\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle \in S'_{\mathbf{B}}$, as we have shown in the paragraph above. This means that the inconsistency of the set $S_{\mathbf{B}}$ is related to the single universal outcome statement in $S_{\mathbf{B}}$. Note that by assumption, we cannot have that this statement is of the form $[\![\mathbf{B}, \bot]\!]$, and it can also not be of the form $[\![\mathbf{B}, \varphi]\!]$ with $\varphi \equiv \bot$; otherwise we could apply the rewrite rule. This implies that for $[\![\mathbf{B}, \psi]\!] \in S_{\mathbf{B}}$, we have that $\mathbb{I}([\![\mathbf{B}, \psi]\!]) \neq \varnothing$.

We now claim the following: there exists an existential outcome statement $\langle\!\langle \mathbf{B}, \rho \rangle\!\rangle \in S'_{\mathbf{B}}$ such that $\mathbb{I}(\{[\![\mathbf{B}, \psi]\!], \langle\!\langle \mathbf{B}, \rho \rangle\!\rangle\}) = \varnothing$. For contradiction, suppose that for every $s \in S'_{\mathbf{B}}$, we have that $\mathbb{I}(\{[\![\mathbf{B}, \psi]\!], s\}) \neq \varnothing$. That means that for every existential outcome statement of the form $\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle \in S'_{\mathbf{B}}$, there is an aggregation rule $F' \in \mathbb{I}(\{[\![\mathbf{B}, \psi]\!], \langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle\})$, so there is an outcome $v \in F'(\mathbf{B})$ such that $v \vDash \varphi$ and $v \vDash \psi$. But then we can again define an aggregation rule that will satisfy $S_{\mathbf{B}}$: let F be arbitrarily defined on every profile other than $\mathbf{B}$ and let $F(\mathbf{B})$ be the union of all these outcome vectors v for each existential outcome statement in $S'_{\mathbf{B}}$, as described above. Then it holds that $F \in \mathbb{I}(S_{\mathbf{B}})$, as all vectors also satisfy ψ, and for every existential outcome statement in $S_{\mathbf{B}}$, there is an outcome vector that satisfies the formula of that outcome statement. This contradicts the fact that $\mathbb{I}(S_{\mathbf{B}}) = \varnothing$. We conclude that there exists an existential outcome statement $\langle\!\langle \mathbf{B}, \rho \rangle\!\rangle \in S'_{\mathbf{B}}$ such that $\mathbb{I}(\{[\![\mathbf{B}, \psi]\!], \langle\!\langle \mathbf{B}, \rho \rangle\!\rangle\} = \varnothing$.

But this implies that $\rho \wedge \psi \equiv \bot$, otherwise $\mathbb{I}(\{[\![\mathbf{B}, \psi]\!], \langle\!\langle \mathbf{B}, \rho \rangle\!\rangle\}) \neq \varnothing$, as we could have an aggregation rule F where $F(\mathbf{B})$ is a singleton containing a vector that satisfies both ψ and ρ. But then, as shown earlier, we cannot have that $\langle\!\langle \mathbf{B}, \rho \wedge \psi \rangle\!\rangle \in S$; otherwise we would be able to rewrite this statement to an inconsistent outcome statement. In general, for any formula $\chi \in \mathcal{L}(X)$ such that $\chi \equiv \rho \wedge \psi$, we have that $\chi \equiv \bot$ as $\rho \wedge \psi \equiv \bot$. As a consequence, S cannot contain an existential outcome statement of the form $\langle\!\langle \mathbf{B}, \chi \rangle\!\rangle$ for any χ with $\chi \equiv \rho \wedge \varphi$. This means that we are able to apply the refinement rule to obtain a new outcome statement $\langle\!\langle \mathbf{B}, \rho \wedge \psi \rangle\!\rangle$, contradicting the fact that S is a leaf node in a saturated tableau, which concludes our proof. □

Having proven the second auxiliary lemma, we are ready to state and prove completeness.

Theorem 3.6 (Completeness)**.** *If there is no consistent aggregation rule that satisfies both the outcome statements in $\mathcal{S}$ and the axiom instances in $\mathcal{A}$, then we can construct a tableau $\mathcal{T}$ that is closed, rooted in $\mathcal{S}$ and licensed by $\mathcal{A}$.*

Proof. Again, we follow the general proof idea of Boixel et al. [BEH22] but adapt it to our own model. We do a proof by contraposition.

Suppose we cannot construct a closed tableau that is rooted in $\mathcal{S}$ and licensed by $\mathcal{A}$. We want to prove that there exists a consistent aggregation rule that satisfies both the outcome statements in $\mathcal{S}$ and the axiom instances in $\mathcal{A}$.

Take any (finite) open saturated tableau $\mathcal{T}$ that is rooted in $\mathcal{S}$ and licensed by $\mathcal{A}$, which exists as a consequence of our assumption and termination. Lemma 3.5 now gives us that any inconsistent node also contains an inconsistent outcome statement, and as $\mathcal{T}$ is open, it should contain at least one consistent leaf node S.

As S is consistent, we have that $\mathbb{I}(S) \neq \varnothing$. Now consider any $F \in \mathbb{I}(S)$. We want to show that this F is consistent, satisfies the outcome statements in $\mathcal{S}$ and satisfies the axiom instances in $\mathcal{A}$.

- Firstly, suppose that F is not consistent. Then there is a profile $\mathbf{B} \in \mathrm{Mod}(\Gamma)^+$ such that $F(\mathbf{B})$ contains an outcome that does not satisfy Γ. But then $F \nvDash [\![\mathbf{B}, \Gamma]\!]$ so we can apply the constraint-driven expansion rule and expand S with a node $S' = S \cup \{[\![\mathbf{B}, \Gamma]\!]\}$, which is a valid expansion as $F \notin \mathbb{I}(S')$, so $\mathbb{I}(S') \subsetneq \mathbb{I}(S)$ (note here that $\mathbb{I}(S') \subseteq \mathbb{I}(S)$ trivially holds as $S \subseteq S'$). This contradicts the fact that S is a leaf node in a saturated tableau $\mathcal{T}$. We conclude that F is consistent.

- Now we show that F satisfies all the outcome statements in $\mathcal{S}$. Note that by Lemma 3.4, we have that $\mathbb{I}(S') \subseteq \mathbb{I}(S)$ if S' is a child node of S. Also note that there is a (finite) path from the root node $\mathcal{S}$ to the leaf node S in $\mathcal{T}$. These two facts give us that $\mathbb{I}(S) \subseteq \mathbb{I}(\mathcal{S})$ and as $F \in \mathbb{I}(S)$, we then find that $F \in \mathbb{I}(\mathcal{S})$, which proves that F satisfies all the outcome statements in $\mathcal{S}$.

- Before we show that F satisfies all the axiom instances in $\mathcal{A}$, we prove that $|\mathbb{I}(S)| = 1$, so F is in fact the only aggregation rule that satisfies the outcome statements in S. For contradiction, suppose that there is another aggregation rule $F' \in \mathbb{I}(S)$ such that $F \neq F'$. That means that there is a profile $\mathbf{B} \in \mathrm{Mod}(\Gamma)^+$ such that $F(\mathbf{B}) \neq F'(\mathbf{B})$. Then we either have that $F(\mathbf{B}) \nsubseteq F'(\mathbf{B})$ or $F(\mathbf{B}) \nsubseteq F'(\mathbf{B})$. Suppose without loss of generality that $F(\mathbf{B}) \nsubseteq F'(\mathbf{B})$ (the other case is proven analogously). Then there is an outcome vector $v \in F(\mathbf{B})$ such that $v \notin F'(\mathbf{B})$. Now we note that $F \nvDash [\![\mathbf{B}, \varphi_{F'(\mathbf{B})}]\!]$ (see Definition 2.13), as the outcome $v \in F(\mathbf{B})$ does not satisfy any of the clauses in $\varphi_{F'(\mathbf{B})}$ as it is not equal to any of the vectors in $F'(\mathbf{B})$ (this follows from the contrapositive of Lemma 2.5), so $v \nvDash \varphi_{F'(\mathbf{B})}$. Additionally, we have that $F' \nvDash \langle\!\langle \mathbf{B}, \neg\varphi_{F'(\mathbf{B})} \rangle\!\rangle$, as there is no outcome in $F'(\mathbf{B})$ that does not satisfy $\varphi_{F'(\mathbf{B})}$ by definition. This means that we are able to apply the branching rule for the profile $\mathbf{B}$ and the formula $\varphi_{F'(\mathbf{B})}$: let $S' = S \cup \{[\![\mathbf{B}, \varphi_{F'(\mathbf{B})}]\!]\}$ and $S'' = S \cup \{\langle\!\langle \mathbf{B}, \neg\varphi_{F'(\mathbf{B})} \rangle\!\rangle\}$, then we have that $F \notin \mathbb{I}(S')$ while $F' \notin \mathbb{I}(S'')$, so $\mathbb{I}(S') \subsetneq \mathbb{I}(S)$ and $\mathbb{I}(S'') \subsetneq \mathbb{I}(S)$ (which allows us to apply the branching rule). This contradicts the fact that $\mathcal{T}$ is saturated. We conclude that F is the only aggregation rule in $\mathbb{I}(S)$.

- Finally, suppose for contradiction that there is an axiom instance $A \in \mathcal{A}$ such that $F \nvDash A$. Then we may conclude that $\mathbb{I}(S) \cap \mathbb{I}(A) = \varnothing$, as we have just proved that $\mathbb{I}(S) = \{F\}$. Now take any profile $\mathbf{B} \in \mathrm{Mod}(\Gamma)^+$, then we have that $\mathbb{I}(S) \cap \mathbb{I}(A) \subseteq \mathbb{I}([\![\mathbf{B}, \bot]\!])$, so $S, \{A\} \vDash [\![\mathbf{B}, \bot]\!]$. Also note that for $S' = S \cup \{[\![\mathbf{B}, \bot]\!]\}$, we have that $\mathbb{I}(S') = \varnothing$, so $\mathbb{I}(S') \subsetneq \mathbb{I}(S)$. This means that we may apply the axiom-driven expansion rule to S for the axiom instance $A \in \mathcal{A}$ and the outcome statement $[\![\mathbf{B}, \bot]\!]$, which contradicts the fact that S is a leaf node in a saturated tableau $\mathcal{T}$. We conclude that $F \vDash A$ for all $A \in \mathcal{A}$, so $F \vDash \mathcal{A}$.

Now we have proven that there exists a consistent aggregation rule that satisfies both the outcome statements in $\mathcal{S}$ and the axiom instances in $\mathcal{A}$, which concludes our proof. □

From soundness and completeness, it follows that the calculus described above is in fact correct.

Corollary 3.7 (Correctness)**.** *Let $\mathcal{S}$ be a set of outcome statements and $\mathcal{A}$ be a set of axiom instances. Then we can construct a closed tableau $\mathcal{T}$ that is rooted in $\mathcal{S}$ and licensed by $\mathcal{A}$ if and only if there is no consistent aggregation rule satisfying the outcome statements in $\mathcal{S}$ and the axiom instances in $\mathcal{A}$.*

3.4 Discussion

In this chapter, we defined a tableau-based calculus that allows us to construct a structured proof for the fact that there is no consistent aggregation rule that satisfies a set of outcome statements together with a set of axiom instances. The model is based on the semantics of the outcome statements defined in Chapter 2. The idea of the tableau method is based on the work of Boixel et al. [BEH22]. Although the general approach still coincides, our definition of an outcome statement differs from theirs, and we have defined a calculus for binary aggregation instead of voting theory, so our model has turned out quite differently. Boixel et al. propose three expansion rules: the axiom-driven expansion rule, the branching rule and the simplification rule. We offer three alternatives for these rules for the framework of binary aggregation, and we propose four extra expansion rules. The constraint-driven expansion rule incorporates the integrity constraint into the tableaux. The rewrite rule accounts for the fact that our outcome statements are of a more descriptive kind and require the use of propositional formulas. Finally, the witness rule and refinement rule resemble inferences that one would make, based on the semantics of outcome statements. These last two rules also offer extra interaction between the universal and existential outcome statements, which can sometimes be helpful when constructing a tableau, as will become clear in some of the examples in the next chapter.

The exact preconditions for the different expansion rules are all carefully defined to make sure that soundness, completeness and termination hold. This is especially clear in the proof of termination, where these preconditions are often mentioned. We have also made sure that we will never lose any information in the application of one of the expansion rules. For example, in the witness rule we do not remove the universal outcome statement, otherwise Lemma 3.4 would not hold. There are many other expansion rules one could come up with, and we certainly do not claim that the set of expansion rules defined in Definition 3.2 is the "best" set of expansion rules to adopt. However, in the above we have shown that each expansion rule is selected for a particular reason, and its technical details are carefully considered. Moreover, we have shown that this set makes for a sound, complete and terminating calculus. In the next chapter, we will see the tableaux in action, which will show that it is a calculus that is easy to work with.

The outcome statements work well within the tableau-based calculus, as they are compact and intuitive. Both the language of the outcome statements and the tableau method are dynamical frameworks: they are easily expanded, adapted, or rewritten. This resemblance offers for a smooth interaction between both.

When considering the explainability and transparency of our model, we note two things. Firstly, the rewrite rule might perform a very extensive rewrite that is hard to grasp. In that sense, a big part of the inference may happen within a single application of the rewrite rule. Because of that, one could argue that this rule is too strong and offers too little transparency. However, in this chapter we have defined a general calculus that is sound and complete. It might be interesting to adapt the rewrite rule to force more transparent inferences, e.g. we might restrict the possible rewrite steps.

Another rule that might cover a complex inference is the axiom-driven expansion rule. It is sometimes not so trivial why some outcome statement is forced by a set of outcome statements and an axiom instance. However, when actually implementing a model to generate such justifications in Chapter 6, we will provide a method that is transparent to some extent.

Chapter 4

Justifications

In this chapter, we formally define justifications. First, we define *unstructured justifications*; Boixel and Endriss [BE20] defined these already for voting theory. Section 4.1 is based on their work, but we will adapt it to the framework of binary aggregation and use the outcome statements defined in Chapter 2. An unstructured justification for some outcome statement $[\![\mathbf{B}, \varphi]\!]$ is a set of axioms and a set of instances of those axioms that explains why every consistent aggregation rule that satisfies the instances also should satisfy $[\![\mathbf{B}, \varphi]\!]$, i.e. the outcome statement $[\![\mathbf{B}, \varphi]\!]$ is a consequence of the set of axiom instances.

Consequently, we show how to use the tableau method from the previous chapter to define *structured justifications*, where we add a tableau to an unstructured justification in order to structure the justification. Again, we follow the main ideas from the work of Boixel et al. [BEH22].

4.1 Unstructured Justifications

In this section, we will define unstructured justifications. As described above, an unstructured justification is a set of axioms together with a set of axiom instances that explains why a certain outcome statement $[\![\mathbf{B}, \varphi]\!]$ should hold. The justification is called unstructured, as there is no guide how to read this set; it might be complicated to explain why from this set of axiom instances it follows that $[\![\mathbf{B}, \varphi]\!]$ should hold.

The definition of an unstructured justification makes use of three different sets. Before we state the definition, we introduce these sets.

Definition 4.1. *The* ***corpus*** *of axioms $\mathbb{A}$ is the set of all axioms that are collectively accepted by the group of voters, i.e. each voter deems the axioms in $\mathbb{A}$ to be fair and acceptable. The* ***normative basis*** *is a subset $\mathcal{A}^N \subseteq \mathbb{A}$ of axioms that are relevant for the specific justification. Finally, the* ***explanatory basis*** *$\mathcal{A}^E$ is the set of axiom instances from the axioms in $\mathcal{A}^N$ that are actually used in the justification.*

The roles of these different concepts will become clear in the examples later in this section. Now we adapt the definition of an unstructured justification for voting theory by Boixel and Endriss [BE20] to binary aggregation.

Definition 4.2 (Unstructured Justification)**.** *Let $\mathbb{A}$ be a corpus of axioms and let $[\![\mathbf{B}, \varphi]\!] \in \mathbb{S}$ be a universal outcome statement. An* ***(unstructured) justification*** *for $[\![\mathbf{B}, \varphi]\!]$ is a pair $\langle \mathcal{A}^N, \mathcal{A}^E \rangle$*

consisting of a normative basis $\mathcal{A}^N$ and an explanatory basis $\mathcal{A}^E$ such that the following conditions hold:

- ***Explanatoriness:*** *$\mathcal{A}^E$ explains the outcome statement $[\![\mathbf{B}, \varphi]\!]$ while no proper subset does: we have that $\mathcal{A}^E \models_\Gamma [\![\mathbf{B}, \varphi]\!]$ while for every proper subset $\mathcal{A}' \subsetneq \mathcal{A}^E$ we find that $\mathcal{A}' \not\models_\Gamma [\![\mathbf{B}, \varphi]\!]$.*
- ***Relevance:*** *Every instance in $\mathcal{A}^E$ is an instance of an axiom in $\mathcal{A}^N$: $\mathcal{A}^E \lhd \mathcal{A}^N$ (see Definition 2.10).*
- ***Adequacy:*** *All axioms in $\mathcal{A}^N$ are in the corpus: $\mathcal{A}^N \subseteq \mathbb{A}$.*
- ***Nontriviality:*** *There exists a consistent aggregation rule F such that F satisfies the axioms in $\mathcal{A}^N$: $\mathbb{I}_\Gamma(\mathcal{A}^N) \neq \varnothing$.*

Remark 4.1. *Note here that we only allow for the justification of universal outcome statements. This is due to the fact that existential outcome statements are less descriptive. For example, if we would justify an outcome statement of the form $[\![\mathbf{B}, x_1]\!]$, then that would mean that we should accept the first issue, no matter the final outcome. However, justifying an existential outcome statement like $\langle\!\langle \mathbf{B}, x_1 \rangle\!\rangle$ says a lot less about the final outcome.*

Now we consider some examples to gain some intuition towards the definition above.

Example 4.1. *Let $n = 4$, $m = 2$ and $\Gamma = \top$. Let $\mathbf{B} = \{(1,0),(1,0),(0,1),(1,1)\}$ and suppose we want to justify the fact that every outcome should accept issue 1, given that all axioms introduced in Section 2.3 are collectively accepted. Formally, we then have that*

$$\mathbb{A} = \{\mathcal{A}^{Fai}, \mathcal{A}^{Hom}, \mathcal{A}^{Weak\text{-}Un}, \mathcal{A}^{Strong\text{-}Un}, \mathcal{A}^{Rei}, \mathcal{A}^{Mon}, \mathcal{A}^{Maj}, \mathcal{A}^{Can}\}$$

and we want to justify the outcome statement $[\![\mathbf{B}, x_1]\!]$. An example of an unstructured justification for this corpus and outcome statement would then be

$$\langle \mathcal{A}^N, \mathcal{A}^E \rangle = \langle \{\mathcal{A}^{Maj}\}, \{[\![\mathbf{B}, (x_1 \wedge x_2) \vee (x_1 \wedge \neg x_2)]\!]\} \rangle.$$

Note here that $ext(m(\mathbf{B})) = \{(1,1),(1,0)\}$, which is the set that the formula in the outcome statement in the axiom instance describes. We show that it is in fact a justification for $[\![\mathbf{B}, x_1]\!]$ by checking all the conditions.

- ***Explanatoriness:*** *Note that $\mathbb{I}_\Gamma(\mathcal{A}^E)$ is the set of consistent aggregation rules F for which $F(\mathbf{B}) \subseteq \{(1,1),(1,0)\}$; this follows from the fact that $F \vDash [\![\mathbf{B}, (x_1 \wedge x_2) \vee (x_1 \wedge \neg x_2)]\!]$ implies that any outcome in $F(\mathbf{B})$ should satisfy either $x_1 \wedge x_2$ or $x_1 \wedge \neg x_2$. Lemma 2.5 then gives us that any outcome in $F(\mathbf{B})$ is equal to $(1,1)$ or $(1,0)$. This means that for any such $F \in \mathbb{I}_\Gamma(\mathcal{A}^E)$, we have that $F \vDash [\![\mathbf{B}, x_1]\!]$. We conclude that $\mathcal{A}^E \models_\Gamma [\![\mathbf{B}, x_1]\!]$, where we note that any aggregation rule is consistent as $\Gamma = \top$.*

  Any proper subset of $\mathcal{A}^E$ is the empty set, and it is immediately clear that there exists an aggregation rule $F \in \mathbb{I}_\Gamma(\varnothing)$ for which $F \not\vDash [\![\mathbf{B}, x_1]\!]$. We conclude that explanatoriness is satisfied.

- ***Relevance:*** *It is immediately clear that*

$$\{[\![\mathbf{B}, (x_1 \wedge x_2) \vee (x_1 \wedge \neg x_2)]\!]\} \lhd \{\mathcal{A}^{Maj}\},$$

as the instance on the left is an instance of the axiom $\mathcal{A}^{Maj}$. We conclude that relevance is also satisfied.

- ***Adequacy:*** *This is also immediate: $\mathcal{A}^N = \{\mathcal{A}^{Maj}\} \subseteq \mathbb{A}$.*
- ***Nontriviality:*** *Finally, note that as $\Gamma = \top$, we have that*

$$\mathbb{I}_\Gamma(\mathcal{A}^N) = \mathbb{I}(\mathcal{A}^N),$$

and as $\mathbb{I}(\mathcal{A}^N)$ is clearly nonempty (the rule for which $F(\mathbf{B}) = ext(m(\mathbf{B}))$ is in $\mathbb{I}(\mathcal{A}^N)$ as $ext(m(\mathbf{B}))$ is nonempty for every $\mathbf{B} \in Mod(\Gamma)^+$), nontriviality is also satisfied.

In the following example, we adapt the example in the work of Boixel et al. [BEH22] to binary aggregation.

Example 4.2. *Let $n = 3$, $m = 2$ and $\Gamma = \top$. Suppose the majority-preservation axiom is not in the corpus, i.e. it is not collectively accepted by the group of voters. We find the following set:*

$$\mathbb{A} = \{\mathcal{A}^{Fai}, \mathcal{A}^{Hom}, \mathcal{A}^{Weak\text{-}Un}, \mathcal{A}^{Strong\text{-}Un}, \mathcal{A}^{Rei}, \mathcal{A}^{Mon}, \mathcal{A}^{Can}\}.$$

Now let $\mathbf{B} = \{(1,0),(0,1),(1,1)\}$ and suppose we want to justify the outcome statement $[\![\mathbf{B}, x_1 \wedge x_2]\!]$ from the normative basis $\mathcal{A}^N = \{\mathcal{A}^{Fai}, \mathcal{A}^{Can}, \mathcal{A}^{Rei}\}$. We could then come up with the following non-formal justification: an aggregation rule should elect every outcome for the sub-profile $\{(1,0),(0,1)\}$ because of cancellation, while it should only elect $(1,1)$ for the profile $\{(1,1)\}$ because of faithfulness. Reinforcement should then force that $F(\mathbf{B})$ is the intersection of the two outcomes, which is $\{(1,1)\}$. Formally, we have an unstructured justification $\langle \mathcal{A}^N, \mathcal{A}^E \rangle$ with $\mathcal{A}^N$ as stated above and the following explanatory basis, with $\mathbf{B}' = \{(1,1)\}$ and $\mathbf{B}'' = \{(1,0),(0,1)\}$:

$$\begin{aligned}
\mathcal{A}^E = \{&[\![\mathbf{B}', x_1 \wedge x_2]\!]\} \\
&\cup \{\langle\!\langle \mathbf{B}'', x_1 \wedge x_2 \rangle\!\rangle\} \\
&\cup \{\big(\langle\!\langle \mathbf{B}', x_1 \wedge x_2 \rangle\!\rangle \wedge \langle\!\langle \mathbf{B}'', x_1 \wedge x_2 \rangle\!\rangle\big) \\
&\to \big(((([\![\mathbf{B}', x_1 \wedge x_2]\!] \vee [\![\mathbf{B}'', x_1 \wedge x_2]\!]) \to [\![\mathbf{B}, x_1 \wedge x_2]\!]) \\
&\wedge ((\langle\!\langle \mathbf{B}', \varphi_w \rangle\!\rangle \wedge \langle\!\langle \mathbf{B}'', \varphi_w \rangle\!\rangle) \to \langle\!\langle \mathbf{B}, \varphi_w \rangle\!\rangle)\big)\},
\end{aligned}$$

where φ_w is the formula describing some arbitrary vector $w \in \{0,1\}^m$ (this will not be relevant for the justification). Note here that $\mathbf{B} = \mathbf{B}' + \mathbf{B}''$. Again, we check all the conditions to show that this is indeed an unstructured justification.

- ***Explanatoriness:*** *Let $F \in \mathbb{I}_\Gamma(\mathcal{A}^E)$. The first thing we note is that $F \vDash [\![\mathbf{B}', x_1 \wedge x_2]\!]$ so it also holds that $F \vDash \langle\!\langle \mathbf{B}', x_1 \wedge x_2 \rangle\!\rangle$, as $F(\mathbf{B}') \neq \varnothing$. We also find that $F \vDash \langle\!\langle \mathbf{B}'', x_1 \wedge x_2 \rangle\!\rangle$. This gives us that $F \vDash \langle\!\langle \mathbf{B}', x_1 \wedge x_2 \rangle\!\rangle \wedge \langle\!\langle \mathbf{B}'', x_1 \wedge x_2 \rangle\!\rangle$, so F satisfies the antecedent of the third axiom instance of $\mathcal{A}^E$ so F should satisfy the consequence as well:*

$$F \vDash (([\![\mathbf{B}', x_1 \wedge x_2]\!] \vee [\![\mathbf{B}'', x_1 \wedge x_2]\!]) \to [\![\mathbf{B}, x_1 \wedge x_2]\!]) \wedge ((\langle\!\langle \mathbf{B}', \varphi_w \rangle\!\rangle \wedge \langle\!\langle \mathbf{B}'', \varphi_w \rangle\!\rangle) \to \langle\!\langle \mathbf{B}, \varphi_w \rangle\!\rangle).$$

For this justification, we only care about the first part of the conjunction, so we have:

$$F \vDash ([\![\mathbf{B}', x_1 \wedge x_2]\!] \vee [\![\mathbf{B}'', x_1 \wedge x_2]\!]) \to [\![\mathbf{B}, x_1 \wedge x_2]\!].$$

Because it holds that $F \vDash [\![\mathbf{B}', x_1 \wedge x_2]\!]$ (as it is an outcome statement in $\mathcal{A}^E$), we find that $F \vDash [\![\mathbf{B}, x_1 \wedge x_2]\!]$, which is what we wanted to show.

For the second part of explanatoriness, note that without the third formula of $\mathcal{A}^E$, we can construct an aggregation rule F for which $F \nvDash [\![\mathbf{B}, x_1 \wedge x_2]\!]$ immediately: we can simply define any rule that does satisfy the first two formulas while $F(\mathbf{B})$ contains a vector that is not equal to $(1,1)$, e.g. $F(\mathbf{B}) = \{(1,1),(0,0)\}$. The rest of the definition of F is arbitrary.

Without any of the first two formulas, we can construct an aggregation rule F such that either $F \nvDash \langle\!\langle \mathbf{B}', x_1 \wedge x_2 \rangle\!\rangle$ or $F \nvDash \langle\!\langle \mathbf{B}'', x_1 \wedge x_2 \rangle\!\rangle$. Such a rule would not satisfy the antecedent of the third axiom instance, so we would then be able to define $F(\mathbf{B})$ however we liked, so again we could define $F(\mathbf{B})$ such that there is an outcome $v \in F(\mathbf{B})$ such that $v \neq (1,1)$. We conclude that we cannot omit any formula from $\mathcal{A}^E$.

- ***Relevance:*** *Note here that the first formula is an instance of $\mathcal{A}^{Fai}$, the second is an instance of $\mathcal{A}^{Can}$ and the third is an instance of $\mathcal{A}^{Rei}$.*
- ***Adequacy:*** *This is immediately clear.*
- ***Nontriviality:*** *Because $\Gamma = \top$, we only need to show that $\mathbb{I}(\mathcal{A}^N) \neq \varnothing$. We show that the aggregation rule F for which $F(\mathbf{B}) = ext(m(\mathbf{B}))$ for each profile, is in $\mathbb{I}(\mathcal{A}^N)$. For the profiles in $Mod(\Gamma)^1$, it selects the single ballot that is submitted, so F satisfies $\mathcal{A}^{Fai}$. It is also clear that $\mathcal{A}^{Can}$ is satisfied, as F elects all outcomes in case of a tie (which can only occur in $Mod(\Gamma)^2$).*

  Finally, suppose that $\mathbf{B} = \mathbf{B}' + \mathbf{B}''$ and $F(\mathbf{B}') \cap F(\mathbf{B}'') \neq \varnothing$. For $\mathbf{B} \in Mod(\Gamma)^2$, we have that $\mathbf{B}', \mathbf{B}'' \in Mod(\Gamma)^1$. Then because of faithfulness, $F(\mathbf{B}') \cap F(\mathbf{B}'') \neq \varnothing$ if and only if $F(\mathbf{B}') = F(\mathbf{B}'')$ and $\mathbf{B}' = \mathbf{B}''$. In that case, $\mathbf{B}$ contains two of the same ballots, so $ext(m(\mathbf{B}))$ consists only of that unique ballot, so $F(\mathbf{B}) = F(\mathbf{B}') \cap F(\mathbf{B}'')$.

  For $\mathbf{B} \in Mod(\Gamma)^3$, either $\mathbf{B}'$ or $\mathbf{B}''$ is of size one, and because of faithfulness, either $F(\mathbf{B}')$ or $F(\mathbf{B}'')$ is a singleton. Suppose without loss of generality that $\mathbf{B}' \in Mod(\Gamma)^1$ so $F(\mathbf{B}')$ is a singleton. Now suppose that $F(\mathbf{B}') \cap F(\mathbf{B}'') \neq \varnothing$, then it holds that $F(\mathbf{B}') \cap F(\mathbf{B}'') = F(\mathbf{B}')$. We conclude that for $F(\mathbf{B}') = \{v\}$, it holds that $v \in ext(m(\mathbf{B}'))$ and $v \in ext(m(\mathbf{B}''))$. That means that for any literal $\ell \in L_v$ (see Definition 2.13) we have that $N(\mathbf{B}'', \ell) \geq 1$. We conclude that for any literal $\ell \in L_v$ we have that $N(\mathbf{B}, \ell) \geq 2$, as $\mathbf{B}' = \{v\}$ by faithfulness and $\mathbf{B} = \mathbf{B}' + \mathbf{B}''$, so $v \in ext(m(\mathbf{B}))$. Now note that because $\mathbf{B} \in Mod(\Gamma)^3$, there are no ties so $|ext(m(\mathbf{B}))| = 1$. We conclude that $F(\mathbf{B}) = \{v\}$, so $F(\mathbf{B}) = F(\mathbf{B}') \cap F(\mathbf{B}'')$ and we have proven that $F \vDash \mathcal{A}^{Rei}$.

In the two examples above, we have assumed that $\Gamma = \top$. To illustrate why we only consider consistent aggregation rules for the explanatoriness condition, we provide the reader with the following example.

Example 4.3. *Let $m = 3$, $\Gamma = x_1$ and $\mathcal{A}^E = \{[\![\mathbf{B}, x_2]\!], [\![\mathbf{B}, (x_1 \wedge x_2) \to x_3]\!]\}$; the definition of $\mathcal{A}^N$ is not relevant for the purpose of this example, so we disregard it for now. Now consider the outcome statement $[\![\mathbf{B}, x_3]\!]$. Let F be a consistent aggregation rule such that $F \vDash \mathcal{A}^E$ and let $v \in F(\mathbf{B})$. Then $v \vDash x_1$ as F is consistent and $v \vDash x_2$ as $F \vDash \mathcal{A}^E$. Then because of the second formula in $\mathcal{A}^E$, we may conclude that $v \vDash x_3$, which gives us that $F \vDash [\![\mathbf{B}, x_3]\!]$, so $\mathcal{A}^E \vDash_\Gamma [\![\mathbf{B}, x_3]\!]$. It is easy to see that for any proper subset $\mathcal{A}' \subsetneq \mathcal{A}^E$, it holds that $\mathcal{A}' \nvDash [\![\mathbf{B}, x_3]\!]$, so $\mathcal{A}^E$ could serve as the explanatory basis of an unstructured justification for the outcome statement $[\![\mathbf{B}, x_3]\!]$, given that there exists a suiting*

normative basis $\mathcal{A}^N$. However, note that $\mathcal{A}^E \not\models [\![\mathbf{B}, x_3]\!]$, e.g. take an (inconsistent) aggregation rule F such that $F(\mathbf{B}) = \{(0,1,0)\}$.

For a corpus of axioms $\mathbb{A}$, we might be able to justify different (possibly contradicting) outcome statements for the same profile. This is due to the fact that different normative bases may force different outcome statements to be true, because the axioms in both bases could be different (although both are subsets of the corpus). However, if we fix a (nontrivial) normative basis, we should not be able to justify contradicting outcomes, as the justifications would then lose their persuasiveness. In the following lemma, we prove that this is indeed impossible. It is inspired by Theorem 1 in the paper by Boixel and Endriss [BE20]. We formulate an alternative lemma for our framework.

Lemma 4.1. *Let $\mathcal{A}^N$ be a set of axioms, and let $\mathcal{A}_1^E \lhd \mathcal{A}^N$ and $\mathcal{A}_2^E \lhd \mathcal{A}^N$ be two sets of axiom instances. Furthermore, let $\mathbf{B} \in Mod(\Gamma)^+$ and let $\varphi, \psi \in \mathcal{L}(X)$ such that $\varphi \wedge \psi \wedge \Gamma$ is unsatisfiable. Then we cannot have that $\langle \mathcal{A}^N, \mathcal{A}_1^E \rangle$ is an unstructured justification for $[\![\mathbf{B}, \varphi]\!]$ while $\langle \mathcal{A}^N, \mathcal{A}_2^E \rangle$ is an unstructured justification for $[\![\mathbf{B}, \psi]\!]$.*

Proof. We follow the proof idea from the proof of Theorem 1 by Boixel and Endriss [BE20].

Suppose for contradiction that we have a set of axioms $\mathcal{A}^N$, a profile $\mathbf{B} \in \mathrm{Mod}(\Gamma)^+$, formulas $\varphi, \psi \in \mathcal{L}(X)$, a set of axiom instances $\mathcal{A}_1^E \lhd \mathcal{A}^N$ such that $\langle \mathcal{A}^N, \mathcal{A}_1^E \rangle$ is a justification for $[\![\mathbf{B}, \varphi]\!]$ and a set of axiom instances $\mathcal{A}_2^E \lhd \mathcal{A}^N$ such that $\langle \mathcal{A}^N, \mathcal{A}_2^E \rangle$ is a justification for $[\![\mathbf{B}, \psi]\!]$ while $\varphi \wedge \psi \wedge \Gamma$ is unsatisfiable, for some integrity constraint Γ. As we have that both $\mathcal{A}_1^E \lhd \mathcal{A}^N$ and $\mathcal{A}_2^E \lhd \mathcal{A}^N$, we know that $\mathbb{I}_\Gamma(\mathcal{A}^N) \subseteq \mathbb{I}_\Gamma(\mathcal{A}_1^E)$ and $\mathbb{I}_\Gamma(\mathcal{A}^N) \subseteq \mathbb{I}_\Gamma(\mathcal{A}_2^E)$; this follows from the fact that every axiom instance $A \in \mathcal{A}_1^E$ is an instance of some axiom in $\mathcal{A}^N$ (and the same holds for $\mathcal{A}_2^E$), so any consistent aggregation rule that satisfies the axioms in $\mathcal{A}^N$ also satisfies the axiom instances in both $\mathcal{A}_1^E$ and $\mathcal{A}_2^E$. We then find that $\mathbb{I}_\Gamma(A^N) \subseteq \mathbb{I}_\Gamma(\mathcal{A}_1^E) \cap \mathbb{I}_\Gamma(\mathcal{A}_2^E)$. Since $\mathcal{A}_1^E$ explains $[\![\mathbf{B}, \varphi]\!]$, we find that $\mathbb{I}_\Gamma(\mathcal{A}_1^E) \subseteq \mathbb{I}_\Gamma([\![\mathbf{B}, \varphi]\!])$ (see the explanatoriness condition). Similarly, we find that $\mathbb{I}_\Gamma(\mathcal{A}_2^E) \subseteq \mathbb{I}_\Gamma([\![\mathbf{B}, \psi]\!])$. When we tie all of this together, we get that

$$\begin{aligned}
\mathbb{I}_\Gamma(\mathcal{A}^N) &\subseteq \mathbb{I}_\Gamma(\mathcal{A}_1^E) \cap \mathbb{I}_\Gamma(\mathcal{A}_2^E) \\
&\subseteq \mathbb{I}_\Gamma([\![\mathbf{B}, \varphi]\!]) \cap \mathbb{I}_\Gamma([\![\mathbf{B}, \psi]\!]) \\
&= \mathbb{I}_\Gamma(\{[\![\mathbf{B}, \varphi]\!], [\![\mathbf{B}, \psi]\!]\}) \\
&= \mathbb{I}_\Gamma([\![\mathbf{B}, \varphi \wedge \psi]\!]).
\end{aligned}$$

The first equality follows from the fact that the intersection of the two sets consists exactly of the consistent aggregation rules that satisfy both the outcome statements. The second equality follows from the fact that for every aggregation rule F, the following holds:

$$F \vDash [\![\mathbf{B}, \varphi]\!] \text{ and } F \vDash [\![\mathbf{B}, \psi]\!] \text{ if and only if } F \vDash [\![\mathbf{B}, \varphi \wedge \psi]\!].$$

From this we conclude that for any consistent aggregation rule that satisfies the axioms in $\mathcal{A}^N$, we have that $F \vDash [\![\mathbf{B}, \varphi \wedge \psi]\!]$. Note that by our assumption, $\varphi \wedge \psi \wedge \Gamma$ is unsatisfiable, so there cannot be an outcome vector in $F(\mathbf{B})$ that makes $\varphi \wedge \psi \wedge \Gamma$ true. As a consequence, there is no consistent aggregation rule F for which $F \vDash [\![\mathbf{B}, \varphi \wedge \psi]\!]$. We conclude that $\mathbb{I}_\Gamma([\![\mathbf{B}, \varphi \wedge \psi]\!]) = \varnothing$, and because $\mathbb{I}_\Gamma(\mathcal{A}^N) \subseteq \mathbb{I}_\Gamma([\![\mathbf{B}, \varphi \wedge \psi]\!])$, it should also hold that $\mathbb{I}_\Gamma(\mathcal{A}^N) = \varnothing$. This contradicts the nontriviality condition from Definition 4.2, which states that $\mathbb{I}_\Gamma(\mathcal{A}^N) \neq \varnothing$, which concludes our proof. □

4.2 Structured Justifications

In Section 4.1, we defined the notion of an unstructured justification: a pair consisting of a set of axioms called the normative basis and a set of axiom instances of those axioms called the explanatory basis that justify a universal outcome statement of the form $[\![\mathbf{B}, \varphi]\!]$. Intuitively, the axiom instances in the explanatory basis force $[\![\mathbf{B}, \varphi]\!]$ to hold, i.e. given the input $\mathbf{B}$ any outcome of any consistent aggregation rule that satisfies $\mathcal{A}^E$ should satisfy the formula φ.

In Chapter 3, we defined a tableau-based calculus for binary aggregation. This method is used to prove that there is no consistent aggregation rule that satisfies a set of outcome statements in combination with a set of axiom instances. An important feature of such a tableau is that it offers a structured way to display a proof. This section is about structuring a justification through the use of a tableau. Essentially, a structured justification is an unstructured justification together with a tableau that shows how to structure the justification.

Suppose we already have an unstructured justification $\langle \mathcal{A}^N, \mathcal{A}^E \rangle$ for the outcome statement $[\![\mathbf{B}, \varphi]\!]$. We then construct a tableau as follows. First, we define a root node $\mathcal{S} = \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}$ that states the negation of this outcome statement. We then expand the tableau by applying the different expansion rules. The final goal is to construct a closed tableau that is rooted in $\mathcal{S} = \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}$ and licensed by $\mathcal{A}^E$, which proves that there is **no** consistent aggregation rule that satisfies both $\mathcal{S}$ and the axiom instances in $\mathcal{A}^E$. Because we still have the nontriviality condition, we may then conclude that any consistent aggregation rule that satisfies the axiom instances should also satisfy $[\![\mathbf{B}, \varphi]\!]$. Note here that consistency is also checked within the tableau through the application of the constraint-driven expansion rule. Finally, we may then accept this as a structured justification for the outcome statement $[\![\mathbf{B}, \varphi]\!]$. We construct the tableaux following this contradiction method because it interacts nicely with the case distinctions that are implemented through the use of the branching rule; this allows us to find a contradiction for every possible case in the case distinction, i.e. an inconsistent outcome statement in every leaf node in the tableau.

One important difference with unstructured justifications is that we no longer require the explanatory basis to be minimal; this is due to the fact that the shortest explanatory basis does not necessarily provide the "best" justification, as observed by Boixel et al. [BEH22]. This is why we cannot simply extend the definition of an unstructured justification for the definition of a structured justification.

The following definition is based on the work of Boixel et al. [BEH22], who have defined structured justifications for voting theory.

Definition 4.3 (Structured Justification)**.** *Let $\mathbb{A}$ be a corpus of axioms and let $[\![\mathbf{B}, \varphi]\!]$ be a universal outcome statement with $\mathbf{B} \in Mod(\Gamma)^+$ and $\varphi \in \mathcal{L}(X)$. A **structured justification** for the outcome statement $[\![\mathbf{B}, \varphi]\!]$ is a triple $\langle \mathcal{A}^N, \mathcal{A}^E, \mathcal{T} \rangle$ consisting of a normative basis of axioms, an explanatory basis of axiom instances and a tableau such that the following conditions hold:*

- ***Explanatoriness:*** *$\mathcal{T}$ is a closed tableau that is rooted in the node $\mathcal{S} = \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}$ and licensed by the set $\mathcal{A}^E$.*
- ***Relevance:*** *Every instance in $\mathcal{A}^E$ is an instance of an axiom in $\mathcal{A}^N$: $\mathcal{A}^E \lhd \mathcal{A}^N$.*
- ***Adequacy:*** *All axioms in $\mathcal{A}^N$ are in the corpus: $\mathcal{A}^N \subseteq \mathbb{A}$.*
- ***Nontriviality:*** *There exists a consistent aggregation rule F such that F satisfies the axioms in $\mathcal{A}^N$: $\mathbb{I}_\Gamma(\mathcal{A}^N) \neq \varnothing$.*

Before we provide some illustrative examples, let us first dissect the definition above. As for the unstructured justifications, we want to justify a universal outcome statement $[\![\mathbf{B}, \varphi]\!]$, which serves as an argument for collectively accepting the outcome property φ (or: accepting only outcome vectors that satisfy this formula), given that the profile $\mathbf{B}$ displays the opinions of that group. Now, instead of only offering a set of (collectively accepted) axioms and a set of instances of those axioms, we also provide a tableau that shows how to properly structure the set of instances into an argument that explains why the outcome statement $[\![\mathbf{B}, \varphi]\!]$ is forced by this set.

In order for the tableau to fulfill this role, we require it to be closed, rooted in the singleton containing the negation of the statement we want to justify and licensed by the set of axiom instances $\mathcal{A}^E$; note here that the tableau shows a proof by contradiction. This is captured in the explanatoriness condition.

The other three conditions are identical to the conditions in the definition of an unstructured justification (see Definition 4.1). As noted before, an important difference with unstructured justifications is that we do not require the minimality of the justification. In the following remark, we show how the explanatoriness condition above relates to that of Definition 4.2.

Remark 4.2. *Let $[\![\mathbf{B}, \varphi]\!]$ be some outcome statement and let $\langle \mathcal{A}^N, \mathcal{A}^E, \mathcal{T} \rangle$ be a structured justification for $[\![\mathbf{B}, \varphi]\!]$. Then we know that $\mathcal{T}$ is a closed tableau that is rooted in the node $\mathcal{S} = \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}$ and licensed by the set $\mathcal{A}^E$. But now the soundness of our calculus (see Theorem 3.3) gives us that there exists no consistent aggregation rule that satisfies the outcome statement $\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle$ and the axiom instances in $\mathcal{A}^E$. Note that nontriviality forces the existence of a consistent aggregation rule that satisfies the axioms in $\mathcal{A}^N$, so this rule also satisfies the axiom instances in $\mathcal{A}^E$. Then for any such rule $F \in \mathbb{I}_\Gamma(\mathcal{A}^E)$, it holds that $F \nvDash \langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle$, so $F \vDash \neg\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle$, which is equivalent to saying that $F \vDash [\![\mathbf{B}, \varphi]\!]$. We conclude that $\mathbb{I}_\Gamma(\mathcal{A}^E) \subseteq \mathbb{I}_\Gamma([\![\mathbf{B}, \varphi]\!])$.*

Let us now consider some examples to show the inner workings of the definition. We depict the tableaux following Notation 3.1.

Example 4.4. *We reconsider Example 4.1, which served as an example to illustrate the definition of unstructured justifications. We had the following situation: $n = 4$, $m = 2$, $\Gamma = \top$, $\mathbf{B} = \{(1,0), (1,0), (0,1), (1,1)\}$, $\mathbb{A} = \{\mathcal{A}^{Fai}, \mathcal{A}^{Hom}, \mathcal{A}^{Weak\text{-}Un}, \mathcal{A}^{Strong\text{-}Un}, \mathcal{A}^{Rei}, \mathcal{A}^{Mon}, \mathcal{A}^{Maj}, \mathcal{A}^{Can}\}$ and we wanted to justify the outcome statement $[\![\mathbf{B}, x_1]\!]$.*

The unstructured justification we gave in the example was the following:

$$\langle \mathcal{A}^N, \mathcal{A}^E \rangle = \langle \{\mathcal{A}^{Maj}\}, \{[\![\mathbf{B}, (x_1 \wedge x_2) \vee (x_1 \wedge \neg x_2)]\!]\} \rangle.$$

Where we noted that $ext(m(\mathbf{B})) = \{(1,1), (1,0)\}$, which is the set that the formula in the outcome statement in the axiom instance describes. Now we provide a structured justification $\langle \mathcal{A}^N, \mathcal{A}^E, \mathcal{T} \rangle$, where the first two sets are equal to those of the unstructured justification. As the final three conditions of both justification definitions are equal, we refer to Example 4.1 as we already showed there that the tuple $\langle \mathcal{A}^N, \mathcal{A}^E \rangle$ satisfies these conditions. It only remains to provide a closed tableau $\mathcal{T}$ that is rooted in $\{\langle\!\langle \mathbf{B}, \neg x_1 \rangle\!\rangle\}$ and licensed by $\mathcal{A}^E = \{[\![\mathbf{B}, (x_1 \wedge x_2) \vee (x_1 \wedge \neg x_2)]\!]\}$. Now we define the tableau $\mathcal{T}$ in Figure 4.1, for which it is immediately clear that these three properties hold.

We also show a way to structure the unstructured justification in Example 4.2. This example translates the example presented in the paper of Boixel et al. [BEH22] to the framework of binary aggregation and the defined tableau-based calculus.

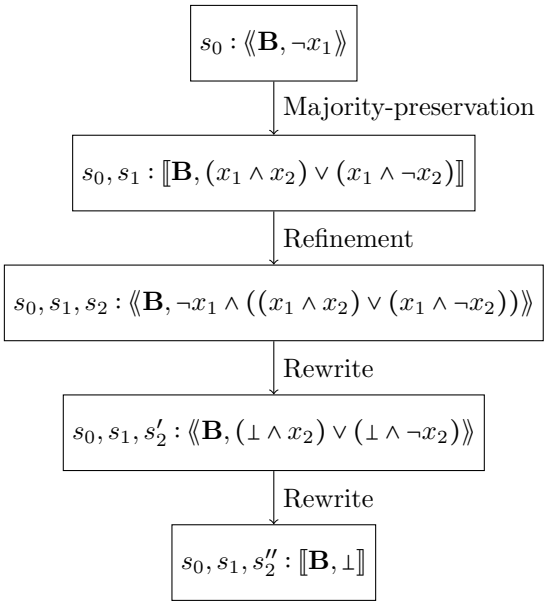

Figure 4.1: The tableau $\mathcal{T}$ in Example 4.4

Example 4.5. *Recall that in Example 4.2, we had a normative basis* $\mathcal{A}^N = \{\mathcal{A}^{Fai}, \mathcal{A}^{Can}, \mathcal{A}^{Rei}\}$ *and an explanatory basis*

$$\begin{aligned}\mathcal{A}^E = \{&[\![\mathbf{B}', x_1 \wedge x_2]\!]\} \\ &\cup \{\langle\!\langle \mathbf{B}'', x_1 \wedge x_2 \rangle\!\rangle\} \\ &\cup \{\big(\langle\!\langle \mathbf{B}', x_1 \wedge x_2 \rangle\!\rangle \wedge \langle\!\langle \mathbf{B}'', x_1 \wedge x_2 \rangle\!\rangle\big) \\ &\to \big(((([\![\mathbf{B}', x_1 \wedge x_2]\!] \vee [\![\mathbf{B}'', x_1 \wedge x_2]\!]) \to [\![\mathbf{B}, x_1 \wedge x_2]\!]) \\ &\wedge ((\langle\!\langle \mathbf{B}', \varphi_w \rangle\!\rangle \wedge \langle\!\langle \mathbf{B}'', \varphi_w \rangle\!\rangle) \to \langle\!\langle \mathbf{B}, \varphi_w \rangle\!\rangle)\big)\},\end{aligned}$$

where $\mathbf{B} = \{(1,0),(0,1),(1,1)\}$, $\mathbf{B}' = \{(1,1)\}$ *and* $\mathbf{B}'' = \{(1,0),(0,1)\}$. *Note here that the first formula in* $\mathcal{A}^E$ *is an instance of faithfulness, the second of cancellation and the third of reinforcement. We want to provide a structured justification for the outcome statement* $[\![\mathbf{B}, x_1 \wedge x_2]\!]$. *Again, we only provide a tableau* $\mathcal{T}$, *as we already showed in Example 4.2 that the last three properties are satisfied by* $\mathcal{A}^N$ *and* $\mathcal{A}^E$. *In the tableau in Figure 4.2, note that in the reinforcement step, we make use of the fact that both the precondition of the third formula in* $\mathcal{A}^E$, $\langle\!\langle \mathbf{B}', x_1 \wedge x_2 \rangle\!\rangle \wedge \langle\!\langle \mathbf{B}'', x_1 \wedge x_2 \rangle\!\rangle$, *and the formula* $[\![\mathbf{B}', x_1 \wedge x_2]\!] \vee [\![\mathbf{B}'', x_1 \wedge x_2]\!]$ *are satisfied, so we may conclude* $[\![\mathbf{B}, x_1 \wedge x_2]\!]$ *and add this statement through an application of the axiom-driven expansion rule; this inference is explained more elaborately in Example 4.2.*

Note here that the application of the branching rule does not add any semantic value to the tableau, but it might still contribute to the explainability.

Finally, we also show how to turn the unstructured justification in Example 4.3 into a structured justification by constructing a tableau.

Example 4.6. *Recall that in Example 4.3, we had the following scenario:* $m = 3$, $\Gamma = x_1$, $\mathcal{A}^E = \{[\![\mathbf{B}, x_2]\!], [\![\mathbf{B}, (x_1 \wedge x_2) \to x_3]\!]\}$ *and* $\mathcal{A}^N$ *was irrelevant; just suppose it is some axiom set such that relevance, adequacy and nontriviality hold. We want to justify the outcome statement* $[\![\mathbf{B}, x_3]\!]$. *In Figure 4.3, we construct a tableau* $\mathcal{T}$ *that is closed, rooted in* $\mathcal{S} = \{\langle\!\langle \mathbf{B}, \neg x_3 \rangle\!\rangle\}$ *and licensed by* $\mathcal{A}^E$.

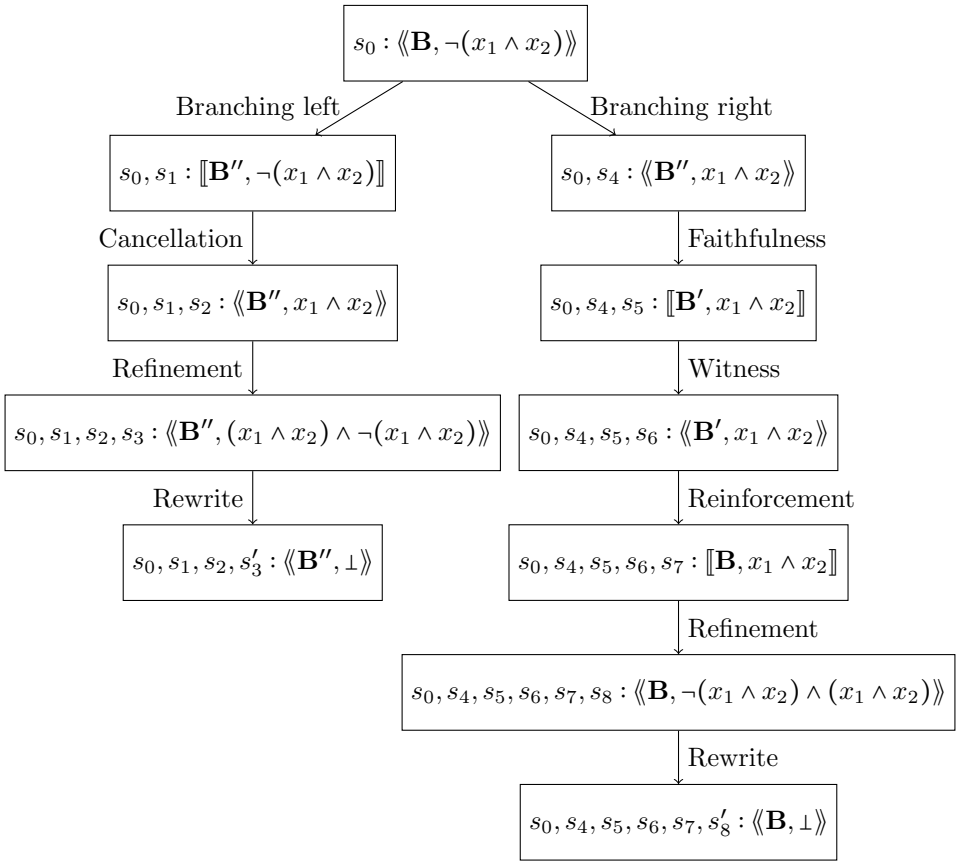

Figure 4.2: The tableau $\mathcal{T}$ in Example 4.5

In the examples above, we have obtained structured justifications based on unstructured justifications. However, given a (satisfiable) normative basis, we could also directly try to construct a closed tableau that is rooted in the singleton containing the negation of the outcome statement that we want to justify. During this construction, one could use any axiom instance from an axiom in the normative basis, and when the tableau is closed, the explanatory basis simply becomes the set of all axioms used in the applications of the axiom-driven expansion rules.

We end the section by showing that, as with unstructured justifications (see Lemma 4.1), we cannot justify contradictory statements using structured justifications.

Lemma 4.2. *Let $\mathcal{A}^N$ be a set of axioms, and let $\mathcal{A}_1^E \lhd \mathcal{A}^N$ and $\mathcal{A}_2^E \lhd \mathcal{A}^N$ be two explanatory bases. Furthermore, let $\mathbf{B} \in Mod(\Gamma)^+$ and let $\varphi, \psi \in \mathcal{L}(X)$ such that $\varphi \wedge \psi \wedge \Gamma$ is unsatisfiable. Then we cannot have that $\langle \mathcal{A}^N, \mathcal{A}_1^E, \mathcal{T}_1 \rangle$ is a structured justification for $[\![\mathbf{B}, \varphi]\!]$ while $\langle \mathcal{A}^N, \mathcal{A}_2^E, \mathcal{T}_2 \rangle$ is a structured justification for $[\![\mathbf{B}, \psi]\!]$.*

Proof. From Remark 4.2, it follows that $\mathbb{I}_\Gamma(\mathcal{A}_1^E) \subseteq \mathbb{I}_\Gamma([\![\mathbf{B}, \varphi]\!])$ and $\mathbb{I}_\Gamma(\mathcal{A}_2^E) \subseteq \mathbb{I}_\Gamma([\![\mathbf{B}, \psi]\!])$. The rest of the proof is analogous to the proof of Lemma 4.1. □

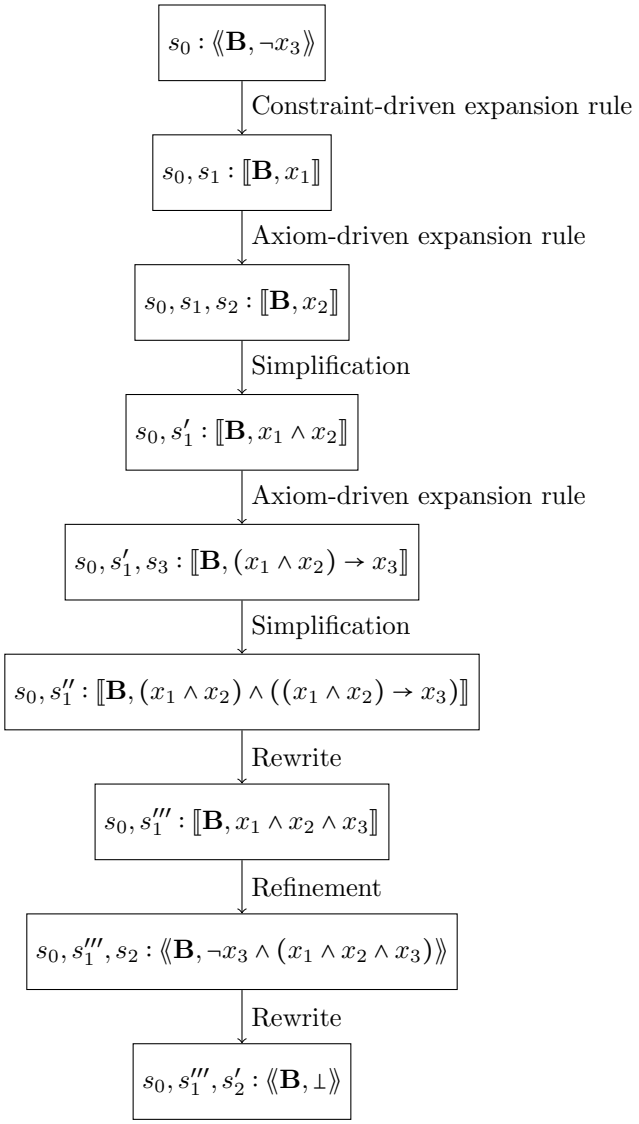

Figure 4.3: The tableau $\mathcal{T}$ in Example 4.6

4.3 Discussion

In Section 4.1, we defined unstructured justifications. In essence, an unstructured justification for an outcome statement $[\![\mathbf{B}, \varphi]\!]$ consists of a subset of the collectively accepted axioms (the normative basis) together with a set of specific instances of those axioms (the explanatory basis) that contain an explanation as to why it should hold that for any consistent aggregation rule F that satisfies the axioms in the normative basis, it should hold that any outcome vector in $F(\mathbf{B})$ satisfies the formula φ. Consequently, we showed several examples to illustrate the definition. Finally, we proved in Lemma 4.1 that we cannot justify contradicting statements.

In Section 4.2, we defined structured justifications. A structured justification extends the definition of an unstructured justification by adding a tableau which structures the explanatory basis. We also illustrated this definition through the use of examples and showed that we cannot justify contradicting statements with structured justifications; see Lemma 4.2.

The first observation we make is that the explanatoriness condition for unstructured justifications considers only consistent aggregation rules. This is due to the fact that these are the only "allowed" rules, so we need not consider all aggregation rules for the entailment. This also parallels the explanatoriness condition for the structured justification, where the integrity constraint is taken into account by the constraint-driven expansion rule. This is also shown in detail in Remark 4.2.

Boixel and Endriss [BE20] define unstructured justifications for voting theory. Boixel et al. [BEH22] have extended this definition and defined structured justifications for voting theory. The general idea of both definitions coincides with that of our definitions of unstructured and structured justifications, although they consider different frameworks and our tableau-based calculus differs substantially from theirs; see Section 3.4.

Although the nontriviality condition we defined is a direct translation of the definition by Boixel et al. [BEH22], it might be somewhat harder to check within our framework. This is due to the fact that the axiomatic method has not been researched as extensively for binary aggregation as it has been for voting theory. There has been a lot of research on the different voting rules and the axioms they satisfy, e.g. by Arrow [Arr63] and Brandt et al. [Bra+16]. For voting theory, it is relatively easy to check for a set of axioms $\mathcal{A}^N$ whether it is satisfiable or not; maybe it might be already proven somewhere that there is a specific voting rule that satisfies all the axioms in $\mathcal{A}^N$. For binary aggregation, the axiomatic method has been researched a bit less thoroughly, which is why it might sometimes be harder to check the satisfiability of a set of axioms $\mathcal{A}^N$. However, in Chapter 6 we will show how to check the nontriviality condition through the use of a SAT solver.

In Section 1.4, we have introduced a set of binary aggregation axioms. These are adaptions of the judgment aggregation axioms in the works of Grossi and Pigozzi [GP22] and Lang et al. [Lan+17]. The set of axioms at hand is somewhat restrictive; some axioms are too strong, like majority-preservation, and other axioms are too weak, like monotonicity. Because of the generality of the model we have defined, one could easily introduce new axioms to make for more interesting and intricate justifications. For now, our set of axioms will do.

Part II

Implementation

Chapter 5

Encoding Binary Aggregation

In the second part of this thesis, we will construct an algorithm that allows us to generate sets of axiom instances that resemble unstructured justifications and trees that offer an alternative to the tableaux defined in Chapter 3, when given a normative basis and an outcome statement that should be justified. In order to do so, we will first encode the framework of binary aggregation in Python. Consequently, we will encode the axioms presented in Section 2.3. With this encoding at hand, we are ready to construct the desired algorithm, which we will do in Chapter 6.

Geist and Peters [GP17] show a way of encoding the framework of voting theory, and specifically axioms, in a computer. They do so by encoding an axiom as a CNF-formula: a conjunction over a set of clauses (see Section 1.1). This approach aligns with the way we have expressed our axioms in the language of $\mathcal{L}(O)$ in Section 2.3, where we rewrote an axiom as a set of axiom instances. It remains to rewrite these axiom instances as clauses, which we will do in Section 5.3.

In a Jupyter Notebook, Endriss [End23] offers a way of implementing a similar method to that of Geist and Peters in Python. For any input profile $\mathbf{R}$ and every possible outcome x, he introduces one propositional variable $p_{\mathbf{R},x}$ that is satisfied by a voting rule if and only if x is in the output of that rule under input profile $\mathbf{R}$. We will also use Python for our encoding, and we adapt the implementation for voting theory by Endriss to binary aggregation, following the general structure and making use of functions provided in the existing implementation if possible.

In the following chapter, we will adapt the encoding of the framework offered by Geist and Peters [GP17] and implemented in Python by Endriss [End23] to binary aggregation. In Section 5.1, we will encode the general framework of binary aggregation. We show how to encode the basic concepts of the framework and how to encode outcome statements. We also explain some of our considerations regarding the encoding. Consequently, in Section 5.2, we show how to encode aggregation rules. This is done by encoding extra dependencies between the outcome statements to make sure that our encoding behaves according to the semantics introduced in Chapter 2. These dependencies will also be denoted as axioms. We conclude the chapter with Section 5.3, where we show how to encode these axioms and the axioms introduced in Section 2.3.

The encoding that is described in this chapter and in Chapter 6 can be found in the Jupyter Notebook on `https://zenodo.org/records/16614542` [Jon25].

5.1 Encoding the Framework

The first things we need to encode are the basic definitions from binary aggregation.

Encoding Issues, Voters and Profiles

Following the work of Endriss [End23], we will encode voters, issues and profiles as integers. In principle, we want to start counting at zero for each of these (as binary counting also starts at zero), but as the integers for the issues should represent the variables $\{x_1, \ldots, x_m\}$, we use the integers $\{1, \ldots, m\}$ to encode them.

Convention 2. *The m **issues** are represented by the integers $\{1, \ldots, m\}$. The n **voters** are represented by the integers $\{0, \ldots, n-1\}$.*

for $\Gamma = \top$, the number of possible profiles follows from the number of issues and voters. We will follow the work of Endriss [End23], but extend the encoding to allow for profiles of different sizes.

Without taking the integrity constraint into consideration, there are 2^m possible ballots: a binary choice for each of the m issues. In a profile, there are up to n voters that have a choice between 2^m ballots. To allow for a structured way of counting the different profiles, we decided to take the order of the profiles into consideration, thus encoding them as ordered lists (or: sequences) instead of multisets. This gives rise to the following definition.

Definition 5.1. *The **ordered domain** is the set of (ordered) sequences of ballots of size one up to n:*

$$Mod(\Gamma)^{+*} := \{(B_1, \ldots, B_k) \mid B_1, \ldots, B_k \in Mod(\Gamma) \text{ and } 1 \leq k \leq n\}.$$

*Elements of $Mod(\Gamma)^{+*}$ we call **ordered profiles**. For a consistent aggregation rule $F : Mod(\Gamma)^{+} \to 2^{Mod(\Gamma)} \setminus \{\varnothing\}$, an ordered profile $\mathbf{B} \in Mod(\Gamma)^{+*}$ and a formula $\varphi \in \mathcal{L}(X)$, we say that*

$$F \vDash \langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle \iff F \vDash \langle\!\langle \mathbf{B}', \varphi \rangle\!\rangle,$$

*where $\mathbf{B}'$ is the multiset that contains exactly the elements that are in the ordered profile $\mathbf{B}$, so for $\mathbf{B} = (B_1, \ldots, B_k)$, we find that $\mathbf{B}' = \{B_1, \ldots, B_k\}$. We may also call $\mathbf{B}'$ the multiset **corresponding to** $\mathbf{B}$.*

We then find the following equality (for $\Gamma = \top$):

$$|\mathrm{Mod}(\Gamma)^{+*}| = |\bigcup_{1 \leq k \leq n} \{(B_1, \ldots, B_k) \mid B_1, \ldots, B_k \in \mathrm{Mod}(\Gamma)\}| = \sum_{1 \leq k \leq n} (2^m)^k.$$

This way of counting allows for easy retrieval of the ballots when presented with an integer representing a profile. The anonymity axiom (which will be introduced in Section 5.2) will make sure that elements of $\mathrm{Mod}(\Gamma)^{+*}$ that are equal as multisets, i.e. they have the same corresponding multiset, will be semantically unified in the framework.

Convention 3. *We represent the **profiles** by the integers $\{0, \ldots, \sum_{1 \leq k \leq n} (2^m)^k - 1\}$, where we distinguish between the profiles $\{B, B'\}$ and $\{B', B\}$ in order to allow for a nice counting of the profiles. The profiles are counted in base 2^m (the number of different ballots).*

The correspondence between the set of integers above and the different profiles becomes clear in the following example.

Example 5.1. *Suppose that $n = 2$ and $m = 3$. Then the ordered profile that is represented by the integer 0 is $((0,0,0))$. The (ordered) profile represented by the integer 7 is the profile $((1,1,1))$ and 8 then represents the profile $((0,0,0),(0,0,0))$. The next profile (represented by 9)*

is $((1,0,0),(0,0,0))$*. The profile* $((1,1,1),(0,0,0))$ *is represented by 15 and* $((0,0,0),(1,0,0))$ *is represented by 16.*

Note here that we assume that $\Gamma = \top$, while it might be the case that Γ restricts certain profiles. This will be encoded in Section 5.2, where we consider both the input and output restriction of the integrity constraint.

Encoding Outcome Statements

Consequently, we want to encode the different outcome statements. Theoretically, we have an existential outcome statement $\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle$ and a universal outcome statement $[\![\mathbf{B}, \varphi]\!]$ for every ordered profile $\mathbf{B} \in \mathrm{Mod}(\Gamma)^{+*}$ and formula $\varphi \in \mathcal{L}(X)$. However, there are infinitely many formulas $\varphi \in \mathcal{L}(X)$. We do find the following lemma.

Lemma 5.1. *There are* 2^{2^m} *formulas in* $\mathcal{L}(X)$ *up to equivalence.*

Proof. This follows from the fact that any two formulas that are made true by the same valuations on X are equivalent by definition, so we can represent a formula $\varphi \in \mathcal{L}(X)$ by the set of (complete) valuations on X that make φ true. There are 2^m valuations on X: any variable $x_i \in X$ (for $i \in \{1, \ldots, m\}$) is either true or false. Then there are 2^{2^m} subsets of valuations, each corresponding to an equivalence class of formulas in $\mathcal{L}(X)$. This concludes our proof. □

As this number of formulas (up to equivalence) is double exponential, it is undesirable to implement, as the size of the set of outcome statements that we need to encode will blow up very quickly. For this reason, we only encode a fragment of $\mathcal{L}(X)$.

When deciding on a fragment of $\mathcal{L}(X)$ for our encoding, we want to take the following into consideration. We have that $\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle \equiv \neg[\![\mathbf{B}, \neg\varphi]\!]$ and $[\![\mathbf{B}, \varphi]\!] \equiv \neg\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle$ for any formula $\varphi \in \mathcal{L}(X)$ and profile $\mathbf{B} \in \mathrm{Mod}(\Gamma)^{+}$, which means that our encoding of the fragment of $\mathcal{L}(X)$ should be closed under negation, where we note that some of the axioms and the expansion rules show that there is a narrow interaction between the universal and existential outcome statements.

The considerations above have led to the adoption of the following fragment of $\mathcal{L}(X)$, allowing for both expressivity within the languages and easy negation in $\mathcal{L}(O)$ without having a double exponential number of formulas.

Definition 5.2. *We define the following subset of* $\mathcal{L}(X)$*:*

$$\mathcal{L}(X)^{\wedge,\vee} = \{\varphi \in \mathcal{L}(X) \mid \varphi = \bigwedge_{\ell \in L} \ell \text{ or } \varphi = \bigvee_{\ell \in L} \ell \text{ for } L \subset X^* \text{ such that } \varphi \notin \bot \text{ and } \varphi \notin \top\}.$$

In other words, $\mathcal{L}(X)^{\wedge,\vee}$ *is the fragment of* $\mathcal{L}(X)$ *that contains all formulas that are either a conjunction or a disjunction of some set of literals in* X^**, but they cannot contain both* x *and* $\neg x$ *for any variable* $x \in X$*.*

This fragment allows for a straightforward ternary encoding: for every issue, either it does not appear in the formula, it appears positively or it appears negatively. This gives rise to 3^m possible ternary vectors. Then, one more binary is needed in order to encode whether the formula is a conjunction or a disjunction, leaving us with $2 \cdot (3^m)$ different formulas. For readability, we may sometimes write these formulas as $\mathrm{dis}(v)$ and $\mathrm{con}(v)$, where v is a ternary string, deciding for each issue whether it appears positively, negatively or not at all. This encoding is illustrated in the following example.

Example 5.2. *For $m = 2$ and conjunctive formulas, we find the following correspondence.*

Integer	Ternary string	Formula in $\mathcal{L}(X)^{\wedge,\vee}$
0	00	con(00)
1	01	x_1
2	02	$\neg x_1$
3	10	x_2
4	11	$x_1 \wedge x_2$
5	12	$\neg x_1 \wedge x_2$
6	20	$\neg x_2$
7	21	$x_1 \wedge \neg x_2$
8	22	$\neg x_1 \wedge \neg x_2$

Table 5.1: Encoding of some formulas in $\mathcal{L}(X)^{\wedge,\vee}$

The formula con(00) *is not defined in our framework, but is only there so that we can structurally count in ternary and later easily retrieve the formulas represented in the rightmost column.*

Now that we have obtained the encoding of voters, issues, profiles and formulas in $\mathcal{L}(X)^{\wedge,\vee}$, we are ready to encode the literals corresponding to the outcome statements in $\mathbb{S}$. Again, we simply use integers for this encoding.

In the Jupyter Notebook developed by Endriss [End23], the fundamental literals of the encoding for voting theory are existential statements describing the outcome for a certain input profile. The literals that we will encode are the outcome statements described in Section 2.1, which function as a more indirect alternative to the statements introduced in the Jupyter Notebook of Endriss.

For every profile $\mathbf{B} \in \mathrm{Mod}(\Gamma)^{+*}$ and for every formula $\varphi \in \mathcal{L}(X)^{\wedge,\vee}$, there is an existential outcome statement $\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle$. Instead of encoding the universal outcome statements, we encode the negations of the existential outcome statements, where we note that any universal outcome statement is equivalent to the negation of some existential outcome statement. For example, the statement $[\![\mathbf{B}, x_1 \wedge \neg x_2]\!]$ is equivalent to $\neg\langle\!\langle \mathbf{B}, \neg x_1 \vee x_2 \rangle\!\rangle$ and the statement $[\![\mathbf{B}, x_1 \vee x_2 \vee x_3]\!]$ is equivalent to $\neg\langle\!\langle \mathbf{B}, \neg x_1 \wedge \neg x_2 \wedge \neg x_3 \rangle\!\rangle$.

Remark 5.1. *Note here that we consider outcome statements with ordered profiles instead of multisets. This causes no problems, as we will make sure the ordered profiles that are equal as multisets are treated as the same profile: there are simply multiple representatives for every multiset $\mathbf{B} \in Mod(\Gamma)^+$. For now, we will consider outcome statements with ordered profiles $\mathbf{B} \in Mod(\Gamma)^{+*}$ instead of multiset profiles $\mathbf{B} \in Mod(\Gamma)^+$.*

When encoding the literals, we first count the statements $\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle$ where φ is a conjunction. The first literal, represented by a 1 instead of a 0 (as we want to be able to negate it in the clauses later on), is the existential statement $\langle\!\langle ((0, \ldots, 0)), \mathrm{con}(0 \cdots 0) \}$. The statement corresponding to the integer 2 then is $\langle\!\langle ((0, \ldots, 0)), x_1 \rangle\!\rangle$; the counting follows that of Example 5.2. The integer $3^m + 1$ is the outcome statement $\langle\!\langle ((1, \ldots, 0)), \mathrm{con}(0 \cdots 0) \rangle\!\rangle$. After we've counted every outcome statement with a conjunctive formula, we turn to the disjunctive formulas.

This method gives rise to the following function, where b is the integer that represents the ordered profile, v is the integer that represents the vector describing the formula (see Example 5.2), **con_or_dis** is either a 0 (for conjunction) or a 1 (for disjunction) and profile_sum(n) is the number of ordered profiles of size up to n.

```
def posLiteral(b, v, con_or_dis):
    return b*(3**m) + v + 1 + con_or_dis*(3**m)*(profile_sum(n))
```

The function above allows us to encode all the relevant information of an outcome statement in one integer. The negations are encoded by simply taking the negative values. The output of the function above under an inputted outcome statement is the integer representing that outcome statement.

Remark 5.2. *Finally, observe that* $\bot \notin \mathcal{L}(X)^{\wedge,\vee}$*, so we will not be able to derive a contradiction through an outcome statement of the form* $\langle\!\langle \mathbf{B}, \bot \rangle\!\rangle$ *or* $[\![\mathbf{B}, \bot]\!]$*. Instead, we will derive a contradiction whenever we accept both a literal and its negation, e.g.* $\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle$ *and* $[\![\mathbf{B}, \neg\varphi]\!]$*. This will become clear in the algorithms described in the next chapter.*

5.2 Encoding Consistent Aggregation Rules

As discussed in the previous section, we have encoded the outcome statements $\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle$ and $\neg\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle$ for every $\mathbf{B} \in \mathrm{Mod}(\Gamma)^{+*}$ and $\varphi \in \mathcal{L}(X)^{\wedge,\vee}$. This gives rise to the following set of propositional variables:

$$\mathbb{S}^* := \{\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle \mid \mathbf{B} \in \mathrm{Mod}(\Gamma)^{+*} \text{ and } \varphi \in \mathcal{L}(X)^{\wedge,\vee}\} \cup \{[\![\mathbf{B}, \varphi]\!] \mid \mathbf{B} \in \mathrm{Mod}(\Gamma)^{+*} \text{ and } \varphi \in \mathcal{L}(X)^{\wedge,\vee}\}.$$

Note here that $\mathbb{S}^*$ is not really a fragment of the set of outcome statements $\mathbb{S}$ (see Definition 2.5) because the profiles in the definition above are ordered while the profiles that appear in Definition 2.5 are multisets. However, the encoding of the anonymity axiom will make sure that any two elements $\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle, \langle\!\langle \mathbf{B'}, \varphi \rangle\!\rangle \in \mathbb{S}^*$, where $\mathbf{B}$ and $\mathbf{B'}$ contain exactly the same elements (i.e. they are equal except for their orderings) are treated equally. Because our encoding will not differentiate between these two statements, we can consider the ordered profiles as multisets, as the ordering becomes irrelevant as a consequence of this anonymity axiom.

The fundamental purpose of outcome statements is to describe aggregation rules. Intuitively, when we give a list of outcome statements that is detailed enough, we are able to describe any single aggregation rule F. This allows us to define a correspondence between (consistent) aggregation rules and valuations on the set $\mathbb{S}^*$ that we have encoded. Here, a valuation on $\mathbb{S}^*$ decides for any profile $\mathbf{B} \in \mathrm{Mod}(\Gamma)^{+*}$ and $\varphi \in \mathcal{L}(X)^{\wedge,\vee}$ whether $\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle$ is true or $\neg\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle$ (which corresponds to the variable $[\![\mathbf{B}, \neg\varphi]\!] \in \mathbb{S}^*$) is true. In the encoding, any valuation on $\mathbb{S}^*$ then corresponds to a set of integers such that for any outcome statement, either its corresponding integer is in that set (if the outcome statement is true in the valuation) or its negation is (if the outcome statement is false in the valuation). Here we also follow the general idea in the work of Geist and Peters [GP17] and the implementation in Python by Endriss [End23].

However, not every such valuation actually corresponds to an aggregation rule. For example, a valuation on $\mathbb{S}^*$ could make $[\![\mathbf{B}, x_1 \wedge x_2]\!]$ true while it makes $[\![\mathbf{B}, x_1]\!]$ false, which does not correspond to any valid aggregation rule. As a consequence, we need to impose further restrictions on the valuations that are allowed in order to induce a one-to-one correspondence between certain valuations on $\mathbb{S}^*$ and consistent aggregation rules.

5.2.1 Valuation Restrictions as Axioms

We will introduce the different restrictions and denote them as axioms consisting of axiom instances that are propositional formulas over $\mathbb{S}^*$, so that they can be easily encoded as CNF-formulas over the propositional variables in $\mathbb{S}^*$.

Anonymity

As discussed in the first part of this thesis, our framework is anonymous, which means that we do not distinguish between the different voters. As a consequence, profiles are (unordered) multisets. This is due to the fact that we only care what ballots are submitted and not who submitted which ballot. However, in the encoding of the profiles, the order of the ballots actually does matter, e.g. $\{B, B'\} \neq \{B', B\}$. The following axiom considers ordered profiles that are equal as multisets, i.e. they are permutations of each other. These ordered profiles have the same corresponding multiset (see Definition 5.1), so they should behave identically in our encoding.

$$\mathcal{A}^{Anon} := \{\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle \leftrightarrow \langle\!\langle \mathbf{B'}, \varphi \rangle\!\rangle \mid \mathbf{B}, \mathbf{B'} \in \mathrm{Mod}(\Gamma)^{+*}, \varphi \in \mathcal{L}(X)^{\wedge,\vee} \text{ and } \mathbf{B} \overset{Multiset}{=} \mathbf{B'}\},$$

where $\mathbf{B} \overset{Multiset}{=} \mathbf{B'}$ means that the ordered lists of ballots $\mathbf{B}$ and $\mathbf{B'}$ are equal as multisets. For any two ordered profiles that satisfy this condition, a valuation on $\mathbb{S}^*$ that satisfies the anonymity axiom does not distinguish between statements of the form $\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle$ and $\langle\!\langle \mathbf{B'}, \varphi \rangle\!\rangle$: they are either both true or both false. This allows us to view all ordered profiles as different representatives of the same multiset.

Taking the contraposition of the two implications in the axiom instances, together with the observation that $\{\neg\varphi \mid \varphi \in \mathcal{L}(X)^{\wedge,\vee}\} = \mathcal{L}(X)^{\wedge,\vee}$, will give us the bi-implication for the universal outcome statements.

Integrity Constraint

The second restriction we introduce is that of the integrity constraint. It is both a restriction on the possible valuations on $\mathbb{S}^*$ and a restriction on the domain of input profiles. The integrity constraint is submitted in the form of a CNF-formula. That way, we can simply restrict the domain of profiles by checking for every profile that we encoded if all the ballots in that profile satisfy all the clauses of the constraint.

Convention 4. *The **integrity constraint** Γ is inputted in the form of a CNF-formula.*

Consequently, we want to constrain the output by disallowing any complete conjunction over X, i.e. a conjunction that either contains x or $\neg x$ (but not both) for every issue $x \in X$, that does not satisfy the integrity constraint. A complete conjunction is formally defined as follows.

Definition 5.3. *We say that a formula $\varphi \in \mathcal{L}(X)$ is a **complete conjunction** if $\varphi = \bigwedge_{\ell \in L} \ell$ such that $L \subset X^*$ and for every $x \in X$ either $x \in L$ or $\neg x \in L$ but not both.*

It will later become clear that, together with the other valuation restrictions introduced in this section, it suffices to only restrict these complete conjunctions in the outcome statements.

For the integrity constraint, we find the following axiom:

$$\mathcal{A}^{IC} := \{\neg\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle \mid \mathbf{B} \in \mathrm{Mod}(\Gamma)^{+*} \text{ and } \varphi \in \mathcal{L}(X)^{\wedge,\vee} \text{ is a complete conjunction such that } \varphi \not\models \Gamma\},$$

where we note that there is exactly one truth assignment on X that makes φ true, so $\varphi \not\models \Gamma$ then implies that this unique truth assignment does not satisfy Γ.

At Least One

The third restriction on the possible truth assignments on $\mathbb{S}^*$ is also implemented in the Jupyter Notebook by Endriss [End23], although we have adapted it to fit the framework of binary aggregation. It is the only axiom that is necessary to capture voting rules in the Python encoding of voting theory by Endriss.

As any aggregation rule should have a nonempty set of possible outcome vectors, there should be at least one existential outcome statement that is true for every input profile. Because this axiom can be combined with axioms later in this section, it will suffice to only consider the literals for the following axiom:

$$\mathcal{A}^{At\text{-}Least\text{-}One} := \{ \bigvee_{\ell \in X^*} \langle\!\langle \mathbf{B}, \ell \rangle\!\rangle \mid \mathbf{B} \in \mathrm{Mod}(\Gamma)^{+*} \}.$$

The axiom above states that for any profile $\mathbf{B} \in \mathrm{Mod}(\Gamma)^{+*}$, there should be at least one outcome vector that satisfies one of the literals in X^*.

Atomic Equivalence

For a literal $\ell \in X^*$, the formulas $\mathrm{con}(\ell)$ and $\mathrm{dis}(\ell)$ are semantically equal. For this reason, we need to define an axiom that will bind these formulas in our encoding. This is more of a technical restriction which is implemented for syntactical purposes, hence the seemingly trivial axiom instances. Before we can state it, we give one more definition.

Definition 5.4. *Let $\mathbf{B} \in Mod(\Gamma)^{+*}$, let $\ell \in X^*$ and let F be some aggregation rule, then we say that*

$$F \vDash \langle\!\langle \mathbf{B}, con(\ell) \rangle\!\rangle \iff F \vDash \langle\!\langle \mathbf{B}, \ell \rangle\!\rangle \iff F \vDash \langle\!\langle \mathbf{B}, dis(\ell) \rangle\!\rangle,$$

where we note that the satisfaction of an outcome statement with an ordered profile is defined in Definition 5.1.

We now define the axiom as follows:

$$A^{Atom\text{-}Equiv} := \{ \langle\!\langle \mathbf{B}, \mathrm{con}(\ell) \rangle\!\rangle \leftrightarrow \langle\!\langle \mathbf{B}, \mathrm{dis}(\ell) \rangle\!\rangle \mid \mathbf{B} \in \mathrm{Mod}(\Gamma)^{+*} \text{ and } \ell \in X^* \}.$$

Note here that from Definition 5.4 it follows that any consistent aggregation rule satisfies $\langle\!\langle \mathbf{B}, \mathrm{con}(\ell) \rangle\!\rangle$ if and only if it satisfies $\langle\!\langle \mathbf{B}, \mathrm{dis}(\ell) \rangle\!\rangle$, where both $\mathrm{con}(\ell)$ and $\mathrm{dis}(\ell)$ correspond to the same formula in $\mathcal{L}(X)$, namely ℓ.

Conjunction Upwards

Before we introduce the next axiom, we note the following.

Notation 5.2. *For the following four axioms, we will make use of subsets of X^*, denoted by L. We will either take the conjunction or the disjunction over the literals in L to denote different formulas. As $\bot, \top \notin \mathcal{L}(X)^{\wedge,\vee}$, we can never take the conjunction or disjunction over a set of literals L where both $x, \neg x \in L$ for some variable $x \in X$. For readability, we will not denote this in every axiom. Whenever we describe a set $L \subset X^*$ in the following four axioms, we assume that for no $x \in X^*$ we have that both $x \in L$ and $\neg x \in L$.*

It will become clear in the axioms that the axiom instances with literal sets that do not satisfy this condition are equivalent to $\top$ and thus do not add any information to the axiom.

In the semantics of outcome statements, we can never have that $\langle\!\langle \mathbf{B}, x_1 \rangle\!\rangle$ is true while neither $\langle\!\langle \mathbf{B}, x_1 \wedge x_2 \rangle\!\rangle$ nor $\langle\!\langle \mathbf{B}, x_1 \wedge \neg x_2 \rangle\!\rangle$ are true. This is due to the fact that the first statement implies that there is a (complete) outcome vector v that satisfies x_1, which means that v should also either satisfy x_2 or $\neg x_2$. The following axiom encapsulates and generalizes this idea.

$$\mathcal{A}^{Con\text{-}Up} := \{\langle\!\langle \mathbf{B}, \bigwedge_{\ell \in L} \ell \rangle\!\rangle \rightarrow \big(\langle\!\langle \mathbf{B}, x \wedge \bigwedge_{\ell \in L} \ell \rangle\!\rangle \vee \langle\!\langle \mathbf{B}, \neg x \wedge \bigwedge_{\ell \in L} \ell \rangle\!\rangle\big) \mid \mathbf{B} \in \mathrm{Mod}(\Gamma)^{+*}, L \subset X^* \text{ such that } x, \neg x \notin L\}.$$

Note here that whenever we would consider a set $L \subset X^*$ such that there exists a variable $x' \in X$ with both $x' \in L$ and $\neg x' \in L$, the antecedent in the axiom instance would never be satisfied as $\bigwedge_{\ell \in L} \ell \equiv \bot$.

Conjunction Downwards

If an outcome statement of the form $\langle\!\langle \mathbf{B}, \bigwedge_{\ell \in L} \ell \rangle\!\rangle$ holds, then for any subset $L' \subseteq L$, the outcome statement $\langle\!\langle \mathbf{B}, \bigwedge_{\ell \in L'} \ell \rangle\!\rangle$ should also hold: if there is an outcome vector that satisfies the conjunction of the literals in L, then it also satisfies any conjunction of the literals of a subset of L. This downwards relation is captured in an axiom as follows:

$$\mathcal{A}^{Con\text{-}Down} := \{\langle\!\langle \mathbf{B}, \bigwedge_{\ell \in L} \ell \rangle\!\rangle \rightarrow \langle\!\langle \mathbf{B}, \bigwedge_{\ell \in L'} \ell \rangle\!\rangle \mid \mathbf{B} \in \mathrm{Mod}(\Gamma)^{+*} \text{ and } L' \subseteq L \subset X^*\}.$$

Again, whenever we consider a set $L \subset X^*$ such that there is a variable $x \in X$ with both $x \in L$ and $\neg x \in L$, the antecedent in the axiom instance would never be satisfied as $\bigwedge_{\ell \in L} \ell \equiv \bot$.

Disjunction Upwards

Now we define two parallel axioms for disjunction. If $\langle\!\langle \mathbf{B}, \bigvee_{\ell \in L} \ell \rangle\!\rangle$ holds for some set of literals $L \subset X^*$, then for any superset $L' \supseteq L$, the statement $\langle\!\langle \mathbf{B}, \bigvee_{\ell \in L'} \ell \rangle\!\rangle$ should hold, as the outcome vector that satisfies one of the literals in L also satisfies one of the literals in L'. We define the following axiom:

$$\mathcal{A}^{Dis\text{-}Up} := \{\langle\!\langle \mathbf{B}, \bigvee_{\ell \in L} \ell \rangle\!\rangle \rightarrow \langle\!\langle \mathbf{B}, \bigvee_{\ell \in L'} \ell \rangle\!\rangle \mid \mathbf{B} \in \mathrm{Mod}(\Gamma)^{+*} \text{ and } L \subseteq L' \subset X^*\}.$$

Note here that whenever $x, \neg x \in L$ for some $x \in X$, we have that both $\bigvee_{\ell \in L} \ell \equiv \top$ and $\bigvee_{\ell \in L'} \ell \equiv \top$.

Disjunction Downwards

If $\langle\!\langle \mathbf{B}, \bigvee_{\ell \in L} \ell \rangle\!\rangle$ holds for some set $L \subset X^*$, then that means that for at least one of the literals $\ell \in L$, the statement $\langle\!\langle \mathbf{B}, \ell \rangle\!\rangle$ should hold because the outcome vector satisfying the disjunction satisfies one of the literals of the disjunction. This gives rise to the disjunction downwards axiom:

$$\mathcal{A}^{Dis\text{-}Down} := \{\langle\!\langle \mathbf{B}, \bigvee_{\ell \in L} \ell \rangle\!\rangle \rightarrow \big(\bigvee_{\ell \in L} \langle\!\langle \mathbf{B}, \ell \rangle\!\rangle\big) \mid \mathbf{B} \in \mathrm{Mod}(\Gamma)^{+*} \text{ and } L \subset X^*\}.$$

Note that whenever $x, \neg x \in L$ for some $x \in X$, it holds that $\big(\bigvee_{\ell \in L} \langle\!\langle \mathbf{B}, \ell \rangle\!\rangle\big)$ is always true.

Box implies Diamond

We define one more axiom in this section, which mirrors the fact that if some universal statement $[\![\mathbf{B}, \varphi]\!]$ holds, then because there is always at least one outcome vector, the existential statement

$\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle$ should also hold:

$$\mathcal{A}^{Box\text{-}Imp\text{-}Dia} := \{\neg\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle \rightarrow \langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle \mid \mathbf{B} \in \mathrm{Mod}(\Gamma)^{+*}, \varphi \in \mathcal{L}(X)^{\wedge,\vee}\}.$$

An interesting observation to make here is that when writing the instances of the axiom as clauses, we find disjunctions of the form $\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle \vee \langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle$ (given the equivalence $\neg p \rightarrow q \equiv p \vee q$), which may also be considered as some variant of the law of the excluded middle: for any formula φ, either there exists an outcome vector that satisfies it, or there is an outcome vector that satisfies its negation.

5.2.2 Correspondence Lemma

The axioms above show formulas that any truth assignment over the literals in $\mathbb{S}^*$ should satisfy, based on the semantics of the outcome statements. In the following lemma, we show that there is a direct correspondence between consistent aggregation rules on the one hand and truth assignments on $\mathbb{S}^*$ that satisfy all the axioms above on the other.

Definition 5.5. *We define the corpus of* ***restriction axioms*** *as follows:*

$$\mathbb{A}^{Rest} := \{\mathcal{A}^{Anon}, \mathcal{A}^{IC}, \mathcal{A}^{At\text{-}Least\text{-}One}, \mathcal{A}^{At\text{-}Equiv}, \mathcal{A}^{Con\text{-}Up}, \mathcal{A}^{Con\text{-}Down}, \mathcal{A}^{Dis\text{-}Up}, \mathcal{A}^{Dis\text{-}Down}, \mathcal{A}^{Box\text{-}Imp\text{-}Dia}\}.$$

Lemma 5.3. *There is a one-to-one correspondence between consistent aggregation rules* $F : \mathit{Mod}(\Gamma)^+ \rightarrow 2^{\mathit{Mod}(\Gamma)} \setminus \{\varnothing\}$ *and truth assignments on* $\mathbb{S}^*$ *that satisfy the axioms in* $\mathbb{A}^{Rest}$*. Moreover, in this correspondence, a consistent aggregation rule* F *satisfies an outcome statement* $s \in \mathbb{S}^*$ *if and only if that outcome statement (as a propositional variable) is true in the corresponding truth assignment on* $\mathbb{S}^*$*.*

Proof. Let Γ be some integrity constraint. First, we prove that for any consistent aggregation rule there is an assignment on $\mathbb{S}^*$ such that the condition above holds.

Let F be some consistent aggregation rule and let $\mathcal{V}$ be the truth assignment on $\mathbb{S}^*$ such that $\mathcal{V}$ makes a literal $s \in \mathbb{S}^*$ true if and only if it is satisfied by F, i.e. $F \vDash s$ (see Definition 5.1). The correspondence described in the lemma then holds by definition of $\mathcal{V}$. It only remains to show that $\mathcal{V}$ satisfies all the axioms in $\mathbb{A}^{Rest}$.

- First we show anonymity. By definition, an outcome statement $\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle \in \mathbb{S}^*$ is satisfied by F if and only if $F \vDash \langle\!\langle \mathbf{B''}, \varphi \rangle\!\rangle$, where $\mathbf{B''}$ is the multiset corresponding to the ordered profile $\mathbf{B} \in \mathrm{Mod}(\Gamma)^{+*}$ (see Definition 5.1). Now consider any formula $\varphi \in \mathcal{L}(X)^{\wedge,\vee}$ and any two ordered profiles $\mathbf{B}, \mathbf{B'} \in \mathrm{Mod}(\Gamma)^{+*}$ that have the same corresponding multiset, i.e. they contain exactly the same ballots, but they are ordered differently. Let $\mathbf{B''} \in \mathrm{Mod}(\Gamma)^{+}$ be the multiset that corresponds to both $\mathbf{B}$ and $\mathbf{B'}$. Then the following equivalences hold: $F \vDash \langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle$ if and only if $F \vDash \langle\!\langle \mathbf{B''}, \varphi \rangle\!\rangle$ if and only if $F \vDash \langle\!\langle \mathbf{B'}, \varphi \rangle\!\rangle$. Now it follows by definition of $\mathcal{V}$ that $\mathcal{V}$ makes $\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle$ true if and only if it makes $\langle\!\langle \mathbf{B'}, \varphi \rangle\!\rangle$ true. We conclude that $\mathcal{V}$ satisfies $\mathcal{A}^{Anon}$.

- $\mathcal{A}^{IC}$ is satisfied as for any complete conjunction $\varphi \in \mathcal{L}(X)^{\wedge,\vee}$ (see Definition 5.3) such that $\varphi \nvDash \Gamma$ and any ordered profile $\mathbf{B} \in \mathrm{Mod}(\Gamma)^{+*}$, it holds that $F \vDash \neg\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle$, as F is consistent. But then $\neg\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle$ is true in $\mathcal{V}$ by definition of $\mathcal{V}$. We conclude that $\mathcal{V}$ satisfies $\mathcal{A}^{IC}$.

- It is clear that $F \vDash \bigvee_{\ell \in X^*} \langle\!\langle \mathbf{B}, \ell \rangle\!\rangle$ for every $\mathbf{B} \in \mathrm{Mod}(\Gamma)^{+*}$, as $F(\mathbf{B}) \neq \varnothing$, so there is at least one outcome vector $v \in F(\mathbf{B})$, and for this vector it trivially holds that $v \vDash \ell$ for some $\ell \in X^*$.

From there it follows that for every $\mathbf{B} \in \mathrm{Mod}(\Gamma)^{+*}$, there is at least one $\ell \in X^*$ such that $\langle\!\langle \mathbf{B}, \ell \rangle\!\rangle$ is true in $\mathcal{V}$. We conclude that $\mathcal{V}$ satisfies $\mathcal{A}^{At\text{-}Least\text{-}One}$.

- It is immediately evident that F satisfies $\mathcal{A}^{At\text{-}Equiv}$. As a consequence, $\mathcal{V}$ does so as well.

- To show that $F \vDash \mathcal{A}^{Con\text{-}Up}$, suppose that $F \vDash \langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle$ for some $\mathbf{B} \in \mathrm{Mod}(\Gamma)^{+*}$, some conjunction $\varphi \in \mathcal{L}(X)^{\wedge\vee}$ and some issue-variable $x \in X$ such that neither x nor $\neg x$ appear in the conjunction φ. Let $\mathbf{B}'$ be the multiset corresponding to the ordered profile $\mathbf{B}$. Then there is an outcome vector $v \in F(\mathbf{B}')$ such that $v \vDash \varphi$ by definition. But as this outcome vector is a "complete" vector, i.e. it either accepts or rejects each issue in X, it also accepts or rejects the issue x. This means that either $v \vDash \varphi \wedge x$ or $v \vDash \varphi \wedge \neg x$. We conclude that $F \vDash \langle\!\langle \mathbf{B}', \varphi \wedge x \rangle\!\rangle$ or $F \vDash \langle\!\langle \mathbf{B}', \varphi \wedge \neg x \rangle\!\rangle$. We then return to our ordered profile $\mathbf{B} \in \mathrm{Mod}(\Gamma)^{+*}$ and find that $F \vDash \langle\!\langle \mathbf{B}, \varphi \wedge x \rangle\!\rangle$ or $F \vDash \langle\!\langle \mathbf{B}, \varphi \wedge \neg x \rangle\!\rangle$. In conclusion, we have that $F \vDash \langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle \to (\langle\!\langle \mathbf{B}, \varphi \wedge x \rangle\!\rangle \vee \langle\!\langle \mathbf{B}, \varphi \wedge \neg x \rangle\!\rangle)$. By definition of $\mathcal{V}$ we then find that $\mathcal{V}$ also satisfies $\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle \to (\langle\!\langle \mathbf{B}, \varphi \wedge x \rangle\!\rangle \vee \langle\!\langle \mathbf{B}, \varphi \wedge \neg x \rangle\!\rangle)$, so $\mathcal{V}$ satisfies $\mathcal{A}^{Con\text{-}Up}$ as we had taken an arbitrary profile and conjunction.

- The fact that $F \vDash \mathcal{A}^{Con\text{-}Down}$ follows from the simple observation that whenever $F \vDash \langle\!\langle \mathbf{B}, \bigwedge_{\ell \in L} \ell \rangle\!\rangle$ for some set of literals $L \subset X^*$, then for any subset $L' \subseteq L$ it should hold that $F \vDash \langle\!\langle \mathbf{B}, \bigwedge_{\ell \in L'} \ell \rangle\!\rangle$, as the outcome vector that satisfies $\bigwedge_{\ell \in L} \ell$ evidently also satisfies $\bigwedge_{\ell \in L'} \ell$. We conclude that for any $\mathbf{B} \in \mathrm{Mod}(\Gamma)^{+*}$ and $L' \subseteq L \subset X^*$, it holds that $F \vDash \langle\!\langle \mathbf{B}, \bigwedge_{\ell \in L} \ell \rangle\!\rangle \to \langle\!\langle \mathbf{B}, \bigwedge_{\ell \in L'} \ell \rangle\!\rangle$. From there it follows that $\mathcal{V} \vDash \langle\!\langle \mathbf{B}, \bigwedge_{\ell \in L} \ell \rangle\!\rangle \to \langle\!\langle \mathbf{B}, \bigwedge_{\ell \in' L} \ell \rangle\!\rangle$ by definition, so $\mathcal{V} \vDash \mathcal{A}^{Con\text{-}Down}$.

- To prove disjunction upwards, the essential observation is that whenever an outcome vector satisfies some disjunction $\bigvee_{\ell \in L} \ell$ for $L \subset X^*$, it will also satisfy $\bigvee_{\ell \in L'} \ell$ for any superset of literals $L' \supseteq L$, as it satisfies (at least) one literal in L which is then also in L'. We find that $F \vDash \mathcal{A}^{Dis\text{-}Up}$ so also that $\mathcal{V} \vDash \mathcal{A}^{Dis\text{-}Up}$.

- For disjunction downwards, note that if there is some outcome vector that satisfies $\bigvee_{\ell \in L} \ell$ for $L \subset X^*$, then that means that it satisfies at least one of the literals in L. We find that $F \vDash \mathcal{A}^{Dis\text{-}Down}$ which implies that $\mathcal{V} \vDash \mathcal{A}^{Dis\text{-}Down}$.

- The fact that F satisfies $\mathcal{A}^{Box\text{-}Imp\text{-}Dia}$ follows from the fact that $F(\mathbf{B})$ is never empty for any $\mathbf{B} \in \mathrm{Mod}(\Gamma)^{+}$, so there is always at least one outcome vector in $F(\mathbf{B})$. This means that whenever every outcome vector in $F(\mathbf{B})$ satisfies some formula φ, then there also exists an outcome vector that satisfies that formula φ. The translation to $\mathrm{Mod}(\Gamma)^{+*}$ and $\mathcal{V}$ then follows immediately.

Now we prove the other direction of the one-to-one correspondence. Let $\mathcal{V}$ be some valuation on $\mathbb{S}^*$ that satisfies all the axioms in $\mathbb{A}^{Rest}$. We need to find the corresponding consistent aggregation rule. We claim that the corresponding rule F is defined as follows. For every $\mathbf{B} \in \mathrm{Mod}(\Gamma)^{+}$, let $F(\mathbf{B})$ be exactly the set of (complete) outcome vectors v for which the literal $\langle\!\langle \mathbf{B}', \varphi_v \rangle\!\rangle$ is true in $\mathcal{V}$, where $\mathbf{B}' \in \mathrm{Mod}(\Gamma)^{+*}$ is some ordering of the ballots in the multiset $\mathbf{B}$ (see Definition 2.13 for the definition of φ_v). Note here that as $\mathcal{V}$ satisfies $\mathcal{A}^{Anon}$, the choice of $\mathbf{B}'$ is arbitrary.

First, we show that F, defined in this way, is consistent. Note that it cannot be the case that $F(\mathbf{B})$ contains an inconsistent outcome vector v for some $\mathbf{B} \in \mathrm{Mod}(\Gamma)^{+}$, as that would mean that for any $\mathbf{B}' \in \mathrm{Mod}(\Gamma)^{+*}$ that contains exactly the ballots that appear in $\mathbf{B}$, it is the case that

$\mathcal{V}$ satisfies the statement $\langle\!\langle \mathbf{B}', \varphi_v \rangle\!\rangle$ while $\varphi_v \nvDash \Gamma$ by assumption, which contradicts the fact that $\mathcal{V} \vDash \mathcal{A}^{IC}$. We conclude that F is consistent.

Consequently, we show that $F(\mathbf{B}) \neq \varnothing$ for any $\mathbf{B} \in \mathrm{Mod}(\Gamma)^+$. Consider an arbitrary $\mathbf{B} \in \mathrm{Mod}(\Gamma)^+$ and let $\mathbf{B}' \in \mathrm{Mod}(\Gamma)^{+*}$ be an ordered profile containing exactly the ballots in $\mathbf{B}$. From the fact that $\mathcal{V} \vDash \mathcal{A}^{\textit{At-Least-One}}$, it follows that there is at least one $\ell \in X^*$ such that $\mathcal{V} \vDash \langle\!\langle \mathbf{B}', \ell \rangle\!\rangle$. Now we can recursively apply the conjunction upwards axiom, for which we have that $\mathcal{V} \vDash \mathcal{A}^{\textit{Con-Up}}$, and conclude that there should be at least one vector $v \in \{0,1\}^m$ such that $\mathcal{V} \vDash \langle\!\langle \mathbf{B}', \varphi_v \rangle\!\rangle$. From this we infer that $v \in F(\mathbf{B})$ by definition of F, so $F(\mathbf{B}) \neq \varnothing$.

It remains to show that an outcome statement $s \in \mathbb{S}^*$ is satisfied by F if and only if it is true in $\mathcal{V}$. We will only prove it for existential outcome statements (for both directions); the universal statements then follow by taking the contrapositive of both directions of the equivalence, together with the observation that $\{\neg\varphi \mid \varphi \in \mathcal{L}(X)^{\wedge,\vee}\} = \mathcal{L}(X)^{\wedge,\vee}$.

First, we prove the direction from left to right. Suppose that $F \vDash s$ for some existential outcome statement $s \in \mathbb{S}^*$. We make a case distinction.

- Suppose that $s = \langle\!\langle \mathbf{B}, \ell \rangle\!\rangle$ for some $\ell \in X^*$ and $\mathbf{B} \in \mathrm{Mod}(\Gamma)^{+*}$ and let $\mathbf{B}' \in \mathrm{Mod}(\Gamma)^+$ be the multiset corresponding to $\mathbf{B}$. Then that means that there is some $v \in F(\mathbf{B}')$ such that $v \vDash \ell$, which implies that ℓ appears in the conjunction φ_v. We then find that $\mathcal{V} \vDash \langle\!\langle \mathbf{B}, \varphi_v \rangle\!\rangle$ by definition of F. Now as $\mathcal{V} \vDash \mathcal{A}^{\textit{Con-Down}}$ and ℓ appears as a literal in the conjunction φ_v, we may conclude that $\mathcal{V} \vDash \langle\!\langle \mathbf{B}, \ell \rangle\!\rangle$, so $\mathcal{V} \vDash s$.

- Now suppose that $s = \langle\!\langle \mathbf{B}, \bigwedge_{\ell \in L} \ell \rangle\!\rangle$ where $|L| > 1$ and $L \subset X^*$. Let $\mathbf{B}'$ again be the multiset corresponding to the ordered profile $\mathbf{B}$, then there is an outcome vector $v \in F(\mathbf{B}')$ such that $v \vDash \bigwedge_{\ell \in L} \ell$, but that means that any literal that appears in L also appears in φ_v. From the definition of F, it then follows that $\mathcal{V}$ makes $\langle\!\langle \mathbf{B}, \varphi_v \rangle\!\rangle$ true. As any literal in L appears in the conjunction φ_v, we can apply the conjunction downwards rule (as $\mathcal{V} \vDash \mathcal{A}^{\textit{Con-Down}}$) and conclude that $\mathcal{V} \vDash \langle\!\langle \mathbf{B}, \bigwedge_{\ell \in L} \ell \rangle\!\rangle$.

- Finally, suppose that $s = \langle\!\langle \mathbf{B}, \bigvee_{\ell \in L} \ell \rangle\!\rangle$, with $|L| > 1$ and $L \subset X^*$ and let $\mathbf{B}'$ be the multiset corresponding to $\mathbf{B}$. Then there exists an outcome $v \in F(\mathbf{B}')$ such that $v \vDash \bigvee_{\ell \in L} \ell$. This means that there is a literal $\ell \in L$ such that $v \vDash \ell$. But that means that ℓ appears in the conjunction φ_v. As $v \in F(\mathbf{B}')$, we find that $\mathcal{V} \vDash \langle\!\langle \mathbf{B}, \varphi_v \rangle\!\rangle$ by definition of F. As the literal ℓ appears in the conjunction φ_v, we may again use the conjunction downwards axiom and find that $\mathcal{V} \vDash \langle\!\langle \mathbf{B}, \ell \rangle\!\rangle$. Finally, we may then apply the disjunction upwards axiom (since $\mathcal{V} \vDash \mathcal{A}^{\textit{Dis-Up}}$) and find that $\mathcal{V} \vDash \langle\!\langle \mathbf{B}, \bigvee_{\ell' \in L} \ell' \rangle\!\rangle$, where we note that $\ell \in L$.

For the proof of the other direction of the implication, suppose that $\mathcal{V} \vDash s$ for some outcome statement $s \in \mathbb{S}^*$. We again make a case distinction.

- If $s = \langle\!\langle \mathbf{B}, \ell \rangle\!\rangle$ for some $\ell \in X^*$, we may use the fact that $\mathcal{V} \vDash \mathcal{A}^{\textit{Con-Up}}$ and repeatedly apply instances of it to conclude that there is some complete conjunction φ_v with $\varphi_v \vDash \ell$ (so ℓ appears in the conjunction φ_v) and $\mathcal{V} \vDash \langle\!\langle \mathbf{B}, \varphi_v \rangle\!\rangle$. Then it follows from the definition of F that $v \in F(\mathbf{B}')$, where $\mathbf{B}'$ is the multiset corresponding to the ordered profile $\mathbf{B} \in \mathrm{Mod}(\Gamma)^{+*}$. As $\varphi_v \vDash \ell$ and $v \in F(\mathbf{B}')$, we conclude that $F \vDash \langle\!\langle \mathbf{B}', \ell \rangle\!\rangle$, which gives us that $F \vDash \langle\!\langle \mathbf{B}, \ell \rangle\!\rangle$ by Definition 5.1.

- Now suppose that $s = \langle\!\langle \mathbf{B}, \bigwedge_{\ell \in L} \ell \rangle\!\rangle$ for some literal set L such that $|L| > 1$ and $L \subset X^*$. Then we can again use the fact that $\mathcal{V} \vDash \mathcal{A}^{\textit{Con-Up}}$ and find a complete conjunction φ_v such that

$\varphi_v \vDash \bigwedge_{\ell \in L} \ell$ (so every literal $\ell \in L$ appears in the conjunction φ_v) and $\mathcal{V} \vDash \langle\!\langle \mathbf{B}, \varphi_v \rangle\!\rangle$. From there it follows by definition of F that $v \in F(\mathbf{B'})$ where $\mathbf{B'}$ is again the multiset corresponding to $\mathbf{B}$. We conclude that $F \vDash \langle\!\langle \mathbf{B'}, \varphi_v \rangle\!\rangle$, so we also have that $F \vDash \langle\!\langle \mathbf{B'}, \bigwedge_{\ell \in L} \ell \rangle\!\rangle$, as any literal in L appears in φ_v. We conclude that $F \vDash \langle\!\langle \mathbf{B}, \bigwedge_{\ell \in L} \ell \rangle\!\rangle$.

- Finally, suppose that $s = \langle\!\langle \mathbf{B}, \bigvee_{\ell \in L} \ell \rangle\!\rangle$ for $|L| > 1$ and $L \subset X^*$. Then we may use the fact that $\mathcal{V} \vDash \mathcal{A}^{Dis\text{-}Down}$ and find that there is a literal $\ell \in L$ for which $\mathcal{V} \vDash \langle\!\langle \mathbf{B}, \ell \rangle\!\rangle$. As $\mathcal{V} \vDash \mathcal{A}^{Con\text{-}Up}$, we conclude that there is a complete conjunction φ_v such that ℓ appears in φ_v and $\mathcal{V} \vDash \langle\!\langle \mathbf{B}, \varphi_v \rangle\!\rangle$. This means that $v \in F(\mathbf{B'})$ by definition of F, where $\mathbf{B'}$ is the multiset corresponding to $\mathbf{B}$. Then it also holds that $F \vDash \langle\!\langle \mathbf{B'}, \ell \rangle\!\rangle$ as ℓ appears in φ_v and $v \in F(\mathbf{B'})$. We conclude that $F \vDash \langle\!\langle \mathbf{B'}, \ell \rangle\!\rangle$ so it also holds that $F \vDash \langle\!\langle \mathbf{B'}, \bigvee_{\ell \in L} \ell \rangle\!\rangle$, as $\ell \in L$. Finally, we find that it also holds that $F \vDash \langle\!\langle \mathbf{B}, \bigvee_{\ell \in L} \ell \rangle\!\rangle$.

This concludes our case distinction. □

A careful reader might note here that we have not used every axiom from $\mathbb{A}^{Rest}$ for the proof of the second part of the correspondence, which means that we could have proven the lemma above for a subset of $\mathbb{A}^{Rest}$. However, in Chapter 6, the importance of the different axioms in $\mathbb{A}^{Rest}$ will become clear. The ones that are not used in the proof above and thus could have been omitted will improve the explainability of the justifications that are generated in the following chapter.

5.3 Encoding Axioms

In the previous section, we have proven the correspondence between consistent aggregation rules and truth valuations on $\mathbb{S}^*$ that satisfy the axioms in $\mathbb{A}^{Rest}$. In this section, we will conclude the implementation of the framework by showing how to encode the axioms in Python. First, we show how to encode the axioms introduced in the previous section. Consequently, we show how to encode the axioms from Section 2.3. Again, the encoding that is described here can be found in the Jupyter Notebook on `https://zenodo.org/records/16614542` [Jon25].

Every axiom is a set of axiom instances, which will be propositional formulas over $\mathbb{S}^*$ in our encoding. Instead of encoding a set of these axiom instances, we follow the work of Geist and Peters [GP17] and Endriss [End23] and encode the axioms as CNF-formulas, where each clause in the formula corresponds to an axiom instance. The CNF-formula representing an axiom is then simply a big conjunction over all these clauses. However, not every axiom instance can be written as a disjunction. Because of that, we might split some axiom instances into multiple smaller clauses. For example, an axiom instance of the form $p \leftrightarrow q$ we might write as $(\neg p \vee q) \wedge (\neg q \vee p)$; this does not affect the CNF-structure as we are still taking a conjunction over a set of disjunctions.

When provided with the CNF-formula, Endriss [End23] has shown how to encode it in Python through the use of one or more for-loops.

5.3.1 Encoding the Valuation Restrictions

First, we encode the axioms described in Section 5.2.

For **anonymity**, we want to have a clause for every two ordered profiles that are equal as multisets and every formula in our fragment. Because we cannot rewrite a bi-implication as a disjunction, we have rewritten it as the conjunction of two disjunctions (each representing one of

the two implications). This gives rise to the following CNF-formula.

$$\varphi^{Anon} := \bigwedge_{\substack{\mathbf{B},\mathbf{B'}\in \mathrm{Mod}(\Gamma)^{+*}\colon \\ \mathbf{B} \overset{Multiset}{=} \mathbf{B'}}} \bigwedge_{\varphi\in\mathcal{L}(X)^{\wedge,\vee}} \big((\neg\langle\!\langle \mathbf{B},\varphi\rangle\!\rangle \vee \langle\!\langle \mathbf{B'},\varphi\rangle\!\rangle) \wedge (\neg\langle\!\langle \mathbf{B'},\varphi\rangle\!\rangle \vee \langle\!\langle \mathbf{B},\varphi\rangle\!\rangle)\big)$$

The encoding of (the outcome restriction of) the **integrity constraint** is very straightforward and follows directly from its encoding as an axiom.

$$\varphi^{IC} := \bigwedge_{\mathbf{B}\in\mathrm{Mod}(\Gamma)^{+*}} \bigwedge_{\substack{\varphi\in\mathcal{L}(X)^{\wedge,\vee} \\ \varphi \text{ is a complete conjunction} \\ \text{such that } \varphi\not\vDash\Gamma}} \neg\langle\!\langle \mathbf{B},\varphi\rangle\!\rangle$$

The axiom instances of the **at least one** axiom are already written as disjunctions, which makes the encoding of the CNF-formula evident.

$$\varphi^{At\text{-}Least\text{-}One} := \bigwedge_{B\in\mathrm{Mod}(\Gamma)^{+*}} \bigvee_{\ell\in X^*} \langle\!\langle \mathbf{B},\ell\rangle\!\rangle.$$

Like for anonymity, the **atomic equivalence** axiom also requires the rewriting of an equivalence as a conjunction of two clauses.

$$\varphi^{At\text{-}Equiv} := \bigwedge_{B\in\mathrm{Mod}(\Gamma)^{+*}} \bigwedge_{\ell\in X^*} \big((\neg\langle\!\langle \mathbf{B},\mathrm{dis}(\ell)\rangle\!\rangle \vee \langle\!\langle \mathbf{B},\mathrm{con}(\ell)\rangle\!\rangle) \wedge (\neg\langle\!\langle \mathbf{B},\mathrm{con}(\ell)\rangle\!\rangle \vee \langle\!\langle \mathbf{B},\mathrm{dis}(\ell)\rangle\!\rangle)\big)$$

The encoding of the rest of the valuation restriction axioms is straightforward and happens analogously to the ones above. For that reason, we will not show them here.

Following the work of Endriss [End23], we can now simply encode the axioms above by taking a for-loop for every conjunction and consequently denoting every clause as a list of integers that correspond to outcome statements. For example, the formula above is encoded as follows. At the end of Section 5.1, the encoding of the 'posLiteral' function is shown. The 'literal_outcomes' function simply returns the vectors representing the set X^*.

```
def cnf_atomic_equivalence():
cnf = []
for b in IC_profiles:
    for v in literal_outcomes():
        cnf.append([negLiteral(b, v, 1), posLiteral(b, v, 0)])
        cnf.append([negLiteral(b, v, 0), posLiteral(b, v, 1)])

return cnf
```

5.3.2 Encoding the Binary Aggregation Axioms

Now we turn to the encoding of the axioms introduced in Section 2.3. We start with **faithfulness**.

For every ordered profile that contains only one ballot, so it is of the form $\mathbf{B} = (B)$, we want to add the clause $[\![\mathbf{B},\varphi_B]\!]$. This is encoded by adding a unit clause containing the outcome statement $\neg\langle\!\langle \mathbf{B},\neg\varphi_B\rangle\!\rangle$ for each ordered profile of size one.

$$\varphi^{Fai} := \bigwedge_{B\in\mathrm{Mod}(\Gamma) \text{ and } \mathbf{B}=(B)} \neg\langle\!\langle \mathbf{B},\neg\varphi_B\rangle\!\rangle$$

The axiom instances of the **homogeneity** axiom are bi-implications. We will rewrite these clauses as conjunctions over two smaller clauses that represent both directions of the bi-implication. We note the following equivalences:

$$[\![\mathbf{B}, \varphi]\!] \leftrightarrow [\![k\mathbf{B}, \varphi]\!] = \neg\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle \leftrightarrow \neg\langle\!\langle k\mathbf{B}, \neg\varphi \rangle\!\rangle \equiv (\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle \vee \neg\langle\!\langle k\mathbf{B}, \neg\varphi \rangle\!\rangle) \wedge (\neg\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle \vee \langle\!\langle k\mathbf{B}, \neg\varphi \rangle\!\rangle).$$

As it is easy to see that $\{\neg\varphi \mid \varphi \in \mathcal{L}(X)^{\wedge,\vee}\} = \mathcal{L}(X)^{\wedge,\vee}$ (through the use of De Morgan's laws), we do not have to negate all formulas φ in the rightmost formula above and can instead simply use φ, as we iterate over every formula in $\mathcal{L}(X)^{\wedge,\vee}$. Before we can state the CNF-formula, we need to define the addition operator for ordered profiles.

Definition 5.6. *Let $\mathbf{B}', \mathbf{B}'' \in Mod(\Gamma)^{+*}$ with $\mathbf{B}' = (B'_1, \ldots, B'_i)$ and $\mathbf{B}'' = (B''_1, \ldots, B''_j)$, then we define the **addition of these ordered profiles** as follows:*

$$B' + B'' := (B'_1, \ldots, B'_i, B''_1, \ldots, B''_j).$$

For some $k \in \mathbb{N}_{>0}$ and some profile $\mathbf{B} \in Mod(\Gamma)^{+*}$, we then define

$$k\mathbf{B} := \underbrace{\mathbf{B} + \cdots + \mathbf{B}}_{k \text{ times}}.$$

Now we can state the CNF-formula for homogeneity.

$$\varphi^{Hom} := \bigwedge_{\mathbf{B} \in \mathrm{Mod}(\Gamma)^{+*}} \left(\bigwedge_{\substack{\mathbf{B}' \in \mathrm{Mod}(\Gamma)^{+*}: \\ \mathbf{B}' = k\mathbf{B} \text{ for some } k \in \mathbb{N}_{>0}}} \left(\bigwedge_{\varphi \in \mathcal{L}(X)^{\wedge,\vee}} \left(\left(\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle \vee \neg\langle\!\langle \mathbf{B}', \varphi \rangle\!\rangle \right) \wedge \left(\neg\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle \vee \langle\!\langle \mathbf{B}', \varphi \rangle\!\rangle \right) \right) \right) \right)$$

Remark 5.3. *First, we note that we have lost certain instances in our encoding that are described in the axiom in Section 2.3. This is due to the fact that we have only encoded a fragment of $\mathcal{L}(X)$. However, it is easy to see that this encoding is still equivalent to the original homogeneity axiom presented in Section 1.4. This follows from the fact that for every outcome $v \in \{0,1\}^m$, we have that $\langle\!\langle \mathbf{B}, \varphi_v \rangle\!\rangle \leftrightarrow \langle\!\langle k\mathbf{B}, \varphi_v \rangle\!\rangle$ is a subformula of φ^{Hom}. As a consequence, for every aggregation rule F that satisfies this formula, we get that $F(\mathbf{B}) = F(k\mathbf{B})$ (because $v \in F(\mathbf{B})$ if and only if $v \in F(k\mathbf{B})$) so homogeneity is satisfied. Conversely, it is easy to see that any aggregation rule that satisfies homogeneity also satisfies the formula above.*

Remark 5.4. *Another important observation to make here is the following: we are now only considering the ordered profiles $\mathbf{B}, \mathbf{B}' \in Mod(\Gamma)^{+*}$ where $\mathbf{B}' = k\mathbf{B}$ (for some $k \in \mathbb{N}_{>0}$) in that specific ordering. However, in our anonymous framework we do not distinguish between different orderings of the profiles, and we only care if $\mathbf{B}'$ contains exactly the same ballots as the ordered profile $k\mathbf{B}$. In that sense, our CNF-formula does not cover all the cases and is thus inadequate. However, the anonymity axiom accounts for the cases that φ^{Hom} does not cover, as it gives us the equivalence of any two outcome statements $\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle$ and $\langle\!\langle \mathbf{B}', \varphi \rangle\!\rangle$ where $\mathbf{B}$ and $\mathbf{B}'$ are different orderings of the same multiset of ballots.*

The two unanimity axioms are straightforward to implement, as they both consist of unit clauses. By slight abuse of notation, we consider elements of the ordered profiles. First we show **weak**

unanimity.

$$\varphi^{Weak\text{-}Un} := \bigwedge_{\mathbf{B}\in \mathrm{Mod}(\Gamma)^{+*}} \bigwedge_{\substack{\ell\in X^*:\\ \forall B\in\mathbf{B}: B\vDash\ell}} \langle\!\langle \mathbf{B}, \ell \rangle\!\rangle$$

For **strong unanimity**, we want a universal outcome statement instead of an existential one.

$$\varphi^{Strong\text{-}Un} := \bigwedge_{\mathbf{B}\in \mathrm{Mod}(\Gamma)^{+*}} \bigwedge_{\substack{\ell\in X^*:\\ \forall B\in\mathbf{B}: B\vDash\ell}} \neg\langle\!\langle \mathbf{B}, \neg\ell \rangle\!\rangle$$

The axiom instances in the **reinforcement** axiom are the most complex $\mathcal{L}(O)$-formulas out of all the axioms (see Section 2.3). However, we are able to split a single axiom instance into three different formulas that are easily rewritable as clauses. An axiom instance of the reinforcement axiom is of the following form:

$$\big(\langle\!\langle \mathbf{B}', \varphi_v \rangle\!\rangle \wedge \langle\!\langle \mathbf{B}'', \varphi_v \rangle\!\rangle\big) \to \big(((\llbracket \mathbf{B}', \varphi \rrbracket \vee \llbracket \mathbf{B}'', \varphi \rrbracket) \to \llbracket \mathbf{B}, \varphi \rrbracket) \wedge ((\langle\!\langle \mathbf{B}', \varphi_w \rangle\!\rangle \wedge \langle\!\langle \mathbf{B}'', \varphi_w \rangle\!\rangle) \to \langle\!\langle \mathbf{B}, \varphi_w \rangle\!\rangle)\big).$$

This formula is equivalent to the conjunction of the following three formulas:

- $\big(\langle\!\langle \mathbf{B}', \varphi_v \rangle\!\rangle \wedge \langle\!\langle \mathbf{B}'', \varphi_v \rangle\!\rangle \wedge \llbracket \mathbf{B}', \varphi \rrbracket\big) \to \llbracket \mathbf{B}, \varphi \rrbracket$,
- $\big(\langle\!\langle \mathbf{B}', \varphi_v \rangle\!\rangle \wedge \langle\!\langle \mathbf{B}'', \varphi_v \rangle\!\rangle \wedge \llbracket \mathbf{B}'', \varphi \rrbracket\big) \to \llbracket \mathbf{B}, \varphi \rrbracket$,
- $\big(\langle\!\langle \mathbf{B}', \varphi_v \rangle\!\rangle \wedge \langle\!\langle \mathbf{B}'', \varphi_v \rangle\!\rangle \wedge \langle\!\langle \mathbf{B}', \varphi_w \rangle\!\rangle \wedge \langle\!\langle \mathbf{B}'', \varphi_w \rangle\!\rangle\big) \to \langle\!\langle \mathbf{B}, \varphi_w \rangle\!\rangle$.

The three formulas above are all easy to rewrite as clauses, as the negation of the antecedent is a disjunction by De Morgan's laws in each of the formulas.

Now we are ready to present the CNF-formula for the encoding of the axiom.

$$\begin{aligned}\varphi^{Rei} := \bigwedge_{\substack{\mathbf{B},\mathbf{B}',\mathbf{B}''\in \mathrm{Mod}(\Gamma)^{+*}:\\ \mathbf{B}=\mathbf{B}'+\mathbf{B}''}} \bigwedge_{v,w\in\{0,1\}^m} \bigwedge_{\varphi\in\mathcal{L}(X)^{\wedge,\vee}} \Big(&\big(\neg\langle\!\langle \mathbf{B}', \varphi_v \rangle\!\rangle \vee \neg\langle\!\langle \mathbf{B}'', \varphi_v \rangle\!\rangle \vee \langle\!\langle \mathbf{B}', \neg\varphi \rangle\!\rangle \vee \neg\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\big) \\ &\wedge\big(\neg\langle\!\langle \mathbf{B}', \varphi_v \rangle\!\rangle \vee \neg\langle\!\langle \mathbf{B}'', \varphi_v \rangle\!\rangle \vee \langle\!\langle \mathbf{B}'', \neg\varphi \rangle\!\rangle \vee \neg\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\big) \\ &\wedge\big(\neg\langle\!\langle \mathbf{B}', \varphi_v \rangle\!\rangle \vee \neg\langle\!\langle \mathbf{B}'', \varphi_v \rangle\!\rangle \vee \neg\langle\!\langle \mathbf{B}', \varphi_w \rangle\!\rangle \vee \neg\langle\!\langle \mathbf{B}'', \varphi_w \rangle\!\rangle \vee \langle\!\langle \mathbf{B}, \varphi_w \rangle\!\rangle\big)\Big)\end{aligned}$$

Remark 5.5. *As for homogeneity, we observe that we are now only considering the triples of ordered profiles* $\mathbf{B}, \mathbf{B}', \mathbf{B}'' \in \mathit{Mod}(\Gamma)^{+*}$ *where* $\mathbf{B} = \mathbf{B}' + \mathbf{B}''$ *in that specific ordering. As explained in Remark 5.4,* φ^{Anon} *will make sure that our encoding accounts for the cases that are not covered in* φ^{Rei}.

Remark 5.6. *Finally, we note that, as for the encoding of homogeneity, we have also lost several axiom instances in this encoding, as a consequence of the fact that we have only encoded a fragment of* $\mathcal{L}(X)$. *Analogously to what we showed in Remark 5.3, it is almost immediate that our formula still encodes the original axiom presented in Section 1.4. The formulas of the form* φ_v *together with the first two clauses in* φ^{Rei} *force that* $F(\mathbf{B}) \subseteq F(\mathbf{B}')$ *and* $F(\mathbf{B}) \subseteq F(\mathbf{B}'')$ *whenever* $F(\mathbf{B}') \cap F(\mathbf{B}'') \neq \varnothing$*; note here that we consider the instances for* $\varphi = \neg\varphi_v$. *The final clause gives us that* $F(\mathbf{B}') \cap F(\mathbf{B}'') \subseteq F(\mathbf{B})$.

The **monotonicity** axiom instances make use of universal outcome statement. Before we can state the CNF-formula, we have to define the notion of an ℓ-improvement (see Definition 1.8) for ordered profiles.

Definition 5.7. *We say that an ordered profile* $\mathbf{B}' \in Mod(\Gamma)^{+*}$ *is an* ℓ*-improvement of* $\mathbf{B} \in Mod(\Gamma)^{+*}$ *if* $\mathbf{B}$ *and* $\mathbf{B}'$ *are equal except for one index, say* i*, and* B_i *and* B_i' *are the same ballot except for the fact that* $B_i' \vDash \ell$ *while* $B_i \nvDash \ell$*.*

This definition is weaker than the original definition because it requires the ballots that differ to be in the exact same index of the ordered profiles. However, this will again be resolved by the anonymity axiom.

For the encoding of the clauses, recall that the axiom instances of monotonicity use universal outcome statements. We find the following formula.

$$\varphi^{Mon} \coloneqq \bigwedge_{\substack{\mathbf{B},\mathbf{B}' \in \mathrm{Mod}(\Gamma)^{+*}: \\ \mathbf{B}' \text{ is an } \ell\text{-improvement of } \mathbf{B}}} \langle\!\langle \mathbf{B}, \neg\ell \rangle\!\rangle \vee \neg \langle\!\langle \mathbf{B}', \neg\ell \rangle\!\rangle$$

Now we encode the **majority-preservation** axiom. First, we need to redefine $m(\mathbf{B})$ for ordered profiles.

Definition 5.8. *For an ordered profile* $\mathbf{B} \in Mod(\Gamma)^{+*}$*, the* ***majority set*** *is defined as follows:* $m(\mathbf{B}) \coloneqq m(\mathbf{B}')$*, where* $\mathbf{B}'$ *is the multiset corresponding to* $\mathbf{B}$ *and* $m(\mathbf{B})$ *is defined as in Definition 1.11. The set ext*$(m(\mathbf{B}))$ *is then simply defined as in Definition 1.11. Analogously, we define* $N(\mathbf{B}, \varphi) \coloneqq N(\mathbf{B}', \varphi)$*.*

Before we are ready to spell out the CNF, we face one more problem. Our fragment of $\mathcal{L}(X)$ does not contain formulas of the form $\varphi_{\mathrm{ext}(m(\mathbf{B}))}$, as this is a disjunction of conjunctions (see Definition 2.13) and $\mathcal{L}(X)^{\wedge,\vee}$ only allows for either conjunctions or disjunctions. As a solution, we let $\varphi^*_{\mathrm{ext}(m(\mathbf{B}))}$ simply be the conjunction over all the literals in $m(\mathbf{B})$. Then it holds that for any ordered profile $\mathbf{B} \in \mathrm{Mod}(\Gamma)^{+*}$ with $\mathrm{ext}(m(\mathbf{B})) \neq \varnothing$ and any consistent aggregation rule F that $F \vDash [\![\mathbf{B}, \varphi^*_{\mathrm{ext}(m(\mathbf{B}))}]\!]$ if and only if $F \vDash [\![\mathbf{B}, \varphi_{\mathrm{ext}(m(\mathbf{B}))}]\!]$. This follows immediately from the fact that $\varphi_{\mathrm{ext}(m(\mathbf{B}))}$ describes the disjunction over all the possible extensions of the conjunction $\varphi^*_{\mathrm{ext}(m(\mathbf{B}))}$.

Now we define the CNF-formula.

$$\varphi^{Maj} \coloneqq \Big(\bigwedge_{\mathbf{B} \in \mathrm{Mod}(\Gamma)^{+*}} \bigwedge_{v \in \mathrm{ext}(m(\mathbf{B}))} \langle\!\langle \mathbf{B}, \varphi_v \rangle\!\rangle \Big) \wedge \Big(\bigwedge_{\substack{\mathbf{B} \in \mathrm{Mod}(\Gamma)^{+*}: \\ \mathrm{ext}(m(\mathbf{B})) \neq \varnothing}} \neg \langle\!\langle \mathbf{B}, \neg \varphi^*_{\mathrm{ext}(m(\mathbf{B}))} \rangle\!\rangle \Big)$$

To conclude the section, we provide the CNF-formula for the **cancellation** axiom. As the axiom instances are propositional variables, we do not need to rewrite them.

$$\varphi^{Can} \coloneqq \bigwedge_{\substack{\mathbf{B} \in \mathrm{Mod}(\Gamma)^{+*}: \\ \forall x \in X: \\ N(\mathbf{B},x) = N(\mathbf{B},\neg x)}} \bigwedge_{v \in \mathrm{Mod}(\Gamma)} \langle\!\langle \mathbf{B}, \varphi_v \rangle\!\rangle$$

As we have already carefully considered and explained the decisions that we made in the encoding during the chapter, we will not provide a separate discussion for this chapter. We compactly state our results.

First, we have shown how to encode the framework of binary aggregation by following the work of Geist and Peters [GP17] and Endriss [End23]. Consequently, we encoded aggregation rules through axioms that restrict the possible truth valuations on the encoded propositional variables. Finally, we showed how to encode the axioms introduced in Section 2.3.

Chapter 6

Generating Justifications

Now that we have implemented the framework of binary aggregation in Python, we can continue with the main goal of this implementation: generating justifications. The algorithm we will introduce in this chapter consists of two parts. First, we show a method of generating a minimally unsatisfiable subset that resembles an unstructured justification, following the work of Boixel and Endriss [BE20], but adapting it to the framework of binary aggregation. This is shown in Section 6.1. In Section 6.2 we then give an algorithm that structures an unstructured justification by generating a tree that resembles the tableaux defined in Chapter 3.

6.1 Generating Unstructured Justifications

In this section, we show an algorithm that generates a set of axiom instances that resembles an unstructured justification. As stated before, we will follow the method introduced by Boixel and Endriss [BE20]. In their algorithm, they check the satisfiability of different sets of axiom instances with the help of a SAT solver.

A SAT solver is a decision procedure that takes as input a propositional formula and outputs whether the formula is satisfiable or not. For an extensive and detailed description of SAT solvers, we propose the work of Biere et al. [Bie+09]. In this thesis, a SAT solver will be used to check whether certain sets of axiom instances and outcome statements are satisfiable together. We will also use an algorithm based on a SAT solver that computes a *minimally unsatisfiable subset*, which is defined as follows.

Definition 6.1. *We define two notions of an MUS, one in terms of satisfiability by aggregation rules and one in terms of satisfiability by valuations on $\mathbb{S}^*$.*

- *Let $\mathcal{A}$ be some set of axiom instances. In this chapter, we will say that $\mathcal{A}$ is **satisfiable** if there is a consistent aggregation rule F such that $F \vDash \mathcal{A}$ (see Definition 2.7). The set $\mathcal{A}$ is then **unsatisfiable** if there is no consistent aggregation rule F such that $F \vDash \mathcal{A}$.*

  *A **minimally unsatisfiable subset (MUS)** of an unsatisfiable set $\mathcal{A}$ is a subset of axiom instances $\mathcal{A}' \subseteq \mathcal{A}$ such that $\mathcal{A}'$ is unsatisfiable while for every proper subset $\mathcal{A}'' \subsetneq \mathcal{A}'$ it holds that $\mathcal{A}''$ is satisfiable.*

- *For the encoding $\mathbb{S}^*$ and any set of propositional formulas $\mathcal{A}$ over $\mathbb{S}^*$, we say that $\mathcal{A}$ is **unsatisfiable** if there is no valuation $\mathcal{V}$ on $\mathbb{S}^*$ that makes all the formulas in $\mathcal{A}$ true. For any*

*subset $\mathcal{A}' \subseteq \mathcal{A}$ of an unsatisfiable set $\mathcal{A}$, we say that $\mathcal{A}'$ is an **MUS** of $\mathcal{A}$ if $\mathcal{A}'$ is unsatisfiable but every proper subset of $\mathcal{A}'$ is satisfiable (by a valuation on $\mathbb{S}^*$).*

The specific MUS extraction algorithm that we will use is the *deletion-based MUS extractor* from the paper by Marques-Silva [Mar10]. The MUS that we will extract is a subset of some unsatisfiable set of axiom instances in the form of propositional formulas over $\mathbb{S}^*$. This MUS then offers an explanation for the unsatisfiability of this set.

The method of extracting an MUS is also used in the paper on unstructured justifications by Boixel and Endriss [BE20]. They extract an MUS which then serves as an unstructured justification. As stated earlier, we will follow their method.

Now we show the correspondence between minimally unsatisfiable subsets and unstructured justifications. Suppose we have an outcome statement $[\![\mathbf{B}, \varphi]\!] \in O$, a corpus of axioms $\mathbb{A}$ and an unstructured justification $\langle \mathcal{A}^N, \mathcal{A}^E \rangle$ for that outcome statement. Then by explanatoriness, we have that $\mathcal{A}^E \vDash_\Gamma [\![\mathbf{B}, \varphi]\!]$ while for every proper subset $\mathcal{A}' \subsetneq \mathcal{A}^E$ it holds that $\mathcal{A}' \nvDash_\Gamma [\![\mathbf{B}, \varphi]\!]$. As $[\![\mathbf{B}, \varphi]\!] \equiv \neg\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle$, every consistent aggregation rule that satisfies $\mathcal{A}^E$ also satisfies $\neg\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle$. As a consequence, there is no consistent aggregation rule that satisfies both $\mathcal{A}^E$ and $\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle$, and as for every proper subset $\mathcal{A}' \subsetneq \mathcal{A}^E$, it holds that $\mathcal{A}' \nvDash_\Gamma [\![\mathbf{B}, \varphi]\!]$, we find that any proper subset of $\mathcal{A}^E \cup \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}$ is satisfiable by some consistent aggregation rule, where we note that $\mathcal{A}^E$ is satisfiable as $\mathcal{A}^N$ is satisfiable. We present the following lemma, which is an adaptation of the lemma presented by Boixel and Endriss [BE20].

Lemma 6.1. *Let $\mathbb{A}$ be a corpus of axioms and let $\mathcal{A}^N \subseteq \mathbb{A}$ be a set of axioms that is satisfiable. Moreover, let $\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle$ be an outcome statement such that there is no consistent aggregation rule that satisfies both the axioms in $\mathcal{A}^N$ and $\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle$. Then for any set of axiom instances $\mathcal{A}$ such that $\mathcal{A} \lhd \mathcal{A}^N$, the following statements are equivalent:*

- *$\mathcal{A} \cup \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}$ is a minimally unsatisfiable subset of the set $\left(\bigcup_{\mathcal{A}' \in \mathcal{A}^N} \mathcal{A}'\right) \cup \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}$.*
- *The tuple $\langle \mathcal{A}^N, \mathcal{A} \rangle$ is an unstructured justification for the outcome statement $[\![\mathbf{B}, \varphi]\!]$.*

Proof. First suppose that $\mathcal{A} \cup \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}$ is a minimally unsatisfiable subset of the set $\left(\bigcup_{\mathcal{A}' \in \mathbb{A}} \mathcal{A}'\right) \cup \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}$. Relevance, adequacy and nontriviality follow immediately from the assumptions (see Definition 4.2).

It remains to show explanatoriness. Firstly, note that $\mathcal{A}$ is satisfiable as $\mathcal{A}^N$ is satisfiable. Consider any consistent aggregation rule that satisfies $\mathcal{A}$, then as $\mathcal{A} \cup \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}$ is unsatisfiable, we may conclude that $F \nvDash \langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle$, so $F \vDash \neg\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle$. We conclude that $F \vDash [\![\mathbf{B}, \varphi]\!]$, which gives us that $\mathcal{A} \vDash_\Gamma [\![\mathbf{B}, \varphi]\!]$.

Now consider any proper subset $\mathcal{A}'' \subsetneq \mathcal{A}$. Then it holds that $\mathcal{A}'' \cup \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\} \subsetneq \mathcal{A} \cup \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}$, and as the latter is a minimally unsatisfiable subset, we may conclude that $\mathcal{A}'' \cup \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}$ is satisfiable. From this we infer that there exists a consistent aggregation rule F such that $F \vDash \mathcal{A}'' \cup \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}$, which implies that $\mathcal{A}'' \nvDash_\Gamma \neg\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle$. We find that the explanatoriness condition is satisfied, which concludes the proof of the first direction of the equivalence.

Now suppose that $\langle \mathcal{A}^N, \mathcal{A} \rangle$ is an unstructured justification for the outcome statement $[\![\mathbf{B}, \varphi]\!]$. That means that by explanatoriness, $\mathcal{A} \vDash_\Gamma [\![\mathbf{B}, \varphi]\!]$ holds, while for every proper subset $\mathcal{A}' \subsetneq \mathcal{A}$ we have that $\mathcal{A}' \nvDash_\Gamma [\![\mathbf{B}, \varphi]\!]$. The first part immediately gives us that $\mathcal{A} \cup \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}$ is unsatisfiable.

To prove that $\mathcal{A} \cup \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}$ is minimally unsatisfiable, let $\mathcal{A}' \subsetneq \mathcal{A} \cup \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}$. We make a case distinction.

- If $\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle \notin \mathcal{A}'$, then $\mathcal{A}' \subseteq \mathcal{A}$ and as $\mathcal{A} \lhd \mathcal{A}^N$ and $\mathcal{A}^N$ is satisfiable, we may conclude that $\mathcal{A}'$ is satisfiable as well.
- If $\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle \in \mathcal{A}'$, then $(\mathcal{A}' \setminus \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}) \subsetneq \mathcal{A}$. Because of explanatoriness we then find that $(\mathcal{A}' \setminus \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}) \nvdash_\Gamma [\![\mathbf{B}, \varphi]\!]$, so there is a consistent aggregation rule F such that $F \vDash \mathcal{A}' \setminus \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}$ while $F \nvDash [\![\mathbf{B}, \varphi]\!]$, which implies that $F \vDash \langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle$. We find that $F \vDash (\mathcal{A}' \setminus \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}) \cup \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}$, so $F \vDash \mathcal{A}'$. We conclude that $\mathcal{A}'$ is satisfiable.

This concludes our proof. □

The lemma above shows us that finding such an MUS would theoretically grant us an unstructured justification, given the following two conditions.

1. We have a set of axioms $\mathcal{A}^N \subseteq \mathbb{A}$ that is satisfiable.
2. This set is unsatisfiable in combination with the negation of the presented universal outcome statement.

Both conditions we have to check through the use of a SAT solver. We observe the following.

Remark 6.1. *For a set of axioms $\mathcal{A}^N$ and the set of their encodings $\mathcal{A}^{N*}$, there exists a consistent aggregation rule satisfying $\mathcal{A}^N$ if and only if there is a truth valuation on the propositional variables in $\mathbb{S}^*$ that makes the axiom instances in the encoded axioms in $\mathcal{A}^{N*}$ and $\mathbb{A}^{Rest}$ true. This follows immediately from Lemma 5.3. This equivalence still holds if we add the negation of the presented outcome statement to both sides, so we can also check the second condition with a SAT solver.*

We use the SAT solver Glucose3 to check whether there is such a truth valuation on $\mathbb{S}^*$; see the work of Audemard and Simon for a description of the Glucose SAT solver [AS18]. This SAT solver is included in the PySAT toolkit; for a description of this toolkit we refer to the work of Igantiev, Morgado and Marques-Silva [IMM18].

6.1.1 The Algorithm for Unstructured Justifications

Before we present the algorithm, note that we have shown in Subsection 5.3.1 that the encoded framework is anonymous, although we have encoded profiles as ordered lists of ballots. For readability, we will now assume profiles in our encoding to be multisets. For the technical details, we refer to the Jupyter Notebook. Moreover, we define

$$\mathbb{A}^{Rest+} := \bigcup_{\mathcal{A} \in \mathbb{A}^{Rest}} \mathcal{A}$$

as the union of all the axioms in the set of restriction axioms $\mathbb{A}^{Rest}$ (see Definition 5.5). Note that in the following algorithm, we use the notation of the axioms instead of the formulas encoded in Section 5.3; this is because this notation offers for a better illustration of the general idea of the algorithm and following the notation of Section 5.3 would result in a more technical explanation, which we will evade for the purpose of readability.

The first if-statement in Algorithm 1 is checked with the help of a SAT solver, as described earlier. If the set is unsatisfiable, the nontriviality condition for $\mathcal{A}^N$ is not satisfied and the program will stop. For the second if-statement, if $\mathbb{A}^+$ is satisfiable (which would violate the second condition presented above), the MUS extraction tool cannot return an MUS and the algorithm will also stop.

Algorithm 1 An algorithm that extracts an MUS

input: A set of axioms $\mathcal{A}^N$ and a universal outcome statement $[\![\mathbf{B}, \varphi]\!]$
if The set $\left(\bigcup_{\mathcal{A} \in \mathcal{A}^N} \mathcal{A}\right) \cup \mathbb{A}^{Rest+}$ is satisfiable **then**
  Let $\mathbb{A}^+ := \left(\bigcup_{\mathcal{A} \in \mathcal{A}^N} \mathcal{A}\right) \cup \mathbb{A}^{Rest+} \cup \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}$
  if The set $\mathbb{A}^+$ is unsatisfiable **then**
    Extract an MUS $\mathcal{A}' \subseteq \mathbb{A}^+$
    output: The MUS $\mathcal{A}'$
  end if
end if

The MUS that is extracted in Algorithm 1 closely resembles an unstructured justification. Unfortunately, we cannot show that it is an unstructured justification or that it always grants us an unstructured justification, as this is not the case. We do find the following lemma.

Lemma 6.2. *Suppose that Algorithm 1 extracts the MUS $\mathcal{A}' \subseteq \mathbb{A}^+$. Then $\mathcal{A}' \setminus \mathbb{A}^{Rest+}$ is an unsatisfiable subset of $\left(\bigcup_{\mathcal{A} \in \mathcal{A}^N} \mathcal{A}\right) \cup \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}$ in terms of satisfiability by aggregation rules (see Definition 6.1).*

Proof. Suppose the algorithm returns the MUS $\mathcal{A}' \subseteq \mathbb{A}^+$. Then as $\left(\bigcup_{\mathcal{A} \in \mathcal{A}^N} \mathcal{A}\right) \cup \mathbb{A}^{Rest+}$ is satisfiable (as this is explicitly checked in the algorithm), we may conclude that $\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle \in \mathcal{A}'$. Now we may write $\mathcal{A}' = \mathcal{A}'_N \cup \mathcal{A}'_R \cup \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}$ with $\mathcal{A}'_N \subseteq \bigcup_{\mathcal{A} \in \mathcal{A}^N} \mathcal{A}$ and $\mathcal{A}'_R \subseteq \mathbb{A}^{Rest+}$. If we prove that $\mathcal{A}'' = \mathcal{A}'_N \cup \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}$ is an unsatisfiable subset of $\left(\bigcup_{\mathcal{A} \in \mathcal{A}^N} \mathcal{A}\right) \cup \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}$ in terms of the first definition of Definition 6.1, we are done.

Because $\mathcal{A}'$ is an MUS, there is no valuation $\mathcal{V}$ on $\mathbb{S}^*$ that satisfies $\mathcal{A}'$. But then there is no valuation $\mathcal{V}$ that satisfies the axioms in $\mathbb{A}^{Rest}$ and the instances in $\mathcal{A}'$ either. Now we may use Lemma 5.3 and conclude that there is no consistent aggregation rule F that satisfies the axiom instances in $\mathcal{A}'$. Now note that for every consistent aggregation rule F, it holds that $F \vDash \mathbb{A}^{Rest+}$, as is shown in the proof of Lemma 5.3. As a consequence, we also find that $F \vDash \mathcal{A}'_R$ for every consistent aggregation rule F. But because $\mathcal{A}'$ is unsatisfiable, we may now conclude that there is no consistent aggregation rule that satisfies $\mathcal{A}''$. □

Now we consider the explanatoriness condition in Definition 4.2. The first part of the condition requires the explanatory basis to force the outcome statement. We make the following remark.

Remark 6.2. *Because $\mathcal{A}' \setminus \mathbb{A}^{Rest+}$ is an unsatisfiable subset of $\left(\bigcup_{\mathcal{A} \in \mathcal{A}^N} \mathcal{A}\right) \cup \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}$ for any MUS $\mathcal{A}'$ that we extract, we may conclude that any consistent aggregation rule F that satisfies $\mathcal{A}' \setminus (\mathbb{A}^{Rest+} \cup \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\})$ also satisfies $\neg\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle \equiv [\![\mathbf{B}, \varphi]\!]$, so $\mathcal{A}' \setminus (\mathbb{A}^{Rest+} \cup \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}) \vDash_\Gamma [\![\mathbf{B}, \varphi]\!]$.*

The second part of the condition considers the minimality of the explanation. We note the following.

Remark 6.3. *The MUS that is outputted by Algorithm 1 cannot always be translated to an unstructured justification. This is a consequence of the fact that the minimality condition of the generated MUS states that any proper subset $\mathcal{A}$ of the MUS is satisfiable by some valuation $\mathcal{V}$ on the set $\mathbb{S}^*$. However, this valuation need not satisfy the axioms in $\mathbb{A}^{Rest}$, so we cannot use Lemma 5.3 to ensure that $\mathcal{A}$ is satisfiable by some consistent aggregation rule. The valuation $\mathcal{V}$ could specifically make the axiom instances in $\mathcal{A}$ true but not describe a consistent aggregation rule, so it could still be the case that there is no consistent aggregation rule that satisfies $\mathcal{A}$.*

We conclude that the MUS that is generated in Algorithm 1 does not perfectly align with our theoretical definition of an unstructured justification, as the minimality condition of explanatoriness

is not satisfied, as that would require every proper subset of the MUS to be satisfied by some consistent aggregation rule. In the following section, it will become clear that the output of the algorithm does closely resemble an unstructured justification, and it is often possible to actually extract an unstructured justification from this MUS.

Finally, we make the observation that our extracted MUS may contain axiom instances from $\mathbb{A}^{Rest+}$. The axiom instances from $\mathbb{A}^{Rest+}$ give us a lot of extra information about the unsatisfiability of the MUS, as will become clear in the following examples.

6.1.2 Examples of Unstructured Justifications

Now that we have provided the algorithm, it is time to present some results. We show several examples of minimally unsatisfiable subsets that are generated with the algorithm above. Before we do so, we provide two conventions regarding the notation of the axiom instances.

Notation 6.3. *Every outcome statement is denoted as an existential outcome statement or (if it is a universal outcome statement) as the negation of an existential outcome statement. Every formula is rewritten as a disjunction. Recall that* $p \to q \equiv \neg p \vee q$, $(p \wedge q) \to r \equiv \neg p \vee \neg q \vee r$ *and* $p \to (q \vee r) \equiv \neg p \vee q \vee r$.

Example 6.1. *For the first example, we let* $m = 3$, $n = 3$, $\Gamma = x_1 \vee x_2 \vee x_3$ *and*

$$\mathcal{A}^N = \{\mathcal{A}^{Fai}, \mathcal{A}^{Hom}, \mathcal{A}^{Weak\text{-}Un}, \mathcal{A}^{Strong\text{-}Un}, \mathcal{A}^{Rei}, \mathcal{A}^{Mon}\}.$$

The outcome statement we want to justify is $[\![\{(0,1,0),(1,0,0),(1,0,0)\}, x_1 \vee x_2]\!]$. *The SAT solver shows that the set* $\big(\bigcup_{\mathcal{A}\in\mathcal{A}^N} \mathcal{A}\big) \cup \mathbb{A}^{Rest+}$ *is in fact satisfiable. Consequently, the program computes the set*

$$\mathbb{A}^+ = \Big(\bigcup_{\mathcal{A}\in\mathcal{A}^N} \mathcal{A}\Big) \cup \mathbb{A}^{Rest+} \cup \{\langle\!\langle\{(0,1,0),(1,0,0),(1,0,0)\}, \neg x_1 \wedge \neg x_2\rangle\!\rangle\}$$

and extracts the following MUS (which shows us that the set $\mathbb{A}^+$ *is unsatisfiable, otherwise we would not have been able to extract an MUS).*

`Conjunction Upwards:` $\neg\langle\langle(010,100,100);\ \neg x_1 \wedge \neg x_2\rangle\rangle \vee \langle\langle(010,100,100);\ \neg x_1 \wedge \neg x_2 \wedge x_3\rangle\rangle \vee \langle\langle(010,100,100);\ \neg x_1 \wedge \neg x_2 \wedge \neg x_3\rangle\rangle$

`Conjunction Downwards:` $\neg\langle\langle(010,100,100);\ \neg x_1 \wedge \neg x_2 \wedge x_3\rangle\rangle \vee \langle\langle(010,100,100);\ x_3\rangle\rangle$

`Strong Unanimity:` $\neg\langle\langle(010,100,100);\ x_3\rangle\rangle$

`Integrity Constraint:` $\neg\langle\langle(010,100,100);\ \neg x_1 \wedge \neg x_2 \wedge \neg x_3\rangle\rangle$

`None:` $\langle\langle(010,100,100);\ \neg x_1 \wedge \neg x_2\rangle\rangle$

The final clause is the added negation of the originally inputted outcome statement. When trying to construct a human-readable proof, one might end up with the following reasoning: the final (unit) clause forces us to accept either the second or the third literal in the conjunction upwards clause. If we accept the third literal, we find a contradiction with the integrity constraint clause. If we accept the second literal, we have to accept the second literal of the conjunction downwards clause, which contradicts the strong unanimity (unit) clause. This concludes our proof.

Note here that the pair $\langle\{\mathcal{A}^{Strong\text{-}Un}\}, \{[\![\{(0,1,0),(1,0,0),(1,0,0)\}, \neg x_3]\!]\}\rangle$ *evidently serves as an unstructured justification for the statement* $[\![\{(0,1,0),(1,0,0),(1,0,0)\}, x_1 \vee x_2]\!]$.

Example 6.2. *Again, let $m = 3$, $n = 3$, $\Gamma = x_1 \vee x_2 \vee x_3$ but now add majority-preservation; so*

$$\mathcal{A}^N = \{\mathcal{A}^{Fai}, \mathcal{A}^{Hom}, \mathcal{A}^{Weak\text{-}Un}, \mathcal{A}^{Strong\text{-}Un}, \mathcal{A}^{Rei}, \mathcal{A}^{Mon}, \mathcal{A}^{Maj}\}.$$

Suppose we input the universal outcome statement $[\![\{(0,1,0),(1,0,0),(0,1,0)\}, \neg x_1]\!]$. The SAT solver shows that the set $\left(\bigcup_{\mathcal{A}\in\mathcal{A}^N} \mathcal{A}\right) \cup \mathbb{A}^{Rest+}$ is satisfiable, and the algorithm returns the following MUS.

Atomic Equivalence: $\neg\langle\langle(010);\ x_1\rangle\rangle \vee \langle\langle(010);\ x_1\rangle\rangle$
Disjunction Upwards: $\neg\langle\langle(010);\ x_1\rangle\rangle \vee \langle\langle(010);\ x_1 \vee \neg x_2 \vee x_3\rangle\rangle$
Box Implies Diamond: $\langle\langle(010);\ \neg x_1 \wedge x_2 \wedge \neg x_3\rangle\rangle \vee \langle\langle(010);\ x_1 \vee \neg x_2 \vee x_3\rangle\rangle$
Faithfulness: $\neg\langle\langle(010);\ x_1 \vee \neg x_2 \vee x_3\rangle\rangle$
Reinforcement: $\neg\langle\langle(010,100);\ \neg x_1 \wedge x_2 \wedge \neg x_3\rangle\rangle \vee \neg\langle\langle(010);\ \neg x_1 \wedge x_2 \wedge \neg x_3\rangle\rangle$
$\vee \langle\langle(010);\ x_1\rangle\rangle \vee \neg\langle\langle(010,100,010);\ x_1\rangle\rangle$
Majority-Preservation: $\langle\langle(010,100);\ \neg x_1 \wedge x_2 \wedge \neg x_3\rangle\rangle$
None: $\langle\langle(010,100,010);\ x_1\rangle\rangle$

One could again try to translate the MUS into a human-readable proof. The MUS itself is not very helpful, as it is immediately clear that, given the majority-preservation axiom, the outcome statement $[\![\{(0,1,0),(1,0,0),(0,1,0)\}, \neg x_1]\!]$ is justified, as the majority elects $\neg x_1$.

We find an unstructured justification $\langle\{\mathcal{A}^{Fai}, \mathcal{A}^{Maj}, \mathcal{A}^{Rei}\}, \{A_F, A_M, A_R\}\rangle$ for $[\![\{(0,1,0),(1,0,0),(0,1,0)\}, \neg x_1]\!]$, where the axiom instances in the justification are the instances of the respective axioms in the MUS above.

Example 6.3. *For the final example, we show that the program also runs within a reasonable amount of time for $m = 4$ (in specific scenarios). We still let $n = 3$ and $\Gamma = x_1 \vee x_2 \vee x_3$. Here we let*

$$\mathcal{A}^N = \{\mathcal{A}^{Fai}, \mathcal{A}^{Hom}, \mathcal{A}^{Weak\text{-}Un}, \mathcal{A}^{Strong\text{-}Un}, \mathcal{A}^{Mon}\}$$

and we input the statement $[\![\{(1,0,0,0),(1,0,0,0),(1,0,0,0)\}, \neg x_1 \vee \neg x_2]\!]$. Again, the SAT solver shows that the set $\left(\bigcup_{\mathcal{A}\in\mathcal{A}^N} \mathcal{A}\right) \cup \mathbb{A}^{Rest+}$ is satisfiable, and the algorithm returns the following MUS.

Atomic Equivalence: $\neg\langle\langle(1000,1000,1000);\ x_2\rangle\rangle \vee \langle\langle(1000,1000,1000);\ x_2\rangle\rangle$
Conjunction Downwards: $\neg\langle\langle(1000,1000,1000);\ x_1 \wedge x_2\rangle\rangle \vee \langle\langle(1000,1000,1000);\ x_2\rangle\rangle$
Disjunction Upwards: $\neg\langle\langle(1000);\ x_2\rangle\rangle \vee \langle\langle(1000);\ \neg x_1 \vee x_2 \vee x_3 \vee x_4\rangle\rangle$
Faithfulness: $\neg\langle\langle(1000);\ \neg x_1 \vee x_2 \vee x_3 \vee x_4\rangle\rangle$
Homogeneity: $\langle\langle(1000);\ x_2\rangle\rangle \vee \neg\langle\langle(1000,1000,1000);\ x_2\rangle\rangle$
None: $\langle\langle(1000,1000,1000);\ x_1 \wedge x_2\rangle\rangle$

We find the unstructured justification $\langle\{\mathcal{A}^{Fai}, \mathcal{A}^{Hom}\}, \{A_F, A_H\}\rangle$, where A_F is the faithfulness instance from the MUS above and A_H is the homogeneity instance.

6.2 Generating Structured Justifications

In the minimally unsatisfiable subsets that we generated in the previous section, a lot of structure is already implicitly present. As can be seen in Example 6.1, we can relatively easily construct a structured proof based on the extracted MUS. In this section, we will define an algorithm that allows us to structure such an MUS. Unfortunately, the output of this algorithm is not a structured

justification as defined in Chapter 4. However, the tree that we will generate closely resembles the tableau that is used in the definition of a structured justification.

It is roughly done as follows: following the definition of a structured justification (see Definition 4.3), the root node should contain only the negation of the outcome statement that we want to justify. This is the unit clause that is labeled with "None" in the examples in the previous section. Consequently, we gradually add child nodes containing all other unit clauses, such as axiom instances from the faithfulness axiom. After adding all of those, we need to actually start making inferences, e.g. we have the clause $\neg\langle\langle \mathbf{B}, \varphi \rangle\rangle \vee \langle\langle \mathbf{B}, \psi \rangle\rangle$ and the unit clause $\langle\langle \mathbf{B}, \varphi \rangle\rangle$ is already added in one of the nodes, so we may *apply* the clause $\neg\langle\langle \mathbf{B}, \varphi \rangle\rangle \vee \langle\langle \mathbf{B}, \psi \rangle\rangle$ and conclude that the statement $\langle\langle \mathbf{B}, \psi \rangle\rangle$ should hold. This we can do through *unit propagation*. The idea of unit propagation is that if we want to satisfy a clause of literals, and we have already rejected all the literals in the clause except for one, then we may accept that one literal, as it is the only way to still satisfy the clause. For a more elaborate description, we refer to the work of Van Harmelen, Lifschitz and Porter [HLP08].

However, this method does not account for branching within the tree. In our model, the only clauses for which we want to branch are those of the conjunction upwards and disjunction downwards axioms. For those clauses, we make a case distinction in our structured justification, i.e. we branch in the tree. This will become clear in the examples later on.

We face one more problem. Consider the following (hypothetical) MUS:

$$\{[-1], [1, 2, 3], [-2, 4], [-2, -4], [-3, 4], [-3, -4]\}.$$

It is easy to see that it is unsatisfiable as accepting either 2 or 3 will lead to a contradiction. However, suppose that $[1, 2, 3]$ is not an element of one of the branching axioms (conjunction upwards or disjunction downwards). Then we are not allowed to make this case distinction in our structured justification, so we will not be able to structure the MUS above. As a solution, we allow branching over any clause in the MUS in such scenarios.

6.2.1 The Algorithm for Structured Justifications

In this section, we explain the algorithm that is used to structure an MUS that is generated through Algorithm 1 in the previous section. The goal is to obtain a tree that serves as an alternative for a structured justification, licensed by the MUS (minus the negated input statement).

The nodes of the tree that is constructed are labeled with lists of outcome statements. Each new node that is added is labeled by the list of outcome statements of its parent plus one newly introduced outcome statement. This outcome statement is inferred either through unit propagation in combination with a clause in the MUS or through branching (also in combination with a clause in the MUS). In this way, any clause in the MUS corresponds to some expansion rule. The goal is to obtain a tree with only inconsistent leaf nodes. More specifically, for every leaf node there should be an outcome statement s such that both s and $\neg s$ are in that node. In our algorithm, we will call a tree *closed* when this condition is satisfied (instead of requiring that every leaf node contains an inconsistent outcome statement as in Chapter 3). We only add new children to consistent leaf nodes.

In Algorithm 2, we present an outline of the algorithm that is encoded in Python. First, we instantiate the tree with a root node, which is labeled with a list containing only the negation of the outcome statement that we want to justify. We then delete the unit clause that contains only

Algorithm 2 An algorithm that structures an MUS

```
input: An MUS
Instantiate the tree with the clause labeled "None"
Delete this clause from the MUS
for Clause in MUS do
    if Length of clause equals 1 then
        Add the clause to the tree
        Delete the clause from the MUS
    end if
end for
while There is a consistent leaf node do
    for Node in the set of consistent leaf nodes do
        if There is a possible clause to apply then
            Pick a random possible clause
            Expand the tree by adding one or more child nodes through the
            application of the clause (either by unit propagation or branching)
        else if There are no possible clauses to apply then
            Allow branching
        end if
    end for
end while
output: The constructed tree
```

this literal from the inputted MUS.

Consequently, we create a for-loop in which we add and delete all the unit clauses in the MUS. This creates a single branch, as any application of a unit clause consists of adding a single child with one extra literal (the single literal in the applied unit clause). These are then also deleted from the MUS.

Now we enter a while-loop which runs as long as there is a consistent leaf node; if there are no more consistent leaf nodes, every leaf node is inconsistent, so our tree is closed and we have found a tree that offers a way to structure the inputted MUS.

In the while-loop, we go through the list of consistent leaf nodes (note here that the order in which this happens is irrelevant as all these nodes are leaf nodes of separate branches that do not influence each other). For every consistent leaf node, we check the possible clauses that we can use to expand the branch ending in this leaf node, and we randomly pick one. This will append one or more new children of the original leaf node to the tree (depending on the applied clause). The algorithm will never allow for the same clause to be applied in a branch twice, so once a clause is applied in some branch, it can never appear in the set of possible clauses later in that branch.

To ensure that the algorithm always outputs a closed tree, we have to account for the case of the problematic MUS that is described in the introduction of this section. In such a scenario, it might be the case that the standard application of the clauses through unit propagation or the two cases of branching is not sufficient and our algorithm will not be able to construct a closed tree. As a solution, we allow branching, which means that we can apply any clause and branch over the different literals in that clause, adding a child node for every literal in the clause.

There are many ways to turn an MUS into a structured justification. This is dependent on the order in which the different expansion rules are applied. One could argue that the branching should happen as late as possible as it will allow the tree to be relatively small, and it will prevent multiple applications of the same rules in the different branches, as we will see in Example 6.6.

However, a small tree does not necessarily offer an explainable justification. In some cases, the most understandable justification will start with a case distinction, i.e. branching of the tree. In order to somewhat reduce the size of the generated justifications, we have decided to start by adding all the unit clauses, so that those will only be applied once. In terms of explainability, these unit clauses correspond to the set of outcome statements that we may assume (as a consequence of corresponding axioms); they do not require logical inference. For this reason, it makes sense to start by listing all these clauses. After this has been done, we do not prioritize any clause over another and allow for random application until every branch is closed.

6.2.2 Examples of Structured Justifications

Now we show several examples of the usage of Algorithm 2.

Example 6.4. *In the first example, we show what Algorithm 2 does on the input of the MUS that is extracted in Example 6.1, where we had that* $n = m = 3$ *and* $\Gamma = x_1 \vee x_2 \vee x_3$*. We repeat it one more time.*

Conjunction Upwards: $\neg\langle\langle(010,100,100);\ \neg x_1 \wedge \neg x_2\rangle\rangle \vee \langle\langle(010,100,100);\ \neg x_1 \wedge \neg x_2 \wedge x_3\rangle\rangle \vee \langle\langle(010,100,100);\ \neg x_1 \wedge \neg x_2 \wedge \neg x_3\rangle\rangle$
Conjunction Downwards: $\neg\langle\langle(010,100,100);\ \neg x_1 \wedge \neg x_2 \wedge x_3\rangle\rangle \vee \langle\langle(010,100,100);\ x_3\rangle\rangle$
Strong Unanimity: $\neg\langle\langle(010,100,100);\ x_3\rangle\rangle$
Integrity Constraint: $\neg\langle\langle(010,100,100);\ \neg x_1 \wedge \neg x_2 \wedge \neg x_3\rangle\rangle$
None: $\langle\langle(010,100,100);\ \neg x_1 \wedge \neg x_2\rangle\rangle$

In Figure 6.1, we only show the newly added outcome statement of every node for readability. First, we instantiate the tree with the (negation of) the initially inputted outcome statement that needs to be justified, i.e. the clause with the label "None". Afterwards, it is deleted from our MUS. Consequently, we add (and delete) all the unit clauses; in this case an instance of strong unanimity and the integrity constraint. After that, both the conjunction upwards and the conjunction downwards instances are applicable; the algorithm randomly picks the latter and applies it, concluding the first literal of the instance (as seen in the fourth node). Finally, the last possible clause is applied. It is easy to see that both leaf nodes contain an outcome statement and their negation. We conclude that the tree is closed.

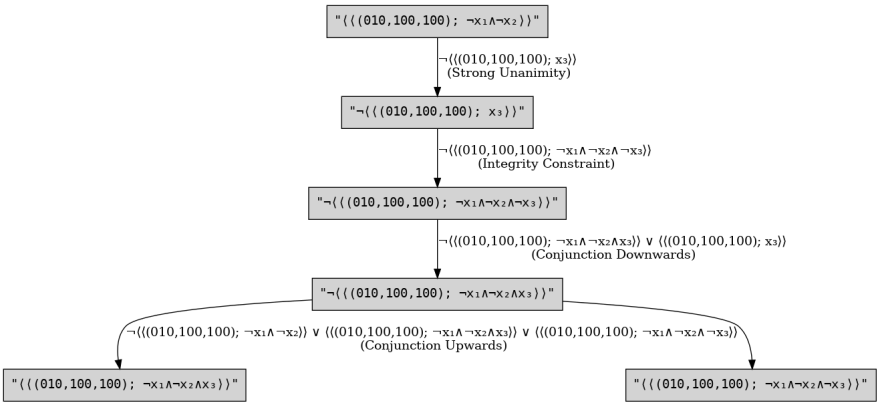

Figure 6.1: The generated tree described in Example 6.4

Example 6.5. *In the second example, let* $n = m = 3$ *and let* $\Gamma = x_1 \vee x_2 \vee x_3$ *again. Let*

$$\mathcal{A}^N = \{\mathcal{A}^{Fai}, \mathcal{A}^{Hom}, \mathcal{A}^{Weak\text{-}Un}, \mathcal{A}^{Strong\text{-}Un}, \mathcal{A}^{Rei}, \mathcal{A}^{Mon}, \mathcal{A}^{Can}\}$$

and suppose we want to justify the outcome statement $[\![\{(0,1,1),(1,0,1),(0,1,0)\}, x_2]\!]$. *First, we generate the following MUS using Algorithm 1.*

Atomic Equivalence: $\neg\langle\langle(011);\ \neg x_2\rangle\rangle \vee \langle\langle(011);\ \neg x_2\rangle\rangle$
Disjunction Upwards: $\neg\langle\langle(011);\ \neg x_2\rangle\rangle \vee \langle\langle(011);\ x_1 \vee \neg x_2 \vee \neg x_3\rangle\rangle$
Box Implies Diamond: $\langle\langle(011);\ x_1 \vee \neg x_2 \vee \neg x_3\rangle\rangle \vee \langle\langle(011);\ \neg x_1 \wedge x_2 \wedge x_3\rangle\rangle$
Faithfulness: $\neg\langle\langle(011);\ x_1 \vee \neg x_2 \vee \neg x_3\rangle\rangle$
Reinforcement: $\neg\langle\langle(011);\ \neg x_1 \wedge x_2 \wedge x_3\rangle\rangle \vee \neg\langle\langle(101,010);\ \neg x_1 \wedge x_2 \wedge x_3\rangle\rangle$
$\vee \langle\langle(011);\ \neg x_2\rangle\rangle \vee \neg\langle\langle(011,101,010);\ \neg x_2\rangle\rangle$
Cancellation: $\langle\langle(101,010);\ \neg x_1 \wedge x_2 \wedge x_3\rangle\rangle$
None: $\langle\langle(011,101,010);\ \neg x_2\rangle\rangle$

When inputting the MUS above, Algorithm 2 generates the tree depicted in Figure 6.2.

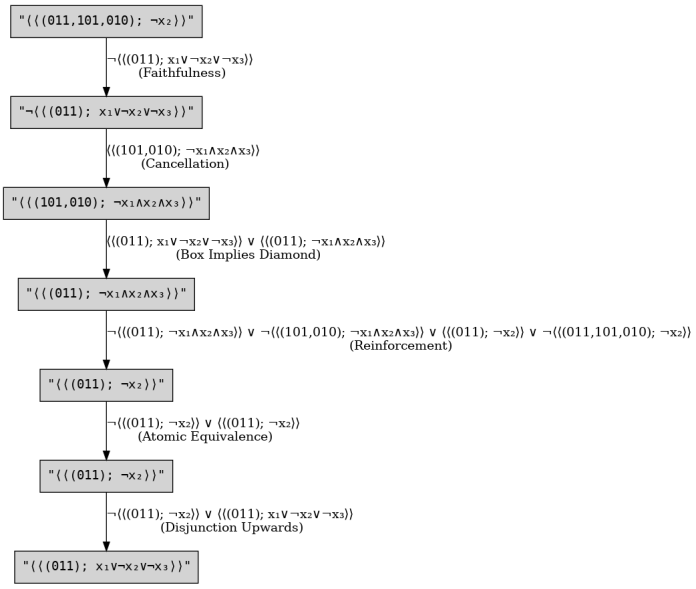

Figure 6.2: The generated tree described in Example 6.5

Example 6.6. *We show one more example of a slightly more intricate tree that branches early on. Although the justification is not so interesting, as it almost immediately follows from unanimity, the example nicely illustrates the workings of our system. We again have that* $n = m = 3$ *but let* $\Gamma = (x_1 \wedge \neg x_2 \wedge \neg x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3)$. *Let*

$$\mathcal{A}^N = \{\mathcal{A}^{Fai}, \mathcal{A}^{Hom}, \mathcal{A}^{Weak\text{-}Un}, \mathcal{A}^{Strong\text{-}Un}, \mathcal{A}^{Rei}, \mathcal{A}^{Mon}\}.$$

We want to justify the outcome statement $[\![\{(1,0,0),(1,0,0),(1,0,0)\}, x_1 \wedge \neg x_2 \wedge \neg x_3]\!]$ *and extract the following MUS.*

Disjunction Upwards: $\neg\langle\langle(100);\ \neg x_1\rangle\rangle \vee \langle\langle(100);\ \neg x_1 \vee x_2 \vee x_3\rangle\rangle$
Disjunction Upwards: $\neg\langle\langle(100);\ x_2\rangle\rangle \vee \langle\langle(100);\ \neg x_1 \vee x_2 \vee x_3\rangle\rangle$

Disjunction Upwards: $\neg\langle\langle(100);\ x_3\rangle\rangle \vee \langle\langle(100);\ \neg x_1 \vee x_2 \vee x_3\rangle\rangle$

Disjunction Downwards: $\neg\langle\langle(100,100,100);\ \neg x_1 \vee x_2 \vee x_3\rangle\rangle \vee \langle\langle(100,100,100);\ x_3\rangle\rangle$
$\vee \langle\langle(100,100,100);\ x_2\rangle\rangle \vee \langle\langle(100,100,100);\ \neg x_1\rangle\rangle$

Faithfulness: $\neg\langle\langle(100);\ \neg x_1 \vee x_2 \vee x_3\rangle\rangle$

Homogeneity: $\langle\langle(100);\ \neg x_1\rangle\rangle \vee \neg\langle\langle(100,100,100);\ \neg x_1\rangle\rangle$

Homogeneity: $\langle\langle(100);\ x_2\rangle\rangle \vee \neg\langle\langle(100,100,100);\ x_2\rangle\rangle$

Homogeneity: $\langle\langle(100);\ x_3\rangle\rangle \vee \neg\langle\langle(100,100,100);\ x_3\rangle\rangle$

None: $\langle\langle(100,100,100);\ \neg x_1 \vee x_2 \vee x_3\rangle\rangle$

Consequently, we generate the tree in Figure 6.3 using Algorithm 2.

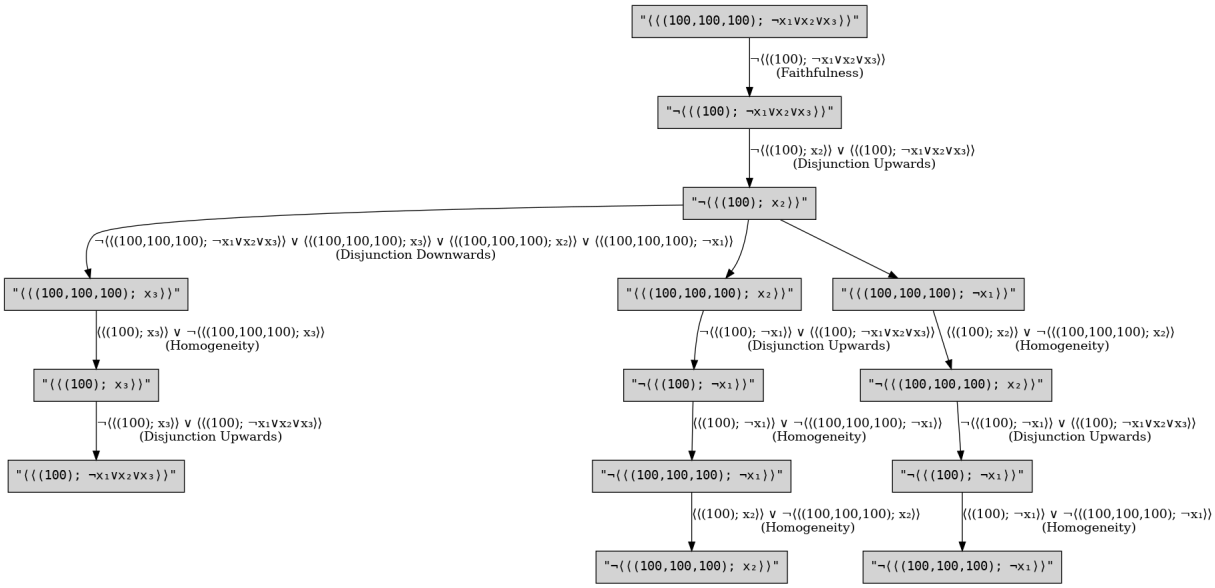

Figure 6.3: The generated tree described in Example 6.6

6.3 Termination, Soundness and Completeness for the Encoded Fragment

If we combine Algorithm 1 with Algorithm 2 by first generating an MUS and then structuring that MUS, we find an algorithm that takes as input a set of axioms $\mathcal{A}^N$ and a universal outcome statement $[\![\mathbf{B}, \varphi]\!]$ with $\mathbf{B} \in \mathrm{Mod}(\Gamma)^{+^*}$ and $\varphi \in \mathcal{L}(X)^{\wedge,\vee}$ and outputs the generated MUS and a tree that structures that MUS. This tree is rooted in $[\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle]$ and every edge corresponds to the application of some axiom instance from the MUS generated by Algorithm 1. Note here that this tree is formally not a closed tableau, but it closely resembles one: every leaf node contains a contradiction in the form of an outcome statement and its negation. We will extensively discuss the differences and similarities of the two calculi in Section 6.4.

In Algorithm 3, we show the algorithm that is described above. If Algorithm 1 does not output an MUS because one of the two preconditions is not satisfied, then Algorithm 3 will not produce any output.

As for the tableau-based calculus in the first part of this thesis, we would like to show termination, soundness and completeness of Algorithm 3.

Algorithm 3 An algorithm that concatenates Algorithm 1 and Algorithm 2

input: A set of axioms $\mathcal{A}^N$ and a universal outcome statement $[\![\mathbf{B}, \varphi]\!]$
Run Algorithm 1
if Algorithm 1 outputs an MUS **then**
    Run Algorithm 2 with as input this MUS
    output: The MUS outputted by Algorithm 1
    and the closed tree outputted by Algorithm 2
end if

Unfortunately, because our SAT solver (Glucose3) does not always terminate (as it can get stuck in an infinite loop), we cannot ensure that Algorithm 1 always terminates, as it is based on that SAT solver. As a consequence, Algorithm 3 does not always terminate. However, Algorithm 1 allows for the usage of any SAT solver. Now consider the SAT solver that simply checks for every possible valuation on $\mathbb{S}^*$ if that valuation satisfies the inputted set. Then this SAT solver always terminates, as there are only finitely many valuations, so if we use this SAT solver, or another one that always terminates, we could show termination. The same holds for the MUS extraction algorithm that is used in Algorithm 1, where we note that we could simply consider every possible subset (of which there are finitely many) and check whether it is minimally unsatisfiable.

Lemma 6.4 (Termination)**.** *Algorithm 3 always terminates, given that Algorithm 1 uses a SAT solver and a MUS extraction algorithm that always terminate.*

Proof. Firstly, note that the first part of the algorithm always terminates as both the SAT solver and the MUS extraction algorithm always terminate; the other steps in this algorithm are trivial. Consequently, we observe that Algorithm 2 always terminates as the MUS that is inputted is finite, and we can never apply the same clause twice in one branch. This concludes our proof. □

Now we turn to soundness. As discussed in Remark 6.3, the MUS that is returned by Algorithm 1 does not always grant us an unstructured justification because the minimality condition of explanatoriness is not always satisfied. This followed from a technical observation concerning the valuations in the encoding. Instead, we find the following lemma.

Lemma 6.5. *Let $[\![\mathbf{B}, \varphi]\!] \in \mathbb{S}^*$ be a universal outcome statement and let $\mathcal{A}^N$ be some set of axioms. If Algorithm 3 outputs the MUS $\mathcal{A}'$ and some tree, under the input $\mathcal{A}^N$ and $[\![\mathbf{B}, \varphi]\!]$, then it holds that $\mathcal{A}' \setminus (\mathbb{A}^{Rest+} \cup \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}) \vDash_\Gamma [\![\mathbf{B}, \varphi]\!]$.*

Proof. This follows immediately from Lemma 6.2 and Remark 6.2. □

Furthermore, note that any tree that is outputted by Algorithm 3 is closed, rooted in $[\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle]$ and contains only applications from the clauses in the MUS $\mathcal{A}' \setminus \{\langle\!\langle \mathbf{B}, \neg\varphi \rangle\!\rangle\}$. This follows immediately from the definition of Algorithm 2.

Unfortunately, because we have only encoded a fragment of $\mathcal{L}(X)$, our system is not complete. For example, we could never generate a closed tree for the outcome statement $[\![\mathbf{B}, x_1 \leftrightarrow x_2]\!]$ as we have no way of encoding this outcome statement into our framework. However, we are able to show completeness of the system with respect to the fragment of $\mathcal{L}(X)$ that we have encoded: $\mathcal{L}(X)^{\wedge,\vee}$. As for termination, we assume that the SAT solver and the MUS extraction tool terminate.

Lemma 6.6 (Completeness for the fragment $\mathcal{L}(X)^{\wedge,\vee}$)**.** *Let $s \in \mathbb{S}^*$ be a universal outcome statement and let $\mathcal{A}^N$ be a satisfiable set of axioms such that $\mathcal{A}^N$ is unsatisfiable in combination with $\neg s$.*

Then the algorithm will generate a closed tree rooted in $[\neg s]$ *with clause applications from some set* $\mathcal{A}' \subseteq \left(\bigcup_{\mathcal{A} \in \mathcal{A}^N} \mathcal{A}\right) \cup \mathbb{A}^{Rest+}$.

Proof. Because there is no consistent aggregation rule that satisfies both $\neg s$ and the axioms in $\mathcal{A}^N$ by assumption, we know by Lemma 5.3 that there is no truth valuation on $\mathbb{S}^*$ that makes $\neg s$, the axioms in $\mathcal{A}^N$ and the axioms in $\mathbb{A}^{Rest}$ true. As a consequence, the MUS extraction tool will be able to extract an MUS $\mathcal{A}' \subseteq \left(\bigcup_{\mathcal{A} \in \mathcal{A}^N} \mathcal{A}\right) \cup \mathbb{A}^{Rest+} \cup \{\neg s\}$.

Given this MUS, the second part of our algorithm will always generate a tree, where we note that in the problematic case described in the introduction of Section 6.2, our algorithm will branch over all clauses, making a nested case distinction over all clauses in the set. Then because the inputted set is an MUS, there is no possible truth assignment, so every case in the distinction will correspond to an inconsistent leaf node. □

6.4 Discussion

In the first part of this thesis, we proposed a theoretical framework in which we can capture structured justifications with the help of tableaux. We introduced a set of expansion rules that mirror the possible inference steps one could make when justifying a certain outcome statement.

In the second part of the thesis, we tried to implement this in Python. In our encoding, we decided to encode a fragment of the possible formulas $\mathcal{L}(X)$ in order to evade an exponential blow-up of the size of the outcome space. We started by constructing an algorithm that extracts an MUS that resembles an unstructured justification (extended with extra axiom instances from the valuation restriction axioms). Finally, we implemented another algorithm that structures this MUS by generating a tree data structure in Python. This tree resembles a tableau from a structured justification. Inevitably, the theory and implementation do not perfectly align. In this section, we will discuss where they coincide and where they differ.

The Extracted MUS and Soundness

Firstly, we note that the MUS that we extract in Algorithm 1 often grants us an unstructured justification, but in Remark 6.3, we show that we cannot guarantee that the MUS grants us an unstructured justification due to some technicalities in our encoding. We could however ensure the generation of an MUS that always grants us an unstructured justification by encoding the valuation restriction axioms as *hard clauses* instead of *soft clauses* in our MUS. Then every truth valuation on $\mathbb{S}^*$ has to satisfy the axioms in $\mathbb{A}^{Rest}$ and would thus correspond to a consistent aggregation rule by Lemma 5.3. This encoding would also allow us to prove soundness of the algorithm, as it would show that every MUS that we generate grants us an unstructured justification.

However, by doing so, we would no longer obtain axiom instances from the restriction axioms in our MUS. As a consequence, the extracted MUS would be less informative, as these axiom instances provide a lot of insights towards the unsatisfiability; see the examples in Subsection 6.1.2. For that reason, we have decided to encode the valuation restriction axioms as soft clauses, as the goal of this implementation was to construct an algorithm that generates explainable justifications.

Expansion Rules

In both the theoretical approach and the implementation, the main idea is to construct a tree that is rooted in the singleton containing (the negation of) some outcome statement and is licensed

by a set of axiom instances. These axiom instances constitute an important part of the inference steps that are made within the tree. The nodes are labeled with sets of outcome statements. The essential property of the constructed tree is that every branch ends in a leaf node that contains a contradiction, i.e. it contains some outcome statement and its negation.

One of the biggest differences between the theoretical model and the implementation is that they are based on different sets of expansion rules. Here it is important to note how the two sets of expansion rules came about. In the theoretical framework, we wanted a set of rules that allows for intuitive inferences while the system is sound and complete. This set of rules tries to mimic possible inferences one could make when coming up with a justification.

The second set of expansion rules were not consciously elected. Instead, they are a consequence of the axioms in Definition 5.5. This set of axioms constitutes the valuation restriction that makes sure that every truth valuation on our encoded propositional variables corresponds to a consistent aggregation rule (that satisfies exactly the outcome statements that are true in the valuation). As a consequence, these axioms provide a detailed description of the inner workings of outcome statements. As the instances of these restriction axioms are inputted in order to generate the MUS (see Section 6.1), they might also end up in the outputted MUS, which resembles an unstructured justification.

When extracting such an MUS, for example the one from Example 6.5, it is returned in the following format, where every integer represents some outcome statement in $\mathbb{S}^*$ and every sublist represents some clause, i.e. axiom instance:

$$[[-169, 15937], [-15937, 15956], [15956, 177], [-15956], [-177, -789, 169, -6649], [789], [6649]].$$

One could instantly walk through the MUS above and try to come up with a way to structure the clauses in a way that is readable and explainable. In the algorithm in Section 6.2, we presented a way to automate this process through the generation of a tree, where clauses correspond to edges in the tree. In that way, every clause becomes an inference step in the tree, which implies that the different axioms, both the valuation restrictions from Section 5.5 and the binary aggregation axioms from Section 2.3, correspond to the different possible inference steps, as is seen in the examples in Subsection 6.2.2.

There are certain similarities between the set of expansion rules from Section 3.2 and the rules induced by the axioms from Section 5.5. For example, the witness rule and the box implies diamond rule describe the same inference.

However, the two sets of rules are not equal. For example, the branching into three nodes that happens in Example 6.6 cannot be reproduced in the theoretical tableau method; one could try to mirror this step through a double application of the branching rule, but this method will always add extra outcome statements that are not added in the triple branching in Example 6.6. Conversely, the branching rule from the set of expansion rules in the theoretical part can not always be mirrored with the set of expansion rules in the implementation.

The Fragment $\mathcal{L}(X)^{\wedge,\vee}$ and Completeness

Apart from the possible inference rules, a big difference between the two frameworks is that our encoding only expresses a fragment of the outcome statements that can be expressed in the theoretical framework. Because of this, the algorithm we have encoded is not complete. However, the following lemma shows us that we can still express every outcome statement with the fragment

$\mathcal{L}(X)^{\wedge,\vee}$.

Lemma 6.7. *Any outcome statement of the form* $\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle$ *or* $[\![\mathbf{B}, \varphi]\!]$ *with* $\mathbf{B} \in Mod(\Gamma)^+$ *and* $\varphi \in \mathcal{L}(X)$ *can be written as either a conjunction or a disjunction of outcome statements of the form* $\langle\!\langle \mathbf{B}, \psi \rangle\!\rangle$ *or* $[\![\mathbf{B}, \psi]\!]$ *with* $\mathbf{B} \in Mod(\Gamma)^+$ *and* $\psi \in \mathcal{L}(X)^{\wedge,\vee}$.

Proof. Consider an outcome statement $\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle$ with $\mathbf{B} \in \mathrm{Mod}(\Gamma)^+$ and $\varphi \in \mathcal{L}(X)$. The formula φ can be rewritten as a DNF-formula (see Section 1.1) of the form $\psi_1 \vee \cdots \vee \psi_k$, where every ψ_i is a conjunction of literals in X^*. But now we find that

$$\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle \equiv \langle\!\langle \mathbf{B}, \psi_1 \vee \cdots \vee \psi_k \rangle\!\rangle \equiv \langle\!\langle \mathbf{B}, \psi_1 \rangle\!\rangle \vee \cdots \vee \langle\!\langle \mathbf{B}, \psi_k \rangle\!\rangle.$$

For the universal outcome statements, we use the fact that any formula in $\mathcal{L}(X)$ can be rewritten as a CNF-formula and then pull this conjunction out, analogously to the proof above. □

Given the lemma above, one could try to extend the encoded framework in order to obtain a method that is complete. In order to do so, we would need to translate any outcome statement that appears in the input or in one of the axiom instances to either a conjunction or a disjunction of outcome statements from our fragment. With that encoding at hand, the framework is in principle expressive enough to generate a justification whenever one could theoretically find one.

We note that we can only justify outcome statements of the form $[\![\mathbf{B}, \varphi]\!]$ with $\varphi \in \mathcal{L}(X)^{\wedge,\vee}$. Because of that, we cannot justify statements of the form $[\![\mathbf{B}, \varphi_{F(\mathbf{B})}]\!]$ (see Definition 2.13), where $F(\mathbf{B})$ describes some outcome set. However, with the help of Lemma 6.7, it would be possible to extend our model and allow for justifications of this type.

We also make the observation that our algorithm does not offer a closed tree in the sense that every leaf node contains an inconsistent outcome statement, as we have not encoded these because $\bot \notin \mathcal{L}(X)^{\wedge,\vee}$. However, this is just a small technicality, because we do require overtly inconsistent leaf nodes: every leaf node should contain some outcome statement and its negation. One could easily extend our model by introducing an inconsistent outcome statement. As it does not really contribute to the model (since we already have clear inconsistencies in the leaf nodes), we have decided to leave it out.

Finally, we note that in Section 5.3, we have encoded the axioms for the fragment of $\mathcal{L}(X)^{\wedge,\vee}$ and we showed that they are equivalent to the axioms in Section 1.4. However, this does restrict the explainability of our model to a certain extent. Observe here that our algorithm will never return axiom instances with outcome statements of the form $\langle\!\langle \mathbf{B}, \varphi \rangle\!\rangle$ or $[\![\mathbf{B}, \varphi]\!]$ with $\varphi \in \mathcal{L}(X) \setminus \mathcal{L}(X)^{\wedge,\vee}$, as these are not encoded. This type of axiom instance could occur in the theoretical framework.

Running Time

To conclude this discussion, we make some remarks concerning the running time of the two algorithms. There are four steps that require attention here.

1. The instantiation of both the restriction axioms and the axioms in $\mathcal{A}^N$.
2. Checking the satisfiability of the conjunction of all the above with a SAT solver; this is the nontriviality condition.
3. Extracting the MUS; this is the essential part of the first algorithm.
4. Turning the MUS into a structured justification; this is done in the second algorithm.

For small examples, with $n = m = 3$, many of the axioms in the first step are instantiated within a matter of seconds. Reinforcement runs the longest; it will take about a minute to instantiate the axiom.

After all the axioms have been instantiated, we check the second and third step. Both happen in a matter of seconds for $n = m = 3$. The fourth step also runs within seconds.

Generally, we may conclude that step 4. will always be relatively quick, as the input is an extracted MUS. Most of the times, such an MUS is substantially smaller than the union of all the axiom instances in the second step.

For bigger n and m the running time grows quickly. For $n = 3$ and $m = 4$, the instantiating of the axioms already takes about an hour. However, the satisfiability of the conjunction of the different formulas is checked within a minute. Note that we would only need to instantiate the axioms once. Thereafter, we can run different examples, for the chosen n, m and Γ. Unfortunately, the *Python 3 Google Compute Engine Backend* on which we run the programs only offers 12 GB of RAM and the extraction of our MUS requires more for this setting. As a consequence, our system crashes and we are unable to extract an MUS. For $n = 4$ and $m = 3$, the instantiating of the axioms also takes around an hour. But here, we are allowed to extract an MUS without exceeding the 12 GB of RAM; the extraction now takes around a minute. Step 2. and 4. both run within several seconds. For $n = 5$, the instantiation in the first step was not finished after three hours.

It makes sense that increasing m leads to a higher running time than increasing n. This is due to the fact that increasing m leads to an increase in both the number of profiles and the number of possible formulas in the outcome statements. Increasing n only increases the number of profiles.

The integrity constraint restricts the set of possible profiles. If the constraint is "strict", i.e. the set of allowed profiles is small, then the algorithm will run quicker, as it needs to check less profiles and thus less axiom instances. In that sense, the integrity constraint also has a lot of influence on the running time of the algorithm. For example, if we let $n = 3$ and $m = 4$ but we let $\Gamma = (x_1 \leftrightarrow x_2) \wedge (x_3 \leftrightarrow x_4)$, then the instantiation of the CNF-formulas takes only a minute.

Finally, we note that the set of axioms that is instantiated in the first step is also relevant for the running time. For example, if we let $n = 3$, $m = 4$ and $\Gamma = x_1 \vee x_2 \vee x_3$ but we do not instantiate the reinforcement axiom, the instantiation only takes about 20 minutes and we are able to run the rest of the steps within a matter of minutes; see Example 6.3.

Conclusion

In this thesis, we defined unstructured and structured justifications for binary aggregation. We also encoded the framework of binary aggregation in Python and constructed an algorithm that extracts an MUS that resembles an unstructured justification (extended with extra axiom instances from the valuation restriction axioms) and an algorithm that offers a way of structuring this MUS by constructing a tree that resembles a tableau. First, we summarize the most important findings per chapter.

In **Chapter 2**, we first introduced outcome statements. Thereafter, we proposed a definition for axioms and axiom instances by expressing them in the language of outcome statements $\mathcal{L}(O)$. Finally, we showed the expressive power of $\mathcal{L}(O)$ by rewriting the axioms of binary aggregation in this language. We also proved that these rewritten axioms are equivalent to the original axiom definitions.

In **Chapter 3**, we defined a tableau-based calculus that allows us to prove the unsatisfiability of sets of outcome statements and axiom instances in a structured manner. The calculus is based on the set of expansion rules that we introduced. These correspond to the different inferences one would make when arguing about aggregation rules. Finally, we proved termination, soundness and completeness of this calculus.

In **Chapter 4**, we introduced the main concepts of the thesis: unstructured and structured justifications for binary aggregation. This chapter concludes the theoretical part of the thesis.

In **Chapter 5**, we showed how we encoded the framework of binary aggregation in Python. An important encoding was that of aggregation rules, which required the introduction and encoding of the valuation restriction axioms. We concluded the chapter by showing how to encode the binary aggregation axioms in Python.

In **Chapter 6**, we introduced two algorithms. The first algorithm takes as input an outcome statement and a set of axioms and generates an MUS that resembles an unstructured justification as output. The second algorithm takes as input such an MUS and then outputs a tree that serves as an alternative for the tableau in a structured justification. If we combine the two algorithms by running them after each other, then we find an algorithm that generates an MUS together with a tree that offers a way to structure the MUS. At the end of the chapter, we discuss termination, soundness and completeness of the algorithm. Finally, we show that for small n and m ($n = m = 3$), we are able to generate justifications quickly. For bigger n or m, the running time quickly grows and it is hard and sometimes impossible to generate justifications for these bigger examples.

Evaluation

The first goal of this thesis was to set up a theoretical model in which we can express and construct justifications. In the first part of this thesis, we have presented a model that fulfills this role. The

three main components are the language of outcome statements, the tableau-based calculus and the definitions of justifications.

As a consequence of the fact that the size of the outcome space for an aggregation rule is double exponential (See Section 2.1), we decided to define outcome statements that describe the outcome set rather than state the possible outcome sets. An easy way to describe a ballot is through the use of formulas over the issue-variables. To describe the outcome set, we decided to introduce an existential and a universal statement. We have then defined a propositional language over these outcome statements. We found that this language offers an intuitive way to formally argue about aggregation rules. Moreover, we showed its expressive power by rewriting the binary aggregation axioms in this language.

The semantics of these outcome statements lies at the core of the tableau-based calculus that we have defined. This tableau method inherits the intuitive nature of the outcome statements and allows for explainable proofs. We also showed termination, soundness and completeness for the calculus, although it required a lot of technical details, and we had to carefully pick the set of expansion rules and their different preconditions.

We concluded the theoretical part by defining unstructured and structured justifications and providing several examples for both definitions. In these examples, it becomes clear that it is relatively hard to come up with good examples in terms of binary aggregation axioms; a lot of the examples come down to simply justifying that the majority outcome should be elected. However, sometimes it might still be valuable to justify such an outcome without explicitly applying the majority-preservation axiom, as this axiom is rather strong and is not collectively accepted in every collective decision-making scenario. For example, consider the situation where 51 people cast the same ballot while 49 people cast the complete opposite ballot; in that scenario, it could be deemed unfair to simply go with the majority. For an interesting discussion on *egalitarian principles* that offer an alternative approach within the related framework of judgment aggregation, we refer to the work of Botan, De Haan, Slavkovik and Terzoupoulou [Bot+23].

Moreover, in certain situations we notice that our set of binary aggregation axioms falls short. To illustrate this, we consider the example below, which is a version of the doctrinal paradox for binary aggregation; see the paper by Kornhauser and Sager [KS93].

Let $m = n = 3$, suppose that $\Gamma = \neg(x_1 \wedge x_2 \wedge x_3)$ and consider the profile displayed below.

$$\begin{array}{c} 1 : (1,1,0) \\ 1 : (1,0,1) \\ 1 : (0,1,1) \end{array}$$

If we were to follow the majority on every issue, which might seem fair to do in this situation, we find an inconsistent outcome vector, namely $(1,1,1)$. We conclude that the profile is not majority-consistent. The other axioms do not really offer any help in a symmetrical situation like the one above. However, one could argue that it should be justifiable to at least pick one of the issues, as every voter accepts two of the three issues. A possible solution is the introduction of *distance-based axioms* that parallel the Kemeny and Slater rules from judgment aggregation; see the work of Grossi and Pigozzi [GP22]. These rules look for the consistent outcomes that are the most similar to the ballots in the profile. Such a distance-based axiom could offer help in a situation like the above, and allow us to justify that we should at least accept one issue.

However, the examples presented in Chapter 4 do show that the tableau method that we defined offers for simple and intuitive visual proofs.

The second goal of this thesis was to encode binary aggregation in a computer and develop an algorithm that generates justifications for binary aggregation. Unfortunately, we have only encoded a fragment of the formulas $\mathcal{L}(X)$, namely $\mathcal{L}(X)^{\wedge,\vee}$. As a consequence, the two-part algorithm presented in Chapter 6 is not complete, i.e. we cannot justify every outcome statement that we could theoretically justify. Moreover, our implementation does not coincide perfectly with the theory. We have already extensively discussed this matter in Section 6.4.

We did manage to construct an algorithm in Python that generates a tree that resembles a structured justification. For small examples, like for $n = m = 3$ (and any integrity constraint Γ), it runs within a minute. For bigger m, the running time quickly grows and it becomes very hard (if not practically impossible) to run the algorithm, as is seen in Section 6.4.

However, if we were to have a very restrictive integrity constraint that substantially narrows down the set of allowed profiles, we are able to run the program for bigger n and m.

We have met the goal of developing an algorithm that generates justifications, although it only works efficiently for small examples and it does not align perfectly with the theory presented in the first part of this thesis.

Future Work

We will end the conclusion by proposing several directions for possible future work.

The first direction we consider is researching complexity results for the justifications defined in this thesis. Boixel and De Haan [BH21] show the computational complexity of two things: firstly, whether an unstructured justification (for voting theory) is correct, and secondly, whether there exists an unstructured justification (for voting theory) in a given setting. As these results would give insights to the applicability and limits of the algorithms introduced in this thesis, it would be relevant to do research in this direction.

We have chosen to adapt the justification definitions from voting theory to binary aggregation (with integrity constraints). Binary aggregation is both a general framework and lends itself for a relatively easy encoding in a computer (as the name suggests). However, there are many frameworks closely related to binary aggregation, such as judgment aggregation, among others. Endriss, Grandi, De Haan and Lang [End+16] give an overview of a few of these frameworks and show that all are equally expressive. However, binary aggregation (with integrity constraints) is less general than some of these other frameworks in terms of *succinctness*; this notion describes how compactly one could translate a setting from one framework to another. By allowing for extra variables (other than the ones in X) in the constraint, the framework would become more general in these terms. Considering the different translations between these frameworks and their generality based on the paper by Endriss et al. [End+16] would be another interesting research direction, as it could possibly lead to an efficient adaption of the justification definitions to these other frameworks.

As discussed in the evaluation above, the binary aggregation axioms may sometimes fall short. This is due to the fact that binary aggregation is a relatively new field of research and the axiomatic method has not yet been deployed extensively for this framework. Together with the possible translations proposed in the paragraph above, it might be of interest to adapt the model to judgment aggregation and see whether we can use the judgment aggregation axioms to find more profound justifications.

In this thesis, we have introduced several tools in the framework of binary aggregation for the purpose of expressing and constructing justifications. But these tools could also be used for different purposes. For example, the tableau-based calculus that we have defined serves as a method to prove the unsatisfiability of a set of axioms together with a set of outcome statements. But if we disregard this set of outcome statements, our calculus offers a structured way of showing impossibility results: if we construct a closed tableau that is licensed by some set of axiom instances, then we can conclude an impossibility of the corresponding axioms.

Moreover, the SAT solver that we use within our encoding can also be used to apply a method like that of Geist and Peters [GP17] to binary aggregation. As SAT solvers have been very helpful tools in the axiomatic method for voting theory and related frameworks [LT09; GE11a; BG16; GP17; End20; Klu+20], it might be interesting to see what they could offer in the research on binary aggregation.

In Section 6.4, we have extensively discussed the differences between our theoretical model and the implementation thereof. Specifically, we have looked at the difference between the set of expansion rules defined in the tableau-based calculus and the set of expansion rules that was induced by the restriction axioms in the implementation. Both these sets are carefully chosen to be both intuitive and fit the technical requirements. However, neither are unique and there are many different ways to substitute these sets. We consider two possible options. Firstly, we could alter the set of expansion rules to match the implementation, i.e. define a theoretical model that aligns better with what we have implemented. The second option is to go the other way around and try to define another set of restriction axioms that aligns more with the set of expansion rules that we have defined.

Finally, one could try to formalize the sketch for the proof of completeness in Section 6.4, which would bring together the theoretical part and the implementation.

The algorithm that we have developed is not perfect, which means it could be improved. In the work of Nardi, Boixel and Endriss [NBE22], they show an algorithm that efficiently computes unstructured justifications for voting theory. One could try to adapt this algorithm to our model and use that instead of the deletion-based MUS extraction algorithm that we use.

In our second algorithm, we randomly pick one of the possible expansion rules for a leaf node. Instead, one could try to implement certain heuristics in order to generate more explainable justifications. For example, one could try to rank the different applicable clauses based on their relevance at that point in the tableau.

Finally, we note that participatory budgeting is a closely related framework, which would allow for more applied examples of our model. One could try to adapt our model to the framework of participatory budgeting and consider the model in a more real-world setting.

Bibliography

[Arr63] Kenneth J. Arrow. *Social Choice and Individual Values*. 2nd. First edition published in 1951. New York: John Wiley and Sons, 1963 (cited on pages 4, 54).

[AS18] Gilles Audemard and Laurent Simon. "On the glucose SAT solver". In: *International Journal on Artificial Intelligence Tools* 27.01 (2018) (cited on page 74).

[ASS02] Kenneth J. Arrow, Amartya Sen, and Kotaro Suzumura, editors. *Handbook of Social Choice and Welfare*. Volume 1. Elsevier, 2002 (cited on page 3).

[BE20] Arthur Boixel and Ulle Endriss. "Automated Justification of Collective Decisions via Constraint Solving". In: *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2020)*. IFAAMAS, May 2020 (cited on pages 6, 18, 44, 48, 54, 72, 73).

[BEH22] Arthur Boixel, Ulle Endriss, and Ronald de Haan. "A Calculus for Computing Structured Justifications for Election Outcomes". In: *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI-2022)*. Feb. 2022. `https://doi.org/10.1609/aaai.v36i5.20414` (cited on pages 6, 15, 25–28, 33–35, 38, 41, 43, 44, 46, 49, 50, 54).

[BEN22] Arthur Boixel, Ulle Endriss, and Oliviero Nardi. "Displaying Justifications for Collective Decisions". In: *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI-2022)*. Demo Paper. July 2022. `https://doi.org/10.24963/ijcai.2022/847` (cited on page 6).

[BG16] Felix Brandt and Christan Geist. "Finding Strategyproof Social Choice Functions via SAT Solving". In: *Journal of Artificial Intelligence Research* 55 (2016), pages 565–602 (cited on pages 4, 91).

[BH21] Arthur Boixel and Ronald de Haan. "On the complexity of finding justifications for collective decisions". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Volume 35. 6. 2021, pages 5194–5201 (cited on page 90).

[Bie+09] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*. Volume 185. IOS Press, 2009 (cited on pages 4, 72).

[Bot+23] Sirin Botan, Ronald de Haan, Marija Slavkovik, and Zoi Terzopoulou. "Egalitarian judgment aggregation". In: *Autonomous Agents and Multi-Agent Systems* 37 (Feb. 2023) (cited on page 89).

[Bra+16] Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D. Procaccia, editors. *Handbook of Computational Social Choice*. Cambridge University Press, 2016 (cited on pages 4, 54).

[CE16] Olivier Cailloux and Ulle Endriss. "Arguing about Voting Rules". In: *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2016)*. Also presented at COMSOC-2016. IFAAMAS, May 2016 (cited on page 6).

[DAg+13] Marcello D'Agostino, Dov M. Gabbay, Reiner Hähnle, and Joachim Posegga, editors. *Handbook of tableau methods*. Springer Science & Business Media, 2013 (cited on page 27).

[DAg13] Marcello D'Agostino. "Tableau methods for classical propositional logic". In: *Handbook of tableau methods*. Edited by Marcello D'Agostino, Dov M. Gabbay, Reiner Hähnle, and Joachim Posegga. Springer Science & Business Media, 2013. Chapter 2 (cited on page 27).

[DH10] Elad Dokow and Ron Holzman. "Aggregation of binary evaluations". In: *Journal of Economic Theory* 145.2 (2010), pages 495–511 (cited on page 5).

[EG14] Ulle Endriss and Umberto Grandi. "Binary Aggregation by Selection of the Most Representative Voter". In: *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI-2014)*. July 2014 (cited on page 5).

[End+16] Ulle Endriss, Umberto Grandi, Ronald de Haan, and Jérôme Lang. "Succinctness of Languages for Judgment Aggregation". In: *Proceedings of the 15th International Conference on Principles of Knowledge Representation and Reasoning (KR-2016)*. Apr. 2016 (cited on page 90).

[End16] Ulle Endriss. "Judgment Aggregation". In: *Handbook of Computational Social Choice*. Edited by F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. D. Procaccia. Cambridge University Press, 2016. Chapter 17 (cited on page 5).

[End20] Ulle Endriss. "Analysis of One-to-One Matching Mechanisms via SAT Solving: Impossibilities for Universal Axioms". In: *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI-2020)*. AAAI Press, Feb. 2020. `https://doi.org/10.1609/aaai.v34i03.5689` (cited on pages 4, 91).

[End23] Ulle Endriss. *SAT Solving for Social Choice Theory*. 2023. `https://nbviewer.org/urls/staff.science.uva.nl/u.endriss/teaching/comsoc/2024/code/sct.ipynb` (visited on 07/01/2025) (cited on pages 56, 57, 59, 60, 62, 67, 68, 71).

[GE10] Umberto Grandi and Ulle Endriss. "Lifting Rationality Assumptions in Binary Aggregation". In: *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-2010)*. July 2010. `https://doi.org/10.1609/aaai.v24i1.7603` (cited on page 5).

[GE11a] Christian Geist and Ulle Endriss. "Automated search for impossibility theorems in social choice theory: Ranking sets of objects". In: *Journal of Artificial Intelligence Research* 40 (2011), pages 143–174. `https://doi.org/10.1613/jair.3126` (cited on pages 4, 91).

[GE11b] Umberto Grandi and Ulle Endriss. "Binary Aggregation with Integrity Constraints". In: *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI-2011)*. July 2011 (cited on pages 5, 10).

[GE13] Umberto Grandi and Ulle Endriss. “First-Order Logic Formalisation of Impossibility Theorems in Preference Aggregation”. In: *Journal of Philosophical Logic* 42.4 (2013), pages 595–618. https://doi.org/10.1007/s10992-012-9240-8 (cited on pages 6, 18).

[GP17] Christian Geist and Dominik Peters. “Computer-aided methods for social choice theory”. In: *Trends in Computational Social Choice* (2017), pages 249–267 (cited on pages 4, 56, 60, 67, 71, 91).

[GP22] Davide Grossi and Gabriella Pigozzi. *Judgment aggregation: a primer*. Springer Nature, 2022 (cited on pages 11, 54, 89).

[Gra12a] Umberto Grandi. *Binary aggregation with integrity constraints*. University of Amsterdam, 2012 (cited on page 5).

[Gra12b] Umberto Grandi. “Binary aggregation with integrity constraints”. In: University of Amsterdam, 2012. Chapter 2, pages 15–27 (cited on page 10).

[HLP08] Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter. *Handbook of knowledge representation*. Elsevier, 2008 (cited on page 78).

[How05] Colin Howson. *Logic with trees: an introduction to symbolic logic*. Routledge, 2005 (cited on page 9).

[IMM18] Alexey Ignatiev, Antonio Morgado, and Joao Marques-Silva. “PySAT: A Python toolkit for prototyping with SAT oracles”. In: *International Conference on Theory and Applications of Satisfiability Testing*. Springer. 2018, pages 428–437 (cited on page 74).

[Jon25] Otto de Jong. *Supplementary material for the thesis "Structured Justifications for Binary Aggregation"*. July 2025. https://doi.org/10.5281/zenodo.16614542. https://doi.org/10.5281/zenodo.16614542 (cited on pages 56, 67).

[Klu+20] Boas Kluiving, Adriaan de Vries, Pepijn Vrijbergen, Arthur Boixel, and Ulle Endriss. “Analysing Irresolute Multiwinner Voting Rules with Approval Ballots via SAT Solving”. In: *Proceedings of the 24th European Conference on Artificial Intelligence (ECAI 2020)*. IOS Press, 2020, pages 131–138 (cited on pages 4, 91).

[KS93] Lewis A. Kornhauser and Lawrence G. Sager. “The one and the many: Adjudication in collegial courts”. In: *California Law Review* 81 (1993) (cited on pages 14, 89).

[Lan+17] Jérôme Lang, Gabriella Pigozzi, Marija Slavkovik, Leendert van der Torre, and Srdjan Vesic. “A partial taxonomy of judgment aggregation rules and their properties”. In: *Social Choice and Welfare* 48 (2017), pages 327–356 (cited on pages 11, 54).

[LP02] Christian List and Philip Pettit. “Aggregating Sets of Judgments: An Impossibility Result”. In: *Economics and Philosophy* 18.1 (2002), pages 89–110. https://doi.org/10.1017/S0266267102001064 (cited on page 5).

[LT09] Fangzhen Lin and Pingzhong Tang. “Computer-aided proofs of Arrow’s and other impossibility theorems”. In: *Artificial Intelligence* 173.11 (2009), pages 1041–1053 (cited on pages 4, 91).

[Mar10] Joao Marques-Silva. “Minimal unsatisfiability: Models, algorithms and applications”. In: *2010 40th IEEE International Symposium on Multiple-Valued Logic*. IEEE. 2010, pages 9–14 (cited on page 73).

[Men09] Elliott Mendelson. *Introduction to mathematical logic*. Chapman and Hall/CRC, 2009 (cited on page 9).

[Nar21] Oliviero Nardi. "A graph-based algorithm for the automated justification of collective decisions". In: *Master's thesis. ILLC, University of Amsterdam* (2021) (cited on page 6).

[NBE22] Oliviero Nardi, Arthur Boixel, and Ulle Endriss. "A Graph-Based Algorithm for the Automated Justification of Collective Decisions". In: *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2022)*. IFAAMAS, May 2022 (cited on pages 6, 91).

[REH20] Simon Rey, Ulle Endriss, and Ronald de Haan. "Designing Participatory Budgeting Mechanisms Grounded in Judgment Aggregation". In: *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR-2020)*. Sept. 2020, pages 692–702. `https://doi.org/10.24963/kr.2020/71` (cited on page 5).

[Rey23] Simon Rey. *Variations on Participatory Budgeting*. University of Amsterdam, 2023 (cited on page 5).

[Tut01] William T. Tutte. *Graph theory*. Volume 21. Cambridge university press, 2001 (cited on page 10).

[Wil75] Robert Wilson. "On the theory of aggregation". In: *Journal of Economic Theory* 10.1 (1975), pages 89–99 (cited on page 5).

[Zwi16] William S. Zwicker. "Introduction to the Theory of Voting". In: *Handbook of Computational Social Choice*. Edited by F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. D. Procaccia. Cambridge University Press, 2016. Chapter 2 (cited on page 4).