Judgment aggregation in dynamic logic of propositional assignments

Arianna Novaro, Umberto Grandi, Andreas Herzig

Judgment Aggregation in Dynamic Logic of Propositional Assignments

Arianna Novaro*[1], Umberto Grandi†[1], and Andreas Herzig‡[2]

[1]IRIT, University of Toulouse
[2]IRIT, CNRS, University of Toulouse

Abstract

Judgment aggregation models a group of agents having to collectively decide over a number of logically interconnected issues starting from their individual opinions. In recent years, a growing literature has focused on the design of logical systems for social choice theory, and for judgment aggregation in particular, making use of logical languages designed *ad hoc* for this purpose. In this paper we deploy the existing formalism of Dynamic Logic of Propositional Assignments (DL-PA), an instance of Propositional Dynamic Logic where atomic programs affect propositional valuations. We show that DL-PA is a well-suited formalism for modeling the aggregation of binary judgments from multiple agents, by providing logical equivalences in DL-PA for some of the best known aggregation procedures, desirable axioms coming from the literature on judgment aggregation, and properties for the safety of the agenda problem.

Keywords: Dynamic Logic; Modal Logic; Social Choice Theory; Computational Social Choice; Automated Reasoning

1 Introduction

Social choice theory gathers mathematical models for the study of collective decision making, such as voting and elections, or the allocation of resources among a group of agents (Arrow et al., 2002). Judgment aggregation is one such model where the individual opinions expressed by the agents over a set of correlated issues are aggregated into a collective choice by means of an aggregation rule. The classical example that initiated this research field is known in the literature as the *discursive dilemma* (List and Pettit, 2002): three agents have to decide over three issues that are logically correlated, and though all agents express consistent views, issue-wise majority voting can lead to inconsistent outcomes (cf. Example 1). The study of the aggregation of binary judgments can be traced back to work by legal scholars (Kornhauser and Sager, 1993) and is now an established framework in artificial intelligence to handle complex collective decisions (Endriss, 2016; Grossi and Pigozzi, 2014).

The evident link between judgment aggregation and (propositional) logic, together with

*arianna.novaro@irit.fr
†umberto.grandi@irit.fr
‡andreas.herzig@irit.fr

the modularity of many judgment aggregation results looking for incompatible combinations of axiomatic properties, inspired researchers to investigate logical formalizations of this framework. Notable examples include the work of Pauly (2007), who formulated judgment aggregation in a minimal logical language to refer to the outcomes of aggregation rules, and of Ågotnes et al. (2011), who designed a Judgment Aggregation Logic with a Hilbert-style axiomatization, later expanded by a natural deduction system proposed by Perkov (2016). Similar formalizations had already been proposed for social choice functions — the main model in social choice theory, where a set of individual preferences represented by linear orders over a set of alternatives has to be aggregated into a collective preference. To give some examples, Arrow's theorem, the cornerstone result of social choice theory, has been expressed in higher-order logics (Wiedijk, 2007; Nipkow, 2009), first-order logic (Grandi and Endriss, 2013) and modal logic (Ciná and Endriss, 2015), while a modal logic for strategic preference aggregation was proposed by Troquard et al. (2011).

The final aim of formalizing aggregation models in a suitable logical language is the formal verification of properties and theoretical results already obtained in the literature and, ultimately, the automated discovery of new theorems. This can be viewed as a sort of 'Hilbert program' where concepts and results that were previously established in an informal language are recast in formal logic. This approach has recently been proven very successful in preference aggregation and voting, thanks to the combination of mathematical lemmas and automated reasoning techniques. In preference aggregation, the seminal work of Tang and Lin (2009) obtained a semi-automated proof of Arrow's theorem by combining two inductive lemmas with SAT-solving. Geist and Endriss (2011) later brought this approach to the level of theorem discovery in the field of ranking sets of objects, automatically testing all combinations of 20 axiomatic properties via SAT-solving, which combined with a general inductive lemma gave rise to 84 impossibility theorems (among which many non-trivial ones). In recent years, a number of open problems in classical social choice theory has then been tackled and solved (Brandl et al., 2016; Brandt and Geist, 2016; Brandt et al., 2017) using a variety of techniques in automated reasoning from SAT-solvers, satisfiability modulo theory (SMT-solvers), and minimal unsatisfiable subset extraction (MUS).

Despite its success, the use of automated reasoning techniques in social choice theory often requires proving hard mathematical lemmas, thus confining its use to specialists. A more friendly and flexible tool is of need, with human-readable formulas that can be easier to understand and manipulate in search for new results and new applications. However, each high-level formalization of judgment aggregation that has been proposed is based on a new logical language, making the use of automated reasoning techniques less immediate. In this paper we aim at attaining both goals by employing a high-level logical language that is also amenable to automated reasoning through the use of a dedicated prover, or possibly through a translation into propositional logic.

For this reason, the logical formalism of our choice is Dynamic Logic of Propositional Assignments or DL-PA (van Eijck, 2000; Balbiani et al., 2013), which is an instance of Propositional Dynamic Logic (Pratt, 1976; Fischer and Ladner, 1979) where atomic programs assign truth values to propositional variables. An existing literature in the fields of knowledge representation and multi-agent systems has proven DL-PA to be a 'unifying language' to express a variety of

frameworks, from belief change operations (Herzig, 2014) and abstract argumentation (Doutre et al., 2014), to interaction in normative systems (Herzig et al., 2011) and social simulations (Gaudou et al., 2011). The present paper aims to add another important setting from the area of computational social choice to the previous list of successful results: i.e., judgment aggregation. Crucially, DL-PA is grounded on propositional logic, meaning that there exists a translation for every DL-PA formula into a propositional one (cf. Section 2.4), easing the application of automated reasoning techniques.

In this paper we translate three classical computational problems in judgment aggregation as the verification of DL-PA specifications, showing the flexibility of the language in a variety of situations. We begin by translating a wide range of aggregation rules proposed in the literature on judgment aggregation as DL-PA programs, guaranteeing that the size of each program remains polynomial in the number of agents and issues. While this translation is straightforward for most well-known rules, we show that non-trivial rules based on minimization also correspond to a relatively simple DL-PA formula. Aggregation rules are usually justified in reason of the axiomatic properties they possess, and the properties themselves often serve to prove limitative results on the boundaries of aggregation — the notorious impossibility theorems. We thus translate as DL-PA formulas the most common aggregation axioms and we interpret them on the DL-PA translation of rules previously obtained. Finally, we focus on the problem of ensuring a safe aggregation process, i.e., identifying constraints whereby aggregating individual judgments yields a consistent result, obtaining DL-PA formulas whose satisfaction corresponds to various levels of safety.

The paper is organized as follows. We start in Section 2 by providing the basic definitions of judgment aggregation and DL-PA, and we set the stage for a translation of the former into the latter. In Section 3 we propose DL-PA programs to compute judgment aggregation procedures. Section 4 provides translations for the axiomatic properties of aggregation functions, and Section 5 focuses on formulas characterizing safe aggregation. In Section 6 we analyze and compare our results with the literature and consider the case for automated reasoning. Section 7 concludes and points to a number of directions for future work.

2 From Judgment Aggregation to DL-PA

In this section we introduce the notation and formal background of both judgment aggregation and *star-free* dynamic logic of propositional assignments. We present the translation of DL-PA into propositional logic, given its importance for using automated reasoning tools. Furthermore, we provide our first contribution by showing how to translate any instance of a judgment aggregation problem into DL-PA.

2.1 Binary Aggregation with Integrity Constraints

In judgment aggregation agents give acceptance/rejection opinions over logically connected issues to get a collective choice. There are two main frameworks that can be considered: the classic *formula-based* model (List and Pettit, 2002), in which individuals vote directly on complex logical formulas, and *binary aggregation with integrity constraints* (Dokow and Holzman, 2009; Grandi

and Endriss, 2011) where agents have binary opinions on atomic issues linked by an integrity constraint. The two formalisms are equivalent, in the sense that we can translate a problem from one setting into the other and vice versa (Endriss et al., 2016), and we choose the latter for ease of presentation.

Let $\mathcal{I} = \{1, \ldots, m\}$ be a finite non-empty set of *issues*, on which the *agents* in the finite non-empty set $\mathcal{N} = \{1, \ldots, n\}$, for odd n (as we shall see, this is just a technical assumption), express a binary opinion. Individual opinions form a *boolean combinatorial domain* $\mathcal{D} = \{0, 1\}^m$, where 1 denotes acceptance and 0 rejection. A simple propositional language can be defined from the set of propositional symbols $\{p_1, \ldots, p_m\}$, with one atom per issue in $\mathcal{I}$. Then, *integrity constraints* are formulas of this language that we denote by IC. They express the logical interdependencies among the issues, where $\mathrm{IC} = \top$ if there is none. Consider now the following classical example of aggregation, known in the literature as the *discursive dilemma* (List, 2012):

Example 1. *Three judges have to decide whether (1) a defendant is liable for breaching a contract, depending on whether (2) the contract forbade a particular action and (3) the defendant did the action anyway. Let thus* $\mathrm{IC} = p_1 \leftrightarrow (p_2 \wedge p_3)$ *be the constraint expressing the aforementioned law on contracts, and consider the profile below:*

	1	2	3
Judge 1	*1*	*1*	*1*
Judge 2	*0*	*0*	*1*
Judge 3	*0*	*1*	*0*
Majority	*0*	*1*	*1*

Observe that while each of the judges respects the constraint, the majority outcome does not.

A *ballot* $B = (b_1 \ldots b_m) \in \mathcal{D}$ is a particular choice of zeroes and ones for the issues. For example, the second judge's ballot in Example 1 is (001). We interchangeably see a ballot B as an assignment of truth values to the propositional variables in $\{p_1, \ldots, p_m\}$. The *Hamming distance* measures in our setting how much two ballots disagree on the issues, and is defined as $H(B, B^*) := |\{j \in \mathcal{I} \mid b_j \neq b_j^*\}|$. For example, if $B_1 = (111)$ and $B_2 = (001)$, we have $H(B_1, B_2) = 2$, since they differ on the first two issues.

The set of all ballots satisfying IC, written $\mathrm{Mod(IC)} = \{B \mid B \models \mathrm{IC}\}$, is called the *models* of IC. For instance, $\mathrm{Mod(IC)} = \{(111), (001), (010)\}$ for the IC presented in Example 1. We denote by B_i the *individual ballot* of agent i and we assume all agents to be rational, i.e., $B_i \in \mathrm{Mod(IC)}$ for all $i \in \mathcal{N}$. A *profile* $\boldsymbol{B} = (B_1, \ldots, B_n)$ collects all the individual ballots of the agents, such that b_{ij} indicates the j-th element of ballot B_i in $\boldsymbol{B}$. The set $N_{j:1}^{\boldsymbol{B}} = \{i \in \mathcal{N} \mid b_{ij} = 1\}$ is the *coalition of supporters* of issue j in $\boldsymbol{B}$. An *aggregation procedure*, which we also call *aggregation rule* or *aggregator*, is a function F mapping a rational profile to a (possibly irrational) non-empty set of ballots. The following is the formal definition:

Definition 1. *Given a set of agents* $\mathcal{N}$*, a set of issues* $\mathcal{I}$ *and an integrity constraint* IC*, an* aggregation procedure *is a function* $F : \mathrm{Mod(IC)}^{\mathcal{N}} \to (2^{\mathcal{D}} \setminus \emptyset)$*, for* $2^{\mathcal{D}}$ *the powerset of* $\mathcal{D}$*.*

A rule is called *resolute* if its outcome is a singleton for every profile, and *irresolute* otherwise. We denote by $F(\boldsymbol{B})_j$ the outcome of a resolute aggregation rule on issue j. An example of aggregator is the majority rule used in Example 1.

2.2 Dynamic Logic of Propositional Assignments

Dynamic Logic of Propositional Assignments has both formulas and programs, and atomic programs modify the truth values of propositional variables. Our choice of logical language to describe problems in judgment aggregation is that of *star-free* DL-PA, meaning that we do not make use of unbounded iteration.[1]

The language of star-free DL-PA is thus given by the following Backus-Naur grammar:

$$\begin{aligned} \varphi &::= p \mid \top \mid \bot \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\pi\rangle\varphi \\ \pi &::= +p \mid -p \mid \pi\,;\pi \mid \pi \cup \pi \mid \varphi? \end{aligned}$$

where p ranges over $\mathbb{P} = \{p, q, \dots\}$, a countable set of propositional variables.

Atomic *formulas* consist of variables and constants $\top$ and $\bot$. Complex formulas are built via negation $\neg$, disjunction $\vee$, and a diamond modality for each program $\langle\pi\rangle$. Other boolean connectives (e.g., conjunction $\wedge$, implication $\rightarrow$, biconditional $\leftrightarrow$, exclusive disjunction $\oplus$) and the dual operator $[\pi]\varphi$ are defined in the usual way. Atomic *programs* $+p$ and $-p$ assign value true or false to variable p, respectively. Sequential composition $\pi\,;\pi'$ executes first π and then π', nondeterministic union $\pi \cup \pi'$ nondeterministically chooses to execute either π or π', and test $\varphi?$ checks that φ holds (and fails otherwise).

A *valuation* v is a subset of $\mathbb{P}$ that specifies the truth value of every propositional variable, so that $\mathbb{V} = 2^{\mathbb{P}} = \{v_1, v_2, \dots\}$ is the set of all valuations. When $p \in v$, we say that p is *true* in v (and we say that p is *false* in v otherwise). As illustrated in Table 1, DL-PA programs are interpreted through a unique relation between valuations.[2]

$$\begin{aligned} \|p\| &= \{v \in \mathbb{V} \mid p \in v\} \\ \|\top\| &= 2^{\mathbb{P}} \\ \|\bot\| &= \emptyset \\ \|\neg\varphi\| &= 2^{\mathbb{P}} \setminus \|\varphi\| \\ \|\varphi \vee \psi\| &= \|\varphi\| \cup \|\psi\| \\ \|\langle\pi\rangle\varphi\| &= \{v \in \mathbb{V} \mid \text{there is } v_1 \text{ s.t. } (v, v_1) \in \|\pi\| \text{ and } v_1 \in \|\varphi\|\} \\ \|+p\| &= \{(v_1, v_2) \mid v_2 = v_1 \cup \{p\}\} \\ \|-p\| &= \{(v_1, v_2) \mid v_2 = v_1 \setminus \{p\}\} \\ \|\pi\,;\pi'\| &= \|\pi\| \circ \|\pi'\| \\ \|\pi \cup \pi'\| &= \|\pi\| \cup \|\pi'\| \\ \|\varphi?\| &= \{(v, v) \mid v \in \|\varphi\|\} \end{aligned}$$

Table 1: Interpretation of DL-PA connectives and programs

[1] This logic is obtained from full DL-PA via elimination of the Kleene star, as shown by Balbiani et al. (2013).

[2] We can construct a Kripke model $M^{\text{DL-PA}}$ for PDL by letting the valuations be the states, by including $\{+p \mid p \in \mathbb{P}\} \cup \{-p \mid p \in \mathbb{P}\}$ in the set of atomic programs, and by considering the identity function for the valuation (Balbiani et al., 2013).

As a notational convention, formulas will start by an uppercase letter, while programs and counters will start by a lowercase letter. The standard programming language primitives can be expressed in PDL, and we thus have $\mathsf{skip} := \top?$, $\mathsf{if}\ \varphi\ \mathsf{then}\ \pi_1\ \mathsf{else}\ \pi_2 := (\varphi?\,;\pi_1) \cup (\neg\varphi?\,;\pi_2)$, $p \hookleftarrow q := \mathsf{if}\ q\ \mathsf{then} +p\ \mathsf{else} -p$ and $\mathsf{if}\ \varphi\ \mathsf{do}\ \pi := \mathsf{if}\ \varphi\ \mathsf{then}\ \pi\ \mathsf{else}\ \mathsf{skip}$. We let the conjunction of an empty set of formulas be $\top$, and the sequential and nondeterministic composition of an empty set of programs be skip. Namely, $\bigwedge_{\varphi\in\emptyset}\varphi := \top$, and $\bigcup_{\pi\in\emptyset}\pi := \mathsf{skip}$, and $;_{\pi\in\emptyset}\,\pi := \mathsf{skip}$.

2.3 Basic DL-PA Programs and Formulas

In this section we present useful $\mathsf{DL\text{-}PA}$ programs and formulas, for the most part already introduced by Balbiani et al. (2013).

In $\mathsf{DL\text{-}PA}$ we can repeatedly execute a program n times or up to n times, as follows:

$$\begin{aligned}
\pi^n &:= \pi\,;\pi^{n-1}\\
\pi^{\leq n} &:= (\mathsf{skip} \cup \pi)\,;\pi^{\leq n-1}
\end{aligned}$$

with the convention that both programs are equal to skip in case $n = 0$.

Any natural number $s \in \mathbb{N}_0$ can be written in $\mathsf{DL\text{-}PA}$ via its binary expression, thanks to a conjunction of $t = \lfloor \log s \rfloor + 1$ variables. If x is the binary expression of s, we use a conjunction of literals q_i and $\neg q_i$, with $i \in \{0, \ldots, \lfloor \log s \rfloor\}$, such that a non-negated variable means that the corresponding binary digit in x is a 1, while a negated variable indicates a 0. For instance, if $s = 14$, we have that $x = 1110$ and the corresponding formula in $\mathsf{DL\text{-}PA}$ is $\mathsf{14} :=\ q_3 \wedge q_2 \wedge q_1 \wedge \neg q_0$.

The following two programs increment (up to 2^{t-1}) and set to zero, i.e., assign truth value false to all the variables in P, a given t-bit counter:

$$\begin{aligned}
\mathsf{incr}(\mathsf{x}^t) &:= \neg\Big(\bigwedge_{0\leq i\leq t-1} q_i^x\Big)?\,;\ \bigcup_{0\leq k\leq t-1} \Big(\big(\neg q_k^x \wedge \bigwedge_{0\leq i\leq k-1} q_i^x\big)?\,;+q_k^x\,;\ \mathop{;}_{0\leq i\leq k-1} -q_i^x\Big)\\
\mathsf{zero}(P) &:= \mathop{;}_{p\in P} -p
\end{aligned}$$

where $\mathsf{x}^t := \{q_i^x \mid 0 \leq i < t\}$ is a set of variables. Observe that in program $\mathsf{zero}(P)$ the order in which elements are set to false does not matter, as it leads to identical interpretations.

We can check whether two numbers are equal, whether one is greater than the other, or whether one is greater than or equal to the other, via the following $\mathsf{DL\text{-}PA}$ formulas:

$$\begin{aligned}
\mathsf{x}^t = \mathsf{y}^t &:= \bigwedge_{0\leq k<t} q_k^x \leftrightarrow q_k^y\\
\mathsf{x}^t > \mathsf{y}^t &:= \bigvee_{0\leq k<t} \Big(\Big(\bigwedge_{k<i<t} (q_i^x \leftrightarrow q_i^y)\Big) \wedge q_k^x \wedge \neg q_k^y\Big)\\
\mathsf{x}^t \geq \mathsf{y}^t &:= \mathsf{x}^t > \mathsf{y}^t \vee \mathsf{x}^t = \mathsf{y}^t
\end{aligned}$$

where the general idea is to compare the digits at the same position in the binary expressions of the two numbers.[3]

---

[3] In some cases we need to compare numbers that can be expressed with *different* minimal amounts of binary digits. In programs where multiple counters are used, we take the maximal value taken by a counter as the upper

We may need to reverse the truth value of some variables in a set P. The first program below flips the truth value of a single nondeterministically chosen variable in P. The second nondeterministically resets the truth value of all variables in P to some new value: as a result, either their truth value has been flipped or not.

$$\begin{aligned}
\mathsf{flip}^1(P) &:= \bigcup_{p \in P} (p \leftarrow\!\!\dashv \neg p) \\
\mathsf{flip}^{\geq 0}(P) &:= \mathop{;}_{p \in P} (+p \cup -p)
\end{aligned}$$

Finally, the next two formulas hold when different types of minimization are achieved. The first is true if and only if it is not possible to make φ true by flipping the truth values of the variables in a strict subset of the non-empty set P. The second holds if and only if the Hamming distance to a state where φ holds is at least s, where the variables outside P are kept constant.

$$\begin{aligned}
\mathsf{D}(\varphi, P) &:= \neg\langle \bigcup_{p \in P} \mathsf{flip}^{\geq 0}(P \setminus \{p\})\rangle\varphi \\
\mathsf{H}(\varphi, P, \geq s) &:= \begin{cases} \top & \text{if } s = 0 \\ \neg\langle \mathsf{flip}^1(P)^{\leq s-1}\rangle\varphi & \text{if } s > 0 \end{cases}
\end{aligned}$$

Observe that $\mathsf{D}(\varphi, P)$ does not imply that φ *will* hold if we flip the truth value of all the variables in P. In our setting this definition suffices, but such an alternative formulation has been given as well by Herzig (2014).

2.4 From DL-PA to Propositional Logic

A translation from DL-PA formulas to propositional formulas was proposed by Balbiani et al. (2013). We here give it for the star-free fragment of DL-PA. It is presented in terms of reduction axioms: equivalences that, first, simplify programs into atomic programs, second, 'push' atomic programs across the boolean connectives inside formulas until they meet an atomic formula, and, third, eliminate atomic programs.

$$\begin{gathered}
[\varphi?]\psi \leftrightarrow \varphi \to \psi \\
[\pi_1 ; \pi_2]\varphi \leftrightarrow [\pi_1][\pi_2]\varphi \\
[\pi_1 \cup \pi_2]\varphi \leftrightarrow [\pi_1]\varphi \wedge [\pi_2]\varphi \\
[\pi]\neg\varphi \leftrightarrow \neg[\pi]\varphi \\
[\pi](\varphi_1 \wedge \varphi_2) \leftrightarrow [\pi]\varphi_1 \wedge [\pi]\varphi_2
\end{gathered}$$

---

bound for *all* other counters in that program. Let t be the maximal number of variables needed to express the maximal value a counter can take in a program: if another number is expressible by using only $k < t$ variables, it will nonetheless be expressed by t variables with $\neg q_i$ for all i such that $k < i \leq t$. We then write x instead of x^t.

$$[+p]q \leftrightarrow \begin{cases} \top & \text{if } p = q \\ q & \text{otherwise} \end{cases}$$

$$[-p]q \leftrightarrow \begin{cases} \bot & \text{if } p = q \\ q & \text{otherwise} \end{cases}$$

For example, the DL-PA formula $[(+p \cup -q)\,;(p \wedge r?)]r \vee q$ is reduced to the propositional formula $(r \to (r \vee q)) \wedge ((p \wedge r) \to r)$. Note that when translating formulas such as $\mathsf{flip}^{\geq 0}(P)$ we are going to visit all valuations with all possible combinations for the values of the variables in P, which gives an exponential explosion in the length of the translation.

2.5 Aggregation Problems Translated into DL-PA

We here show how to translate profiles and aggregation rules from judgment aggregation into DL-PA. The former are turned into a valuation, while the latter become programs. As a first step, let $\mathbb{B} := \{p_{ij} \mid i, j \in \mathbb{N}\}$ be the subset of $\mathbb{P}$ whose variables encode the opinion of agent i on issue j. Analogously, $\mathbb{O} := \{p_j \mid j \in \mathbb{N}\}$ is the subset of $\mathbb{P}$ whose variables refer to the possible output for any issue j.

From these two infinite sets, we derive two finite subsets for specific n agents and m issues. Namely, for $\mathcal{N} = \{1, \ldots, n\}$ and $\mathcal{I} = \{1, \ldots, m\}$, we let $\mathbb{B}^{n,m} := \{p_{ij} \mid i \in \mathcal{N} \text{ and } j \in \mathcal{I}\}$ be the set of propositional variables referring to the decision of the agents in $\mathcal{N}$ on the issues in $\mathcal{I}$, and we let the variables in $\mathbb{O}^m := \{p_j \mid j \in \mathcal{I}\}$ refer to the collective decision on the issues in $\mathcal{I}$. Finally, we define an additional set $\mathbb{U} := \{q_i \mid i \in \mathbb{N}\}$ of variables of $\mathbb{P}$ that are used to encode finitely many counters in our programs. We suppose $\mathbb{B}$, $\mathbb{O}$ and $\mathbb{U}$ to be disjoint.

The following definition identifies the valuations that correspond to a profile in judgment aggregation.

Definition 2. *Valuation* $v_{\boldsymbol{B}}$ translates *profile* $\boldsymbol{B} = (B_1, \ldots, B_n)$ *on* m *issues, in case:*

(i) $v_{\boldsymbol{B}} \subseteq \mathbb{B}^{n,m}$*, and*

(ii) $p_{ij} \in v_{\boldsymbol{B}}$ *if and only if* $b_{ij} = 1$.

The first condition ensures that only variables corresponding to the decision of the agents on the issues could possibly be true in $v_{\boldsymbol{B}}$. This means, in particular, that counters are initially set to zero. According to the second condition, a variable in $v_{\boldsymbol{B}}$ is true if and only if the corresponding entry in profile $\boldsymbol{B}$ has value 1. For example, if we have profile $\boldsymbol{B} = ((01), (00), (10))$ for 3 agents and 2 issues, the set $\mathbb{B}^{3,2} = \{p_{11}, p_{12}, p_{21}, p_{22}, p_{31}, p_{32}\}$ corresponds to the entries in the profile, the set $\mathbb{O}^2 = \{p_1, p_2\}$ handles the outcome of aggregation rules and valuation $v_{\boldsymbol{B}} = \{p_{12}, p_{31}\} \subseteq \mathbb{B}^{3,2}$ encodes the profile.

Let $V^{\mathsf{f}}_{v_{\boldsymbol{B}}} = \{v' \cap \mathbb{O}^m \mid (v_{\boldsymbol{B}}, v') \in \| \mathsf{f}(\mathbb{B}^{n,m}) \|\}$ be the set of valuations v' restricted to $\mathbb{O}^m$ reachable from $v_{\boldsymbol{B}}$ through $\mathsf{f}(\mathbb{B}^{n,m})$. We now translate aggregation rules into DL-PA programs.

Definition 3. *Program* $\mathsf{f}(\mathbb{B}^{n,m})$ translates *aggregation rule* F*, if for all profiles* $\boldsymbol{B}$ *and valuations* $v_{\boldsymbol{B}}$ *translating* $\boldsymbol{B}$ *according to Definition 2, it is the case that* $V^{\mathsf{f}}_{v_{\boldsymbol{B}}} = F(\boldsymbol{B})$.

In Definition 3 we thus compare the truth values of the outcome variables in $\mathbb{O}^m$ in the valuations reachable after the execution of program $\mathsf{f}(\mathbb{B}^{n,m})$ with the outcome ballots of rule F (recall that we interchangeably see a ballot as a valuation over $\{p_1, \ldots, p_m\}$). If the program translates the rule we should get the same set on both sides — a singleton for resolute rules. This definition will be used in the proofs of Section 3 to ensure that our translations are correct.

Before proceeding, we stress an important point. Since aggregation rules are defined over a specific number of issues, number of agents and integrity constraint, the programs we provide as their DL-PA translation are to be intended as general 'program schemas': a set of issues $\mathcal{I}$, set of agents $\mathcal{N}$ and constraint IC need to be provided to completely spell them out.

The integrity constraint is written as a formula IC over variables in $\mathbb{O}^m$. To check whether a particular choice of truth values over $\mathbb{B}^{n,m}$ corresponds to a profile, i.e., all the individual ballots satisfy the constraint, the following formula must hold:

$$\mathsf{Rational}_{\mathrm{IC}}(\mathbb{B}^{n,m}) \ := \bigwedge_{i \in \mathcal{N}} \langle \underset{j \in \mathcal{I}}{;} p_j \leftarrowtail p_{ij} \rangle \mathrm{IC}.$$

That is, we check whether by copying into the outcome variables the truth values of the variables for each individual ballot, the formula for the integrity constraint is true.

The next formula is true if and only if we are in a valuation that possibly corresponds to the encoding of a profile, meaning that the initial conditions of Definition 2 hold:

$$\mathsf{Prof}_{\mathrm{IC}}(\mathbb{B}^{n,m}, \mathbb{O}^m) \ := \ (\bigwedge_{p_j \in \mathbb{O}^m} \neg p_j) \wedge \mathsf{Rational}_{\mathrm{IC}}(\mathbb{B}^{n,m}).$$

Program $\mathsf{prof}_{\mathrm{IC}}(\mathbb{B}^{n,m}, \mathbb{O}^m) := \mathsf{Prof}_{\mathrm{IC}}(\mathbb{B}^{n,m}, \mathbb{O}^m)?$ tests that the properties of a profile hold at the current valuation. Observe that in the codomain of $\| \mathsf{prof}_{\mathrm{IC}}(\mathbb{B}^{n,m}, \mathbb{O}^m) \|$ the outcome variables are false, but this is not enough to conclude that condition (i) of Definition 2 holds. Nevertheless, all programs translating aggregation rules will only need to inspect variables in $\mathbb{B}^{n,m}$ and (possibly) change the truth values of variables in $\mathbb{O}^m$, and they will set to zero all counters as the first step. Hence, the valuation reached after the execution of $\mathsf{prof}_{\mathrm{IC}}(\mathbb{B}^{n,m}, \mathbb{O}^m)$ will be considered as encoding a profile as well.

With a slight abuse of notation, in the rest of this paper we drop the superscripts from $\mathbb{B}^{n,m}$ and $\mathbb{O}^m$ to simplify the reading of the programs. It will be clear from context when we are referring to the infinite sets $\mathbb{B}$ and $\mathbb{O}$ instead.

3 Aggregation Rules

Aggregation rules are the basic bricks of judgment aggregation, since they provide different ways to produce collective choices from the individual judgments of the agents on the issues. In this section we translate as DL-PA programs some of the most studied examples of aggregation procedures. We prove the correctness of our translation for a resolute and an irresolute rule, omitting the proofs whenever they can be obtained as a straightforward adaptation of the presented ones.

3.1 Expressibility of Aggregation Rules

The aptness of DL-PA to model judgment aggregation can be assessed by its capability to express judgment aggregation procedures. Our first general result is thus a positive one: we prove below that *any* aggregation rule, as introduced in Definition 1, can be expressed as a DL-PA program.

Theorem 1. *For every $\mathcal{N}$, $\mathcal{I}$ and* IC*, all aggregation rules $F : \mathrm{Mod}(\mathrm{IC})^{\mathcal{N}} \to 2^{\mathcal{D}} \setminus \emptyset$ are expressible as* DL-PA *programs.*

Proof. We start by examining the case of a resolute aggregator F. Consider the DL-PA program consisting of a sequential composition of sub-programs of the form $\mathsf{if}\ \varphi_{\boldsymbol{B}}\ \mathsf{do}\ \pi_{F(\boldsymbol{B})}$ for each profile $\boldsymbol{B}$, for $\varphi_{\boldsymbol{B}} := (\bigwedge_{j\in\mathcal{I}} \bigwedge_{i\in N^{\boldsymbol{B}}_{j:1}} p_{ij}) \wedge (\bigwedge_{j\in\mathcal{I}} \bigwedge_{i\in(\mathcal{N}\setminus N^{\boldsymbol{B}}_{j:1})} \neg p_{ij})$ and $\pi_{F(\boldsymbol{B})} := \mathbin{;}_{\{j\in\mathcal{I}|F(\boldsymbol{B})_j=1\}} +p_j \,;\, \mathbin{;}_{\{j\in\mathcal{I}|F(\boldsymbol{B})_j=0\}} -p_j$. Namely, $\varphi_{\boldsymbol{B}}$ completely identifies profile $\boldsymbol{B}$ and $\pi_{F(\boldsymbol{B})}$ modifies the outcome variables according to the result of F on profile $\boldsymbol{B}$.

If F is irresolute it is sufficient to consider a sequential composition of sub-programs of the form $\mathsf{if}\ \varphi_{\boldsymbol{B}}\ \mathsf{do}\ \bigcup_{B\in F(\boldsymbol{B})} \pi_B$, where π_B is defined as $\pi_B := \mathbin{;}_{\{j\in\mathcal{I}|b_j=1\}} +p_j \,;\, \mathbin{;}_{\{j\in\mathcal{I}|b_j=0\}} -p_j$, generating a non-deterministic program whose output consists of all outcomes of F. These two types of programs provide a straightforward translation of resolute and irresolute rules. □

Theorem 1 shows that DL-PA is fully expressive when it comes to translating judgment aggregation rules. Nevertheless, observe that the formulas used in the proof are all of size exponential in the number of individuals and issues. More precisely, since all profiles explicitly occur in the specification of the programs, the size is in the order of $2^{|\mathcal{I}|\cdot|\mathcal{N}|}$. Therefore, in the remainder of this section we present compact programs for a selection of well-known rules proposed in the literature on judgment aggregation (cf. Appendix A for an example illustrating the compactness of our translations). The ideas appearing in the construction of these programs suggest how to translate other rules that are not explicitly presented here.

3.2 Simple Aggregation Rules

We begin our study of compact representations of judgment aggregation rules with a section on what we call *simple* rules. By this term we mean resolute rules relatively easy to explain and understand, which regularly occur in real-world scenarios.

3.2.1 Dictatorship of Agent i

While in general the dictatorial rule is an unattractive procedure to use, it gives us perhaps the simplest example of aggregator. The outcome of the dictatorship of some fixed agent $i \in \mathcal{N}$ for all profiles $\boldsymbol{B}$ is her individual ballot. Namely, for all $j \in \mathcal{I}$ $\mathrm{Dictatorship}_i(\boldsymbol{B})_j = 1$ if and only if $b_{ij} = 1$. In the following proposition we find the translation of $\mathrm{Dictatorship}_i$ as a DL-PA program. The given proof serves as an example for all propositions on resolute rules.

Proposition 1. *Let $\mathcal{I}$ and $\mathcal{N}$ be given, and let* $\mathsf{dictatorship}_i(\mathbb{B}) := \mathbin{;}_{j\in\mathcal{I}}(p_j \leftarrow p_{ij})$*. Then, program* $\mathsf{dictatorship}_i$ *translates rule* $\mathrm{Dictatorship}_i$*.*

Proof. To shorten notation, we call F the aggregation rule $\mathrm{Dictatorship}_i$, and f the program $\mathsf{dictatorship}_i$. By Definition 3 we need to show that for all profiles $\boldsymbol{B}$, if $v_{\boldsymbol{B}}$ is the valuation

translating $\boldsymbol{B}$ (cf. Definition 2), and $(v_{\boldsymbol{B}}, v') \in \| \mathsf{f}(\mathbb{B}) \|$, then for all $j \in \mathcal{I}$ we have $F(\boldsymbol{B})_j = 1$ if and only if $p_j \in v'$.

For the left-to-right direction, consider an arbitrary $j \in \mathcal{I}$ such that $F(\boldsymbol{B})_j = 1$: we have to show that $p_j \in v'$. From the definition of F, we have that $F(\boldsymbol{B})_j = 1$ if and only if $b_{ij} = 1$, where agent i is the dictator. Let $v_{\boldsymbol{B}}$ be the valuation translating $\boldsymbol{B}$. By construction, we have that $p_{ij} \in v_{\boldsymbol{B}}$ and $p_j \notin v_{\boldsymbol{B}}$.

Let v' be the outcome valuation, i.e., such that $(v_{\boldsymbol{B}}, v') \in \| \mathsf{f}(\mathbb{B}) \|$. There is exactly one such valuation. By the definition of F and by applying program equivalences, we get that there are valuations $v_1, \ldots, v_{m+1}$ such that $v_1 = v_{\boldsymbol{B}}$, $v_{m+1} = v'$, and

$$(v_1, v_2) \in \| (p_{i1}?\,; +p_1) \cup (\neg p_{i1}?\,; -p_1) \|, \quad \ldots, \quad (v_m, v_{m+1}) \in \| (p_{im}?\,; +p_m) \cup (\neg p_{im}?\,; -p_m) \| \,.$$

Let $a \in \{1, \ldots, m\}$ and b be two indices such that $b = a + 1$ and $(v_a, v_b) \in \| (p_{ij}?\,; +p_j) \cup (\neg p_{ij}?\,; -p_j) \|$ for the arbitrary issue $j \in \mathcal{I}$ we are considering. Let us call $A = \{(v, v^*) \mid p_{ij} \in v \text{ and } v^* = v \cup \{p_j\}\}$ and $B = \{(v, v^*) \mid \neg p_{ij} \in v \text{ and } v^* = v \setminus \{p_j\}\}$. By the interpretation of DL-PA programs given in Table 1 we get that $(v_a, v_b) \in A \cup B$. We thus have to check whether $(v_a, v_b) \in A$ or $(v_a, v_b) \in B$, since A and B are disjoint sets. In order to do so, we have to inspect whether $p_{ij} \in v_a$ or $\neg p_{ij} \in v_a$.

We now show that $p_{ij} \in v_a$, which will allow us to conclude that $p_j \in v'$. In case $a = 1$, we have that $v_a = v_{\boldsymbol{B}}$ and thus $p_{ij} \in v_{\boldsymbol{B}}$ by assumption. Otherwise, note that $v_1, \ldots, v_a$ only (possibly) differ in the truth value assigned to $p_1, \ldots, p_{j-1}$, respectively. Thus, since $p_{ij} \in v_{\boldsymbol{B}} = v_1$ and the truth value of p_{ij} has not been modified in $v_2, \ldots, v_{a-1}$, we have that $p_{ij} \in v_a$. Hence, $(v_a, v_b) \in A$, and $v_b = v_a \cup \{p_j\}$. Observe also that $v_{b+1}, \ldots, v_{m+1}$ only (possibly) differ in the truth values assigned to $p_{j+1}, \ldots, p_m$. Therefore, we can conclude that $p_j \in v_{m+1} = v'$.

For the right-to-left direction, we can equivalently show that if $F(\boldsymbol{B})_j = 0$ then $p_j \notin v'$. The proof is analogous to the one presented above, with the sole exception that we have to show that $(v_a, v_b) \in B$ in order to conclude that $p_j \notin v'$. □

The length of the program $\mathsf{dictatorship}_i$ increases only if the number of issues does, as it considers just the truth values of the dictator's variables while ignoring the rest.

3.2.2 Quota Rules

A *quota rule* specifies for each issue a certain threshold of support that has to be reached in order for the issue to be accepted in the outcome (Dietrich and List, 2007b). The quota q can be any integer such that $1 \leq q \leq |\mathcal{N}|$. In case all issues have the same quota, we speak of *uniform* quota rules. If q_j is the quota for issue $j \in \mathcal{I}$ and $\vec{q} = (q_1, \ldots, q_m)$, we have:

$$\text{Quota}_{\vec{q}}(\boldsymbol{B})_j = 1 \text{ if and only if } |N^{\boldsymbol{B}}_{j:1}| \geq q_j.$$

We now state a result that provides, for every choice of quotas, a DL-PA program translating the corresponding quota rule.

Proposition 2. *Let $\mathcal{I}$ be a set of issues, $\mathcal{N}$ a set of agents, and $1 \leq q_1, \ldots, q_m \leq |\mathcal{N}|$. Let $\mathsf{supp} := \{q_1, \ldots, q_{\log n}\}$ and $\mathsf{quota}_j := \{q'_1, \ldots, q'_{\log n}\}$ for $j \in \mathcal{I}$ be disjoint subsets of $\mathbb{U}$. The*

$\text{Quota}_{\vec{q}}$ *rule is translated into the following* DL-PA *program:*

$$\begin{aligned}\mathsf{quota}_{\vec{q}}(\mathbb{B}) \;:=\; & \mathop{;}_{j\in\mathcal{I}} \mathsf{zero}(\mathsf{quota}_j)\,;\; \mathop{;}_{j\in\mathcal{I}} \mathsf{incr}(\mathsf{quota}_j)^{q_j}\,;\; \mathop{;}_{j\in\mathcal{I}} \big(\mathsf{zero}(\mathsf{supp})\,;\\ & (\mathop{;}_{i\in\mathcal{N}} \mathsf{if}\ p_{ij}\ \mathsf{do}\ \mathsf{incr}(\mathsf{supp}))\,;\mathsf{if}\ \mathsf{supp} \geq \mathsf{quota}_j\ \mathsf{do}\ +p_j\big).\end{aligned}$$

The subprogram $\mathsf{incr}(\mathsf{quota}_j)^{q_j}$ sets the counter variables in the set quota_j to the value q_j and the counter variables in supp keep track of the support for the issue currently inspected.

A special instance of quota rules is the *majority rule*, an intuitive aggregator recurring in many everyday examples, where for all issues the quota is fixed at $\frac{n+1}{2}$. Recall that we assumed the number of agents to be odd, a common assumption that permits to leave aside the question of how to adapt the definition of majority for an even number of agents: should we accept an issue if and only if *more* than half of the agents accept it individually (strict majority), or if *at least* half of the agents accept it (weak majority)? Either way, if we want to keep the rule resolute we generate an outcome biased towards acceptance or rejection in all profiles where an issue is accepted by *exactly* half of the agents.

The majority rule being a quota rule, we could express it in DL-PA with the same program schema of Proposition 2. Nonetheless, we provide below an alternative formulation making use of the counters introduced in Section 2.2, giving us a program of length polynomial in the number of agents.

Proposition 3. *Let $\mathcal{I}$ be a set of issues and $\mathcal{N}$ a set of agents. Let* $\mathsf{pro} := \{q_1, \ldots, q_{\log n}\}$ *and* $\mathsf{con} := \{q'_1, \ldots, q'_{\log n}\}$ *be disjoint subsets of* $\mathbb{U}$*. The majority rule* Maj *is translated into program:*

$$\mathsf{maj}(\mathbb{B}) := \mathop{;}_{j\in\mathcal{I}} \big(\mathsf{zero}(\mathsf{pro}\cup\mathsf{con})\,;\; \mathop{;}_{i\in\mathcal{N}} (\mathsf{if}\ p_{ij}\ \mathsf{then}\ \mathsf{incr}(\mathsf{pro})\ \mathsf{else}\ \mathsf{incr}(\mathsf{con}))\,;\mathsf{if}\ \mathsf{pro} > \mathsf{con}\ \mathsf{do}\ +p_j\big).$$

Since majority is an uniform quota rule, program maj economizes on the number of counters for the quotas. Moreover, it is useful to define it as a separate program since it will be extensively used in the definitions of other aggregation rules.

Note that for the *nomination rule*, i.e., the uniform quota rule with $q = 1$, an even more compact program is simply $\mathsf{nomination}(\mathbb{B}) := \mathop{;}_{j\in\mathcal{I}}(\mathsf{if}\ \bigvee_{i\in\mathcal{N}} p_{ij}\ \mathsf{do}\ +p_j)$.

3.3 Maximization and Minimization Rules

Rules returning a ballot that appears in the input profile, such as Dictatorship_i, guarantee the outcome to satisfy the integrity constraint, while more appealing rules such as majority may fail to do so (as illustrated by Example 1). In this section we present two aggregation rules based on maximization and minimization operations that aim at amending the outcomes of the majority rule when they do not satisfy the constraint.

3.3.1 Maximal Subagenda Rule

The maximal subagenda rule returns all ballots satisfying the integrity constraint and having maximal set-inclusion agreement with the majority outcome (Lang and Slavkovik, 2014):

$$\mathrm{MSA}_{\mathrm{IC}}(\boldsymbol{B}) = \underset{B \models \mathrm{IC}}{\overset{\subseteq}{\arg\max}}\{j \in \mathcal{I} \mid b_j = \mathrm{Maj}(\boldsymbol{B})_j\}.$$

Differently from simple rules, it is not as straightforward to translate $\mathrm{MSA}_{\mathrm{IC}}$ into DL-PA. In fact, we have to encode a maximization operation and we thus need some further notation. Consider the following programs:

$$\begin{aligned}
\mathsf{store}(P) &:= \mathop{;}_{p \in P} p' \leftarrow\!\!\shortmid p \\
\mathsf{restore}^1(P) &:= \bigcup_{p \in P} (p \oplus p'?\,; p \leftarrow\!\!\shortmid p') \\
\mathsf{restore}^{\geq 0}(P) &:= \mathop{;}_{p \in P} (\mathsf{skip} \cup p \leftarrow\!\!\shortmid p').
\end{aligned}$$

The program store stores the truth value of the variables in P in some fresh variables p', program $\mathsf{restore}^1(P)$ restores the truth value of just one variable p that has been previously modified, and program $\mathsf{restore}^{\geq 0}(P)$ restores the truth value of none, some, or all variables.

Inspired by analogous work in the literature on belief change for the Possible Models Approach (Herzig, 2014), we now present and prove the correctness of a program translating $\mathrm{MSA}_{\mathrm{IC}}$. Given that we are dealing with an irresolute rule, we might need to handle multiple outcomes for the same profile, whence the structure of the proof differs from that of Proposition 1. We present it here as an example for all irresolute aggregation rules.

Proposition 4. *Let $\mathcal{I}$ be a set of issues, $\mathcal{N}$ a set of agents and* IC *a propositional formula. The* $\mathrm{MSA}_{\mathrm{IC}}$ *rule is translated into the following* DL-PA *program:*

$$\mathsf{msa}_{\mathrm{IC}}(\mathbb{B}) := \mathsf{maj}(\mathbb{B})\,; \mathsf{store}(\mathbb{O})\,; \mathsf{flip}^{\geq 0}(\mathbb{O})\,; \mathrm{IC}?\,; [\mathsf{restore}^1(\mathbb{O})\,; \mathsf{restore}^{\geq 0}(\mathbb{O})]\neg \mathrm{IC}?.$$

Proof. Consider an arbitrary profile $\boldsymbol{B}$ for a set of agents $\mathcal{N}$ and a set of issues $\mathcal{I}$, and let $v_{\boldsymbol{B}}$ be the valuation translating it according to Definition 2. Given that $\mathrm{MSA}_{\mathrm{IC}}$ is an irresolute rule, we will show that there exists a bijection $g : \mathrm{MSA}_{\mathrm{IC}}(\boldsymbol{B}) \to V_{v_{\boldsymbol{B}}}^{\mathsf{msa}_{\mathrm{IC}}}$ such that $b_j = 1$ if and only if $p_j \in g(B)$ for all $j \in \mathcal{I}$. For $B \in \mathrm{MSA}_{\mathrm{IC}}(\boldsymbol{B})$ define $v' = g(B)$ as follows: $v' = v_{\boldsymbol{B}}$ on all variables in $\mathbb{P} \setminus \mathbb{O}$, and $p_j \in v'$ if and only if $b_j = 1$ for $p_j \in \mathbb{O}$.

We begin by showing that $g(B) = v' \in V_{v_{\boldsymbol{B}}}^{\mathsf{msa}_{\mathrm{IC}}}$ for all $B \in \mathrm{MSA}_{\mathrm{IC}}(\boldsymbol{B})$, proving that the co-domain of g is $V_{v_{\boldsymbol{B}}}^{\mathsf{msa}_{\mathrm{IC}}}$. More precisely, we need to show that there are valuations $v_a, \ldots, v_f$ such that $v_a = v_{\boldsymbol{B}}$, $v_f = v'$, and

$$\begin{aligned}
&(v_a, v_b) \in \|\,\mathsf{maj}(\mathbb{B})\,\|,\ (v_b, v_c) \in \|\,\mathsf{store}(\mathbb{O})\,\|,\ (v_c, v_d) \in \|\,\mathsf{flip}^{\geq 0}(\mathbb{O})\,\|, \\
&(v_d, v_e) \in \|\,\mathrm{IC}?\,\|,\ (v_e, v_f) \in \|\,[\mathsf{restore}^1(\mathbb{O})\,; \mathsf{restore}^{\geq 0}(\mathbb{O})]\neg \mathrm{IC}?\,\|\ .
\end{aligned}$$

Recall that program maj is deterministic: hence, we let v_b be the unique valuation reachable from v_a after its execution. Program $\mathsf{store}(\mathbb{O})$ is deterministic as well, hence let v_c be the unique

valuation reachable from v_b after its execution. We now have to consider two cases: either (a) $\mathrm{Maj}(\boldsymbol{B}) \models \mathrm{IC}$, or (b) $\mathrm{Maj}(\boldsymbol{B}) \not\models \mathrm{IC}$.

(a) Since $B = \mathrm{Maj}(\boldsymbol{B})$, we consider the execution of program $\mathsf{flip}^{\geq 0}(\mathbb{O})$ such that no variable gets its truth value flipped (i.e., $v_d = v_c$). By Proposition 3 we know that $p_j \in v_b$ if and only if $\mathrm{Maj}(\boldsymbol{B})_j = 1$, for all $j \in \mathcal{I}$. Hence, since by assumption $\mathrm{Maj}(\boldsymbol{B}) \models \mathrm{IC}$, we have that $v_c \in \| \mathrm{IC} \|$. Then, take $v_e = v_d = v_f$. It is now easily seen that, since the variables in $\mathbb{O}$ and the fresh ones used by program $\mathsf{store}(\mathbb{O})$ have identical truth value, program $\mathsf{restore}^1(\mathbb{O})$ fails at v_f, and therefore $[\mathsf{restore}^1(\mathbb{O})\,;\mathsf{restore}^{\geq 0}(\mathbb{O})]\neg\mathrm{IC}$ holds vacuously in v_f. Since $v' = v_f = v_c$, and v_c differs from v_b only in the assignment to the fresh variables, and $p_j \in v_b$ if and only if $\mathrm{Maj}(\boldsymbol{B})_j = 1$ for all $j \in \mathcal{I}$, and $B = \mathrm{Maj}(\boldsymbol{B})$, we provided a valuation $v' \in V_{v_{\boldsymbol{B}}}^{\mathsf{msa}_{\mathrm{IC}}}$ such that $p_j \in v'$ if and only if $b_j = 1$, for all $j \in \mathcal{I}$.

(b) Since $B \in \mathrm{MSA}_{\mathrm{IC}}(\boldsymbol{B})$, by the definition of $\mathrm{MSA}_{\mathrm{IC}}$ we know that $B \models \mathrm{IC}$ and that there is a maximal (with respect to set inclusion) subset Q of issues in $\mathrm{Maj}(\boldsymbol{B})$ such that $b_j = \mathrm{Maj}(\boldsymbol{B})_j$. Let now $S = \{p_i \in \mathbb{O} \mid i \notin Q\}$: we take v_d to be the valuation that differs from v_c only in that the truth values of all the variables in S have been flipped. It is immediately seen that $(v_c, v_d) \in \| \mathsf{flip}^{\geq 0}(\mathbb{O}) \|$. By our choice of valuation based on the features of $B \in \mathrm{MSA}_{\mathrm{IC}}(\boldsymbol{B})$ we have that $v_e = v_d$ satisfies IC. By a similar reasoning, we can see that by taking $v_f = v_e$ we get $(v_e, v_f) \in \| [\mathsf{restore}^1(\mathbb{O})\,;\mathsf{restore}^{\geq 0}(\mathbb{O})]\neg\mathrm{IC}? \|$.

In both cases, we have shown that $p_j \in v'$ if and only if $b_j = 1$ for all $j \in \mathcal{I}$.

Let now $g : \mathrm{MSA}_{\mathrm{MSA}}(\boldsymbol{B}) \to V_{v_{\boldsymbol{B}}}^{\mathsf{msa}_{\mathrm{IC}}}$ be the function associating to each $B \in \mathrm{MSA}_{\mathrm{IC}}(\boldsymbol{B})$ the valuation $v' \in V_{v_{\boldsymbol{B}}}^{\mathsf{msa}_{\mathrm{IC}}}$ constructed above. Simple but tedious arguments can be used to show that g is a bijection and thus conclude the proof. □

For a concrete example of an instance of program $\mathsf{msa}_{\mathrm{IC}}$ see Appendix A.

3.3.2 Minimal Number of Atomic Changes Rule

We have seen in Section 2.1 that the Hamming distance $H(B, B')$ between two ballots is the number of issues on which they differ. The minimal number of atomic changes rule returns the ballots satisfying the constraint that result from computing majority on a profile that is minimally distant (in terms of the Hamming distance) from the current profile. The formal definition is the following:

$$\begin{aligned}\mathrm{MNAC}_{\mathrm{IC}}(\boldsymbol{B}) \;=\; \{B \mid \text{there is } \boldsymbol{B}^* \text{ such that } \mathrm{Maj}(\boldsymbol{B}^*) = B, B \models \mathrm{IC},\\ \text{and for all } \boldsymbol{B}' : \sum_{i \in \mathcal{N}} H(B_i, B_i^*) \leq \sum_{i \in \mathcal{N}} H(B_i, B_i')\}.\end{aligned}$$

This rule is also known in the literature as Full_d, for d the Hamming distance (Miller and Osherson, 2009). We find the translation of $\mathrm{MNAC}_{\mathrm{IC}}$ into DL-PA in the next proposition.

Proposition 5. *Let $\mathcal{I}$ be a set of m issues, $\mathcal{N}$ a set of n agents and* IC *a propositional formula.*

The $\mathrm{MNAC}_{\mathrm{IC}}$ *rule is translated into the following* DL-PA *program:*

$$\mathsf{mnac}_{\mathrm{IC}}(\mathbb{B}) := \bigcup_{0 \leq d \leq m \cdot n} \big(\mathsf{H}(\langle \mathsf{prof}_{\mathrm{IC}}(\mathbb{B}, \mathbb{O})\,;\mathsf{maj}(\mathbb{B})\rangle \mathrm{IC}, \mathbb{B}, \geq d)?\,;\mathsf{flip}^1(\mathbb{B})^d\big)\,;$$
$$\mathsf{prof}_{\mathrm{IC}}(\mathbb{B}, \mathbb{O})\,;\mathsf{maj}(\mathbb{B})\,;\mathrm{IC}?.$$

Program $\mathsf{mnac}_{\mathrm{IC}}$ finds the minimal number d of variables in the set $\mathbb{B}$ whose truth values can be flipped such that applying program maj to this new profile leads to a valuation where the outcome satisfies the constraint.

3.4 Preference Aggregation Rules

This section presents rules that have been adapted in judgment aggregation from the literature on preference aggregation (Brams and Fishburn, 2002). The first is the Kemeny rule (Kemeny, 1959), which is also sometimes called *the* distance based rule (Endriss and Grandi, 2014) or *Prototype*$_d$ (Miller and Osherson, 2009). We then present the Slater rule — called *maxcard subagenda rule* by Lang and Slavkovik (2014) and *Endpoint*$_d$ by Miller and Osherson (2009). In both cases, d is the Hamming distance. Finally, we discuss the ranked pairs rule, which also comes under the name of the *ranked agenda rule* (Lang and Slavkovik, 2014).

Similarly to the rules presented in the previous section, translations are not straightforward since we need to encode minimization operations for the Hamming distance and comparisons with the outcome of majority. Since the proofs of correctness of our translations would be in line with that of Proposition 4, they are omitted.

3.4.1 Kemeny Rule

The adaptation of the Kemeny rule to judgment aggregation is a procedure that returns all ballots that satisfy the constraint and that minimize the sum of the Hamming distance to the individual ballots in the profile. The formal definition is as follows:

$$\mathrm{Kemeny}_{\mathrm{IC}}(\boldsymbol{B}) = \operatorname*{argmin}_{B \models \mathrm{IC}} \sum_{i \in \mathcal{N}} H(B, B_i).$$

We first introduce a program that computes the sum of the Hamming distances between the outcome and the profile. Then, the counter dis stores this number.

$$\mathsf{sH}(\mathbb{O}, \mathbb{B}) \;:=\; \mathsf{zero}(\mathsf{dis})\,; \mathop{;}_{i \in \mathcal{N}} \mathop{;}_{j \in \mathcal{I}} \big(\mathsf{if}\ p_j \oplus p_{ij}\ \mathsf{do}\ \mathsf{incr}(\mathsf{dis})\big)$$

We make use of the program sH for translating the $\mathrm{Kemeny}_{\mathrm{IC}}$ rule in the following proposition:

Proposition 6. *Let* $\mathcal{I}$ *be a set of issues,* $\mathcal{N}$ *a set of agents and* IC *a propositional formula. The* $\mathrm{Kemeny}_{\mathrm{IC}}$ *rule is translated into the following* DL-PA *program:*

$$\mathsf{kemeny}_{\mathrm{IC}}(\mathbb{B}) \;:= \mathsf{flip}^{\geq 0}(\mathbb{O})\,;\mathrm{IC}?\,;\mathsf{sH}(\mathbb{O}, \mathbb{B})\,;\mathsf{store}(\mathsf{dis})\,;$$
$$\big(\neg\langle \mathsf{flip}^{\geq 0}(\mathbb{O})\,;\mathrm{IC}?\,;\mathsf{sH}(\mathbb{O}, \mathbb{B})\rangle(\mathsf{dis} < \mathsf{dis}')\big)?$$

Let us give an intuitive breakdown of the program $\mathsf{kemeny}_{\mathrm{IC}}$. Subprogram $\mathsf{flip}^{\geq 0}(\mathbb{O})$ is used to reach an outcome, by nondeterministically choosing the 'right' one. Then, the test IC? checks that the constraint is satisfied, the program $\mathsf{sH}(\mathbb{O}, \mathbb{B})$ adds up the Hamming distances to the individual ballots, and the last line of $\mathsf{kemeny}_{\mathrm{IC}}$ checks that it is not possible to find another outcome that satisfies the constraint and that is even closer to the individual ballots.

3.4.2 Slater Rule

In case the outcome of majority does not satisfy the constraint, the Slater rule outputs those ballots that do satisfy the constraint and that are minimally distant, with respect to the Hamming distance, from the outcome of majority:

$$\mathrm{Slater}_{\mathrm{IC}}(\boldsymbol{B}) = \operatorname*{argmin}_{B \models \mathrm{IC}} H(B, \mathrm{Maj}(\boldsymbol{B})).$$

Proposition 7. *Let $\mathcal{I}$ be a set of m issues, $\mathcal{N}$ a set of agents and* IC *a propositional formula. The* $\mathrm{Slater}_{\mathrm{IC}}$ *rule is translated into the following* DL-PA *program:*

$$\mathsf{slater}_{\mathrm{IC}}(\mathbb{B}) := \mathsf{maj}(\mathbb{B})\,;\ \bigcup_{0 \leq d \leq m} \left(\mathsf{H}(\mathrm{IC}, \mathbb{O}, \geq d)?\,;\mathsf{flip}^1(\mathbb{O})^d\right);\mathrm{IC}?.$$

The program $\mathsf{slater}_{\mathrm{IC}}$ first computes the majority rule, and then it finds the minimal distance d such that by reversing the truth value of d variables in the outcome we reach a valuation where the constraint is satisfied. Observe that $d = 0$ if the majority outcome already satisfies the constraint.

3.4.3 Ranked Pairs Rule

We follow the presentation of the ranked pairs rule given by Endriss and Grandi (2014). The *majority strength* of issue j in profile $\boldsymbol{B}$ is defined as $\mathrm{MS}^{\boldsymbol{B}}(j) = \max\{|N^{\boldsymbol{B}}_{j:0}|, |N^{\boldsymbol{B}}_{j:1}|\}$ and issues can be ordered according to their majority strength with $\succ^{\boldsymbol{B}}_{\tau}$, where $\tau : \mathcal{I} \to \mathcal{I}$ is a permutation that breaks the ties in case two issues have the same majority strength.

For ballot B and partial function $\ell : \mathcal{I} \to \{0, 1\}$, we write $\ell \subseteq B$ if $\ell(j) = b_j$ for every j in the domain of ℓ. Given profile $\boldsymbol{B}$, integrity constraint IC and permutation $\tau : \mathcal{I} \to \mathcal{I}$, we define the total function $\ell^{\boldsymbol{B}}_{\tau,\mathrm{IC}}$ as follows:

$$\textbf{for } j \in \mathcal{I}, \text{ following order } \succ^{\boldsymbol{B}}_{\tau} \textbf{ do}$$

$$\ell^{\boldsymbol{B}}_{\tau,\mathrm{IC}}(j) \ := \begin{cases} \mathrm{Maj}(\boldsymbol{B})_j & \text{if there exists } B \models \mathrm{IC} \text{ such that } \ell^{\boldsymbol{B}}_{\tau,\mathrm{IC}} \subseteq B \\ 1 - \mathrm{Maj}(\boldsymbol{B})_j & \text{otherwise} \end{cases}$$

The ranked pairs rule returns ballots obtained by the above procedure, for a given tie-breaking rule and integrity constraint IC:

$$\mathrm{RP}_{\mathrm{IC}}(\boldsymbol{B}) \ := \ \{\ell^{\boldsymbol{B}}_{\tau,\mathrm{IC}} \mid \tau \text{ is a permutation on } \mathcal{I}\}.$$

Example 2. *Let $\mathcal{N} = \{1, 2, 3\}$ and $\mathcal{I} = \{1, 2, 3, 4\}$ with* $\mathrm{IC} := (p_1 \leftrightarrow p_2) \vee (p_3 \leftrightarrow p_4)$. *Consider the profile $\boldsymbol{B}$ displayed in the following table:*

	1	2	3	4
Agent 1	*1*	*1*	*1*	*0*
Agent 2	*1*	*0*	*1*	*1*
Agent 3	*1*	*0*	*0*	*0*
Majority	*1*	*0*	*1*	*0*

We have that $\mathrm{MS}^{\boldsymbol{B}}(1) = 3$, $\mathrm{MS}^{\boldsymbol{B}}(2) = 2$, $\mathrm{MS}^{\boldsymbol{B}}(3) = 2$ *and* $\mathrm{MS}^{\boldsymbol{B}}(4) = 2$ *are the majority strengths of the issues. Suppose that the tie-breaking function is the following:* $\tau(1) = 4$, $\tau(2) = 3$, $\tau(3) = 2$ *and* $\tau(4) = 1$. *We thus have that* $1 \succ_{\tau}^{\boldsymbol{B}} 4 \succ_{\tau}^{\boldsymbol{B}} 3 \succ_{\tau}^{\boldsymbol{B}} 2$ *is the ordering of the issues according to their majority strength in* $\boldsymbol{B}$ *and the tie-breaking function. Since* $\ell^{\boldsymbol{B}}_{\tau,\mathrm{IC}}(1) = 1$, $\ell^{\boldsymbol{B}}_{\tau,\mathrm{IC}}(4) = 0$, $\ell^{\boldsymbol{B}}_{\tau,\mathrm{IC}}(3) = 1$ *and* $\ell^{\boldsymbol{B}}_{\tau,\mathrm{IC}}(2) = 1$, *we have* $\mathrm{RP}_{\mathrm{IC}}(\boldsymbol{B}) = (1110)$ *for this choice of* τ.

The DL-PA program we present as a translation of the $\mathrm{RP}_{\mathrm{IC}}$ rule is for the special case of no ties in the majority strengths of the issues. The general case can be dealt with by introducing a sub-program ordering the issues according to the tie-breaking rule.

First, we define a program that calculates the majority strength of the issues, assuming the majority outcome has already been computed:

$$\mathsf{majSt}(\mathbb{B}) := \mathop{;}_{j\in\mathcal{I}} \big(\mathsf{zero}(\mathsf{ms}_j)\,;\mathsf{if}\ p_j\ \mathsf{then}\ \big(\mathop{;}_{i\in\mathcal{N}} \mathsf{if}\ p_{ij}\ \mathsf{do}\ \mathsf{incr}(\mathsf{ms}_j)\big)\ \mathsf{else}\ \big(\mathop{;}_{i\in\mathcal{N}} \mathsf{if}\ \neg p_{ij}\ \mathsf{do}\ \mathsf{incr}(\mathsf{ms}_j)\big)\big).$$

The m sets defined below allow us to add and consider one issue at a time in the order of their majority strengths:

$$Q_1 := \{p_j \mid |\mathrm{MS}^{\boldsymbol{B}}_j| \geq |\mathrm{MS}^{\boldsymbol{B}}_l| \text{ for all } l \in \mathcal{I}\}$$
$$Q_k := Q_{k-1} \cup \{p_j \mid p_j \notin Q_{k-1} \text{ and for all } l \in \mathcal{I} \text{ such that } p_l \notin Q_{k-1},\ |\mathrm{MS}^{\boldsymbol{B}}_j| \geq |\mathrm{MS}^{\boldsymbol{B}}_l|\}.$$

Proposition 8. *Let* $\mathcal{I}$ *be a set of* m *issues,* $\mathcal{N}$ *a set of* n *agents and* IC *a propositional formula. The* $\mathrm{RP}_{\mathrm{IC}}$ *rule is translated into the following* DL-PA *program:*

$$\mathsf{rp}_{\mathrm{IC}}(\mathbb{B}) := \mathsf{maj}(\mathbb{B})\,;\mathsf{if}\ \neg\mathrm{IC}\ \mathsf{do}\ \big(\mathsf{majSt}(\mathbb{B})\,;\mathop{;}_{1\leq i\leq m} \big(\bigcup_{j\in\mathcal{I}}\big(\bigwedge_{l\in\mathcal{I}} \mathsf{ms}_j \geq \mathsf{ms}_l?;$$
$$\mathsf{if}\ \neg\langle\mathsf{flip}^{\geq 0}(\mathbb{O}\setminus Q_i)\rangle\mathrm{IC}\ \mathsf{do}\ p_j \leftarrow \neg p_j\,;\mathsf{zero}(\mathsf{ms}_j)\big)\big)\big).$$

The program $\mathsf{rp}_{\mathrm{IC}}$ computes the majority, and if the constraint is not satisfied it calculates the majority strength of the issues. Then, this procedure is applied m times: first, the issue j with highest majority strength at that stage is selected. The program checks whether there is a way to modify the outcome (without changing the truth value of p_j *and* that of the issues already inspected at some previous step) to satisfy the constraint. If not, the truth value of p_j is flipped — and the value of its majority strength is set to zero. At the following step, the nondeterministic choice operator selects the next issue in the ranking; this is possible because the issue has now highest majority strength, since the one of j has been set to zero.

3.5 Summary and Discussion

We provided a DL-PA translation for many known aggregation rules, starting from simple examples such as dictatorship and quota rules, and then moving to more complex ones based on minimization, maximization and on preference aggregation rules. Observe that the latter type of rules can be easily adapted to those selecting the *most representative voter* in a given profile, according to different principles (Endriss and Grandi, 2014). Programs in DL-PA translating most representative voter rules are derivable from those of Kemeny, Slater and ranked pairs by substituting every occurrence of the formula IC by the formula $\mathsf{Repr} := \bigvee_{i\in\mathcal{N}} \big(\bigwedge_{j\in\mathcal{I}} p_{ij} \leftrightarrow p_j \big)$.

Beyond the theoretical interest of exploring the power of DL-PA, the intended application for such a translation is to the *winner determination* problem for judgment aggregation rules (see, e.g., Endriss et al., 2012; Lang and Slavkovik, 2014). By model checking DL-PA specifications we can compute the outcome of a rule on a given profile. For a resolute aggregation rule F, the problem is usually formulated as checking for each issue j whether $F(\boldsymbol{B})_j = 1$ for profile $\boldsymbol{B}$, and this translates in DL-PA into simply checking whether $v_{\boldsymbol{B}} \models [\mathsf{f}(\mathbb{B})]p_j$ holds.

4 Formalization of Axiomatic Properties

Aggregation rules are usually studied and characterized according to which desirable properties, or *axioms*, they satisfy (Dietrich and List, 2007b; Grandi and Endriss, 2011; List, 2012; May, 1952). Following a similar distinction in preference aggregation between *intra-profile* and *inter-profile* conditions (Rubinstein, 1984), we separate *single-profile* and *multi-profile* axioms. While the former relate a profile with the outcome of a rule applied on that profile, the latter link two profiles with the outcomes of the same aggregation rule applied on them. The former are translated into propositional logic, while the latter are translated into DL-PA.

4.1 Single-profile Axioms

For the four single-profile axioms that we present, the full DL-PA machinery is not necessary since we can provide an easy and direct translation into propositional logic. We first introduce the definition in judgment aggregation, and then proceed to translate them. In Theorem 2 we will use the dynamic component of DL-PA to prove the correctness of our translations.

A rule F is *unanimous* if in case all agents agree on some issue j, the outcome of F for issue j agrees with them. Formally:

U : For all $\boldsymbol{B}$, for all $j \in \mathcal{I}$ and for $x \in \{0,1\}$, if for all $i \in \mathcal{N}$ $b_{ij} = x$ then $F(\boldsymbol{B})_j = x$.

A rule is *neutral with respect to the issues* if, when two issues are treated in the same way in the input, they are treated in the same way in the output.

$\mathrm{N}^{\mathcal{I}}$: For all $\boldsymbol{B}$ and any two $j, k \in \mathcal{I}$, if for all $i \in \mathcal{N}$ $b_{ij} = b_{ik}$ then $F(\boldsymbol{B})_j = F(\boldsymbol{B})_k$.

A rule is *neutral with respect to the domain* if, whenever two issues are treated in an opposite way in the input, their output is opposite.

$\mathrm{N}^{\mathcal{D}}$: For all $\boldsymbol{B}$ and any $j, k \in \mathcal{I}$, if for all $i \in \mathcal{N}$ $b_{ij} = 1 - b_{ik}$ then $F(\boldsymbol{B})_j = 1 - F(\boldsymbol{B})_k$.

A rule is *neutral-monotonic*[4] if the acceptance of an issue j in a given profile implies the acceptance of any other issue k which is accepted by a strict superset of individuals.

$\mathrm{M^N}$: For all $\boldsymbol{B}$ and any $j, k \in \mathcal{I}$, if for all $i \in \mathcal{N}$ $b_{ij} = 1$ implies $b_{ik} = 1$, and there is $s \in \mathcal{N}$ such that $b_{sj} = 0$ and $b_{sk} = 1$, then $F(\boldsymbol{B})_j = 1$ implies $F(\boldsymbol{B})_k = 1$.

We are now ready to prove the following result:

Theorem 2. *Let $\mathbb{B}$ be a set of variables for agents in $\mathcal{N}$ and issues in $\mathcal{I}$, let F be an aggregator for n and m, and let f be its* DL-PA *translation. Moreover, let:*

$$\begin{aligned}
\mathsf{U} &:= \bigwedge_{j\in\mathcal{I}} \big(((\bigwedge_{i\in\mathcal{N}} p_{ij}) \to p_j) \wedge ((\bigwedge_{i\in\mathcal{N}} \neg p_{ij}) \to \neg p_j)\big),\\
\mathsf{N}^{\mathcal{I}} &:= \bigwedge_{j\in\mathcal{I}} \bigwedge_{k\in\mathcal{I}} \big((\bigwedge_{i\in\mathcal{N}} (p_{ij} \leftrightarrow p_{ik})) \to (p_j \leftrightarrow p_k)\big),\\
\mathsf{N}^{\mathcal{D}} &:= \bigwedge_{j\in\mathcal{I}} \bigwedge_{k\in\mathcal{I}} \big((\bigwedge_{i\in\mathcal{N}} (p_{ij} \leftrightarrow \neg p_{ik})) \to (p_j \leftrightarrow \neg p_k)\big),\\
\mathsf{M}^{\mathsf{N}} &:= \bigwedge_{j\in\mathcal{I}} \bigwedge_{k\in\mathcal{I}} \big((\bigwedge_{i\in\mathcal{N}} (p_{ij} \to p_{ik}) \wedge \bigvee_{s\in\mathcal{N}} (\neg p_{sj} \wedge p_{sk})) \to (p_j \to p_k)\big).
\end{aligned}$$

Then, the following equivalences hold:

(i) F satisfies U *if and only if* $\models \mathsf{Prof}_{\mathrm{IC}}(\mathbb{B}, \mathbb{O}) \to [\mathsf{f}(\mathbb{B})]\mathsf{U}$.

(ii) F satisfies $\mathrm{N}^{\mathcal{I}}$ *if and only if* $\models \mathsf{Prof}_{\mathrm{IC}}(\mathbb{B}, \mathbb{O}) \to [\mathsf{f}(\mathbb{B})]\mathsf{N}^{\mathcal{I}}$.

(iii) F satisfies $\mathrm{N}^{\mathcal{D}}$ *if and only if* $\models \mathsf{Prof}_{\mathrm{IC}}(\mathbb{B}, \mathbb{O}) \to [\mathsf{f}(\mathbb{B})]\mathsf{N}^{\mathcal{D}}$.

(iv) F satisfies $\mathrm{M^N}$ *if and only if* $\models \mathsf{Prof}_{\mathrm{IC}}(\mathbb{B}, \mathbb{O}) \to [\mathsf{f}(\mathbb{B})]\mathsf{M}^{\mathsf{N}}$.

Proof. We provide a proof for (i), since the remaining parts can be proven in an analogous way. For the left-to-right direction, consider an arbitrary valuation v such that $v \models \mathsf{Prof}_{\mathrm{IC}}(\mathbb{B}, \mathbb{O})$ and suppose for reductio that there is some v' such that $(v, v') \in \| \mathsf{f}(\mathbb{B}) \|$ but $v' \notin \| \mathsf{U} \|$. Given the definition of the interpretation, this means that there is some $j \in \mathcal{I}$ such that $v' \notin \| ((\bigwedge_{i\in\mathcal{N}} p_{ij}) \to p_j) \wedge ((\bigwedge_{i\in\mathcal{N}} \neg p_{ij}) \to \neg p_j) \|$. Assume, without loss of generality, that $v' \notin \| (\bigwedge_{i\in\mathcal{N}} p_{ij}) \to p_j) \|$. Hence, we have that $v' \in \| \bigwedge_{i\in\mathcal{N}} p_{ij} \|$ and $v' \notin \| p_j \|$. Since $v \models \mathsf{Prof}_{\mathrm{IC}}(\mathbb{B}, \mathbb{O})$ and $(v, v') \in \| \mathsf{f}(\mathbb{B}) \|$, we have that v corresponds to some profile $\boldsymbol{B}$ and v' corresponds to $F(\boldsymbol{B})$.

By assumption, F satisfies U: in particular, this means that for all $j \in \mathcal{I}$, if for all $i \in \mathcal{N}$ we have $b_{ij} = 1$, then $F(\boldsymbol{B})_j = 1$. Observe that by the DL-PA programs we provided to translate aggregation rules, v and v' do not differ on the variables in $\mathbb{B}$. Hence, the fact that $v' \in \| \bigwedge_{i\in\mathcal{N}} p_{ij} \|$ implies that $v \in \| \bigwedge_{i\in\mathcal{N}} p_{ij} \|$. Therefore, in profile $\boldsymbol{B}$ we have $b_{ij} = 1$ for all $i \in \mathcal{N}$, which implies that $F(\boldsymbol{B})_j = 1$. This contradicts the fact that v' corresponds to $F(\boldsymbol{B})$ and that $v' \notin \| p_j \|$. Hence, $v' \in \| \mathsf{U} \|$.

For the right-to-left direction, take an arbitrary profile $\boldsymbol{B}$. Suppose, for reductio, that for some $j \in \mathcal{I}$ in profile $\boldsymbol{B}$, we have $b_{ij} = 1$ for all $i \in \mathcal{N}$ and $F(\boldsymbol{B})_j = 0$. Consider now valuations v and v' corresponding to $\boldsymbol{B}$ and $F(\boldsymbol{B})$ respectively. This means that $v \models \mathsf{Prof}_{\mathrm{IC}}(\mathbb{B}, \mathbb{O})$, that

[4]This axiom is non-standard, and has been introduced by Endriss et al. (2012).

$(v, v') \in \| \mathsf{f}(\mathbb{B}) \|$ and that $v' \in \| \mathsf{U} \|$. Since v and v' do not differ on the variables in $\mathbb{B}$, by spelling out the definition of the interpretation we get that $v' \in \| p_j \|$. This contradicts the fact that v' corresponds to $F(\boldsymbol{B})$ and that $F(\mathbf{B})_j = 0$. Therefore, we have that $F(\boldsymbol{B}) = 1$. □

4.2 Multi-profile Axioms

The three multi-profile axioms we present are translated as DL-PA formulas. In fact, to check whether an aggregation rule satisfies these axioms, we need to compare the outcomes of the rule on different profiles. Dealing with multiple profiles means referring to more than one valuation (and applying the program expressing rule F more than once), whence the need for DL-PA.

A rule is *independent* if, whenever an issue j has the same acceptance-rejection pattern in two profiles, the outcome of the rule is identical for j in both of them. Formally:

I : For any $j \in \mathcal{I}$ and profiles $\boldsymbol{B}$ and $\boldsymbol{B}'$, if for all $i \in \mathcal{N}$ $b_{ij} = b'_{ij}$ then $F(\boldsymbol{B})_j = F(\boldsymbol{B}')_j$.

A rule F is *independent-monotonic* if, whenever we consider two profiles such that the second differs from the first in that some agent i first rejected issue j and then she accepts it, if j was accepted in the first outcome then it should still be accepted in the second. Let $(\boldsymbol{B}_{-i}, B'_i) = (B_1, \ldots, B'_i, \ldots, B_n)$ for some profile $\boldsymbol{B}$:

M^{I} : For any issue $j \in \mathcal{I}$, agent $i \in \mathcal{N}$, profiles $\boldsymbol{B} = (B_1, \ldots, B_n)$ and $\boldsymbol{B}' = (\boldsymbol{B}_{-i}, B'_i)$, if $b_{ij} = 0$ and $b'_{ij} = 1$ then $F(\boldsymbol{B})_j = 1$ implies $F(\boldsymbol{B}')_j = 1$.

An *anonymous* rule treats each agent in the same way. That is, by permuting the order of the individual ballots in the input, the output for all issues does not change.

A : For all $\mathbf{B}$ and any permutation $\sigma : \mathcal{N} \to \mathcal{N}$, $F(B_1, \ldots, B_n) = F(B_{\sigma(1)}, \ldots, B_{\sigma(n)})$.

We are now ready to prove the following:

Theorem 3. *Let $\mathbb{B}$ be the set of variables for agents in $\mathcal{N}$ and issues in $\mathcal{I}$, let F be an aggregation rule for n and m, and let f be its* DL-PA *translation. Moreover, for $\mathbb{B}_j := \{p_{ij} \mid i \in \mathcal{N}\}$ let:*

$$
\begin{aligned}
\mathsf{I} := & \bigwedge_{j \in \mathcal{I}} \big((p_j \to [\mathsf{flip}^{\geq 0}(\mathbb{B} \setminus \mathbb{B}_j) \,;\, \mathsf{prof}_{\mathrm{IC}}(\mathbb{B}, \mathbb{O}) \,;\, \mathsf{f}(\mathbb{B})] p_j) \wedge \\
& (\neg p_j \to [\mathsf{flip}^{\geq 0}(\mathbb{B} \setminus \mathbb{B}_j) \,;\, \mathsf{prof}_{\mathrm{IC}}(\mathbb{B}, \mathbb{O}) \,;\, \mathsf{f}(\mathbb{B})] \neg p_j) \big), \\
\mathsf{M}^{\mathsf{I}} := & \bigwedge_{j \in \mathcal{I}} \big(p_j \to \bigwedge_{i \in \mathcal{N}} [+p_{ij} \,;\, \mathsf{prof}_{\mathrm{IC}}(\mathbb{B}, \mathbb{O}) \,;\, \mathsf{f}(\mathbb{B})] p_j \big), \\
\mathsf{A} := & [\mathsf{store}(\mathbb{O}) \,;\, \big(\bigcup_{i,k \in \mathcal{N}} \,;_{j \in \mathcal{I}} \big(\mathsf{if}\ p_{ij} \oplus p_{kj}\ \mathsf{do}\ (\mathsf{flip}^1(\{p_{ij}\}) \,;\, \mathsf{flip}^1(\{p_{kj}\})) \big) \big)^{n-1} \,; \\
& \mathsf{zero}(\mathbb{O}) \,;\, \mathsf{f}(\mathbb{B})] \bigwedge_{j \in \mathcal{I}} (p_j \leftrightarrow p'_j).
\end{aligned}
$$

Then, the following is the case:

(i) F satisfies I *if and only if* $\models \mathsf{Prof}_{\mathrm{IC}}(\mathbb{B}, \mathbb{O}) \to [\mathsf{f}(\mathbb{B})]\mathsf{I}$.

(ii) F satisfies M^{I} *if and only if* $\models \mathsf{Prof}_{\mathrm{IC}}(\mathbb{B}, \mathbb{O}) \to [\mathsf{f}(\mathbb{B})]\mathsf{M}^{\mathsf{I}}$.

(iii) F satisfies A *if and only if* $\models \mathsf{Prof}_{\mathrm{IC}}(\mathbb{B},\mathbb{O}) \rightarrow [\mathsf{f}(\mathbb{B})]\mathsf{A}$.

Proof. The proof for (i) and (ii) being very similar, we just give a proof for (ii). For the left-to-right direction, consider v_1 such that $v_1 \models \mathsf{Prof}_{\mathrm{IC}}(\mathbb{B},\mathbb{O})$ and suppose for reductio that there is some valuation v_2 such that $(v_1, v_2) \in \| \mathsf{f}(\mathbb{B}) \|$ but for some issue $k \in \mathcal{I}$ we have (a) $v_2 \in \| p_k \|$, and (b) $v_2 \notin \| \bigwedge_{i\in\mathcal{N}}[+p_{ik}\,;\mathsf{prof}_{\mathrm{IC}}(\mathbb{B},\mathbb{O})\,;\mathsf{f}(\mathbb{B})]p_k \|$. The latter implies that there is some agent $s \in \mathcal{N}$ such that (c) there is v_3 where $(v_2, v_3) \in \| +p_{sk}\,;\mathsf{prof}_{\mathrm{IC}}(\mathbb{B},\mathbb{O})\,;\mathsf{f}(\mathbb{B}) \|$, and (d) $v_3 \notin \| p_k \|$. Given that $(v_2, v_3) \in \| +p_{sk}\,;\mathsf{prof}_{\mathrm{IC}}(\mathbb{B},\mathbb{O});\mathsf{f}(\mathbb{B}) \|$, then there is some valuation v_a such that $(v_2, v_a) \in \| +p_{sk} \|$ and $(v_a, v_3) \in \| \mathsf{prof}_{\mathrm{IC}}(\mathbb{B},\mathbb{O})\,;\mathsf{f}(\mathbb{B}) \|$. Hence, v_a corresponds to a profile $\boldsymbol{B}^a$ and v_3 corresponds to $F(\boldsymbol{B}^a)$.

Consider now valuations v_1, v_2 and v_a. While v_1 and v_2 do not differ on the variables in $\mathbb{B}$, v_2 and v_a possibly differ on p_{sk}. We now focus on the interesting case: namely, the one where they do differ. If they differ, we have that $p_{sk} \notin v_1$ and $p_{sk} \notin v_2$ but $p_{sk} \in v_a$. Note that since $v_1 \models \mathsf{Prof}_{\mathrm{IC}}(\mathbb{B},\mathbb{O})$ and $(v_1, v_2) \in \| \mathsf{f}(\mathbb{B}) \|$, this means that v_1 corresponds to a profile $\boldsymbol{B}^1$ and v_2 corresponds to $F(\boldsymbol{B}^1)$. Since $b^1_{sk} = 0$ and by (b) we have $F(\boldsymbol{B}^1)_j = 1$, and since by assumption F satisfies M^{I}, and since $\boldsymbol{B}^a$ differs from $\boldsymbol{B}^1$ only on b_{sk}, because $b^a_{sk} = 1$, we must have that $F(\boldsymbol{B}^a)_k = 1$. But this contradicts the fact that by (d) we have $p_k \notin v_3$.

For the right-to-left direction, consider an arbitrary profile $\boldsymbol{B}$. Suppose, for reductio, that there is some issue $j \in \mathcal{I}$, some agent $i \in \mathcal{N}$ and profile $\boldsymbol{B}'$, such that we have $b_{ij} = 0$, $b'_{ij} = 1$, $F(\boldsymbol{B})_j = 0$ and $F(\boldsymbol{B}')_j = 0$. Consider now valuation v_1 corresponding to profile $\boldsymbol{B}$ and valuation v_2 corresponding to $F(\boldsymbol{B})$. Moreover, consider valuation v_a corresponding to profile $\boldsymbol{B}'$ and valuation v_b corresponding to $F(\boldsymbol{B}')$. Since by assumption F satisfies M^{I}, and $p_j \in v_2$, and v_a only differs from v_1 in that $p_{ij} \in v_a$, we see that from the axiom we should get $p_j \in v_b$. This contradicts the fact that v_b is a translation of $F(\boldsymbol{B}')$ and that $F(\boldsymbol{B}')_j = 0$.

To obtain a proof for (iii), recall that it is possible to generate all the $n!$ permutations of n agents by repeatedly swapping the positions of just two agents with at most $n-1$ swaps (Wells, 1961). Note also that the translation of the Anonymity axiom first stores the result of the aggregation rule in some fresh variables p'_j, then for $n-1$ times it nondeterministically selects two agents and swaps their individual ballots by inverting the truth values of the corresponding variables. Finally, it executes again the rule F and it checks that the two results coincide. □

4.3 Summary and Discussion

We have translated judgment aggregation axioms into formulas of either propositional logic or DL-PA. More precisely, we have used propositional logic for single-profile axioms, and DL-PA for multi-profile axioms, since these are the most suited formalisms for the corresponding cases, as explained in Sections 4.1 and 4.2.

The possible applications of such a translation of axioms for aggregation rules are twofold. On the one hand, a negative answer on the model checking for formula $[\mathsf{f}(\mathbb{B})]\mathsf{axiom}$ tells us that rule F does not satisfy the axiom whose DL-PA translation is axiom. On the other hand, while we cannot take a positive answer to this model checking as a definite answer that the aggregation rule (considered for all n and m) does satisfy an axiom, we can still use such a piece of information to gather confidence that the rule indeed satisfies the axiom.

5 Safety of the Agenda

A recurring problem in judgment aggregation is that the outcome of an aggregation process might not satisfy the integrity constraint even though each agent satisfies it in the submitted ballot. One way out of this problem is to investigate whether we can ensure that the outcome of certain groups of aggregation rules will always satisfy a given constraint, provided it has a particular syntactical form. This approach is known in the judgment aggregation literature under the name of *safety of the agenda* (Endriss et al., 2012). In this section we express known agenda properties within the logical formalism of our choice, something that has never been explored in the literature connecting judgment aggregation to logic.

5.1 Safety and Prime Implicants

We begin by introducing *prime implicants* to express basic concepts of the agenda safety problem. For a start, let a *literal* be either a variable p or its negation $\neg p$. A *term* D is a conjunction of distinct literals, and $D - D'$ returns all the literals of D that are not in D'. A term D is an *implicant* of φ if and only if $D \models \varphi$. We present now the formal definition of prime implicants as given by Marchi et al. (2010):

Definition 4. *D is a* prime implicant *of φ if and only if (i) D is an implicant of φ, and (ii) for all literals L in D, $(D - \{L\}) \not\models \varphi$.*

Every formula φ can be rewritten as a conjunction of negations of the prime implicants of $\neg\varphi$ (Marquis, 2000). Since in particular this holds for IC, we assume in the following that integrity constraints are written in this syntactical form.

We now reinterpret some known *agenda properties* of formula-based judgment aggregation for integrity constraints, making use of the concept of prime implicants. Let $\mathbb{P}_\varphi$ be the set of variables occurring in φ.

Definition 5. *A constraint* IC *has the* k-median property (kMP) *if and only if any prime implicant D of* $\neg$IC *is such that $|\mathbb{P}_D| \leq k$. A constraint* IC *has the* simplified median property (SMP) *if and only if any prime implicant D of* $\neg$IC *is such that $|\mathbb{P}_D| = 2$ and for $p, q \in \mathbb{P}_D$ we have that $\neg L_p \wedge \neg L_q$ is also a prime implicant of* $\neg$IC.

For $k = 2$ we speak of the *median-property* (MP). Observe that if IC $= \top$ we do not have any prime implicant of $\neg$IC, which means that the issues are all independent from one another — a condition known as *syntactic simplified median property* (SSMP) in the literature.

We conclude this section by providing the definition of safety for integrity constraints. Given a set of axioms AX, we call the set $\mathcal{F}_{\mathrm{IC}}[\mathrm{AX}] := \{F \mid F$ satisfies all axioms in AX and the domain of F is $\mathrm{Mod(IC)}^{\mathcal{N}}$ for some $\mathcal{N}\}$ a *class of aggregation procedures*. Then:

Definition 6. *An integrity constraint* IC *is* safe for the class $\mathcal{F}_{\mathrm{IC}}[\mathrm{AX}]$ *if and only if for all $F \in \mathcal{F}_{\mathrm{IC}}[\mathrm{AX}]$, we have $F(\boldsymbol{B}) \models \mathrm{IC}$ for all inputs $\boldsymbol{B} \in \mathrm{Mod(IC)}^{\mathcal{N}}$.*

The intuitive meaning is that when an integrity constraint is safe for a group of aggregation rules that all satisfy some axioms, on every input profile the output will satisfy the constraint.

5.2 Agenda Safety in DL-PA

Our first result is a Lemma which characterizes by a DL-PA formula the valuations where some prime implicant of a formula φ is true.

Lemma 1. *Let D be a term such that $\mathbb{P}_D \subseteq \mathbb{P}_\varphi$. Then, D is a prime implicant of φ if and only if $D \models \mathsf{PI}(\mathbb{P}_D, \varphi)$, where $\mathsf{PI}(\mathbb{P}_D, \varphi) := [\mathsf{flip}^{\geq 0}(\mathbb{P}_\varphi \setminus \mathbb{P}_D)]\varphi \wedge [\mathsf{flip}^1(\mathbb{P}_D)]\neg[\mathsf{flip}^{\geq 0}(\mathbb{P}_\varphi \setminus \mathbb{P}_D)]\varphi$.*

Proof. For the left-to-right direction, let D be a prime implicant of φ and suppose, for reductio, that there is some valuation making D true such that $\neg\mathsf{PI}(\mathbb{P}_D, \varphi)$ holds. Observe that if we have $\langle\mathsf{flip}^{\geq 0}(\mathbb{P}_\varphi \setminus \mathbb{P}_D)\rangle\neg\varphi$ we would have a contradiction with condition (i) of Definition 4 (D is not an implicant of φ). In fact, while the literals in D are true at the current valuation $\neg\varphi$ holds. On the other hand, if $\langle\mathsf{flip}^1(\mathbb{P}_D)\rangle[\mathsf{flip}^{\geq 0}(\mathbb{P}_\varphi \setminus \mathbb{P}_D)]\varphi$ is the case, we would have a contradiction with condition (ii) of Definition 4 (D is not prime). In fact, some variable $p_k \in \mathbb{P}_D$ corresponding to a literal L_k in D would make $(D - \{L_k\}) \models \varphi$ hold. Therefore, we have $D \models \mathsf{PI}(\mathbb{P}_D, \varphi)$.

We prove the right-to-left direction by contraposition. Suppose D is not a prime implicant of φ. By Definition 4 this means that either D is not an implicant of φ, which would imply that there is some valuation where D and $\langle\mathsf{flip}^{\geq 0}(\mathbb{P}_\varphi \setminus \mathbb{P}_D)\rangle\neg\varphi$ hold; or that D is not prime, which would imply that in some valuation D holds and also $\langle\mathsf{flip}^1(\mathbb{P}_D)\rangle[\mathsf{flip}^{\geq 0}(\mathbb{P}_\varphi \setminus \mathbb{P}_D)]\varphi$. Thus, in both cases we conclude that there is a valuation where D holds and yet $\mathsf{PI}(\mathbb{P}_D, \varphi)$ does not. □

Lemma 1 allows us to characterize the kMP in the following proposition. A similar result on the SMP follows immediately.

Proposition 9. *Constraint* IC *has the* kMP *if and only if* $\models \neg\mathrm{IC} \to \bigvee_{\substack{P \subseteq \mathbb{P}_{\mathrm{IC}} \\ |P| \leq k}} \mathsf{PI}(P, \neg\mathrm{IC})$.

Proof. For the left-to-right direction, assume that IC has the kMP and suppose, for reductio, that there is some v such that $v \models \neg\mathrm{IC}$ and $v \models \bigwedge_{\substack{P \subseteq \mathbb{P}_{\mathrm{IC}} \\ |P| \leq k}} \neg\mathsf{PI}(P, \neg\mathrm{IC})$. Since $v \not\models \mathrm{IC}$ and IC is written as a conjunction of negations of prime implicants of $\neg\mathrm{IC}$, we know that there must be some prime implicant D of $\neg\mathrm{IC}$ such that $v \models D$ and that $|\mathbb{P}_D| \leq k$. By Lemma 1 we thus get that $v \models \mathsf{PI}(\mathbb{P}_D, \neg\mathrm{IC})$, which contradicts $v \models \bigwedge_{\substack{P \subseteq \mathbb{P}_{\mathrm{IC}} \\ |P| \leq k}} \neg\mathsf{PI}(P, \neg\mathrm{IC})$.

We prove the right-to-left direction by contraposition. Suppose IC does not have the kMP: hence, there is some prime implicant D of $\neg\mathrm{IC}$ such that $|\mathbb{P}_D| \geq k + 1$. We now provide a valuation v such that $v \models \neg\mathrm{IC}$ and $v \not\models \bigvee_{\substack{P \subseteq \mathbb{P}_{\mathrm{IC}} \\ |P| \leq k}} \mathsf{PI}(P, \neg\mathrm{IC})$. Consider valuation v such that $v \models D$ and for all other prime implicants D' of $\neg\mathrm{IC}$, we have $v \not\models D'$ (such a valuation always exists). Since $v \models D$, we get by Definition 4 that $v \models \neg\mathrm{IC}$. Suppose there was some other term D' such that $v \models \mathsf{PI}(\mathbb{P}_{D'}, \neg\mathrm{IC})$, $|D'| \leq k$ and $v \models D'$: by Lemma 1 this would imply that D' is a prime implicant of $\neg\mathrm{IC}$, contradicting our choice of valuation. □

Proposition 10. *Constraint* IC *has the* SMP *if and only if*

$$\models \neg\mathrm{IC} \to \bigvee_{p_i, p_k \in \mathbb{P}_{\mathrm{IC}}} \big(\mathsf{PI}(\{p_i, p_k\}, \neg\mathrm{IC}) \wedge [\mathsf{flip}^1(p_i); \mathsf{flip}^1(p_k)]\mathsf{PI}(\{p_i, p_k\}, \neg\mathrm{IC})\big).$$

Proof. For the left-to-right direction, assume that IC has the SMP and consider an arbitrary valuation v where $v \models \neg\mathrm{IC}$. For reductio, suppose the consequent does not hold. Hence, either

there is no prime implicant of $\neg$IC of size 2 or there is one, but the negation of its literals is not a prime implicant of $\neg$IC. Either way, this contradicts the assumption that IC has the SMP.

For the other direction, assume that IC does not have the SMP. This means that either it has not the MP, or it has the MP but there is a prime implicant of $\neg$IC such that its negated literals are not also a prime implicant of $\neg$IC. In the first case, we would get by Proposition 9 that there is a valuation v such that $v \not\models \mathsf{PI}(\{p_i, p_k\}, \neg\mathrm{IC})$ thus falsifying the consequent. In the second case, we would have $v \not\models [\mathsf{flip}^1(p_i); \mathsf{flip}^1(p_k)]\mathsf{PI}(\{p_i, p_k\}, \neg\mathrm{IC})$, thus falsifying the consequent again. Therefore, $\not\models \bigvee_{p_i,p_k \in \mathbb{P}_{\mathrm{IC}}} \big(\mathsf{PI}(\{p_i, p_k\}, \neg\mathrm{IC}) \wedge [\mathsf{flip}^1(p_i); \mathsf{flip}^1(p_k)]\mathsf{PI}(\{p_i, p_k\}, \neg\mathrm{IC})\big)$. □

Observe that in the consequent of the formula characterizing the kMP we have a disjunction over all subsets of size k of variables of the constraint. This implies that the formula has length exponential in the size of the constraint. While this appears as a negative result it is in line with the literature, since the computational problem of deciding whether a constraint satisfies one of the previous agenda properties is at the second level of the polynomial hierarchy.[5]

6 Discussion and Related Work

In this section we compare our work with the literature on formalizations of judgment aggregation and social choice theory in logical languages. We discuss whether and how aggregation rules, axioms and the agenda safety problem have been treated by other researchers. We then discuss the significance of our results in the formalization of characterization and impossibility theorems from social choice, and we discuss potential applications of automated reasoning techniques.

6.1 Related Work on Logic and Judgment Aggregation

In this paper we thoroughly explored the possibility to translate judgment aggregation into DL-PA, providing both general and specific results for a number of aggregation rules, axioms, and safety of the agenda properties. A translation of the issue-wise majority rule has been proposed by Ågotnes et al. (2011) for Judgment Aggregation Logic, with the goal to express a variant of the discursive dilemma. The formula they use, however, is of exponential size in the number of agents, since they explicitly list all possible majoritarian coalitions in the profile (i.e., sets of agents whose size is more than half the total number of agents). Our formalization in Section 3.2.2 is more compact thanks to the use of counters.

The approach taken by Pauly (2007) to study aggregation rules in formula-based judgment aggregation is quite different from ours. The author focuses on three rules: the majority rule, the consensus rule (i.e., a rule that accepts a formula if and only if all agents accept it) and the dictatorship of agent i. Given an agenda and a group of agents, the author defines sets of collective judgment sets corresponding to the outcome of a specific aggregation rule (among the ones mentioned before) and he provides an axiomatization for these outcome sets. Pauly's approach is non standard since, as we shall see in the following section, when characterizing an aggregation rule we do not usually refer to the structure of the set of its possible outcomes, but on the properties satisfied by the rule on every input.

[5] See the Π_2^p-completeness results by Endriss et al. (2012).

As far as axioms are concerned, logical expressions for a small number of them can be found in Judgment Aggregation Logic (Ågotnes et al., 2011). Given that the goal of the authors is to express the Condorcet's voting paradox as a formula of the logic, they only focus on positive unanimity, independence and non-dictatorship (i.e., an axiom satisfied by all rules that are not a dictatorship of some agent i). Moreover, a translation of the preference aggregation axioms for strong monotonicity, independence of irrelevant alternatives and dictatorship are expressed in the Logic for Social Choice Functions by Ciná and Endriss (2016).

To the best of our knowledge, the only paper that extensively addresses the problem of the safety of the agenda is that of Porello (2017). However, the aim of the author's investigations there is to study whether the safety results we presented in Section 5 continue to hold if the underlying logic of the agenda is non-classical.

6.2 Quantifying over Aggregation Rules

As mentioned at the beginning of this paper, one of the goals of formalizing a given setting in computational social choice is to represent known impossibility or characterization results, such as Arrow's Theorem. In judgment aggregation, a result that is analogous to that of Arrow for its importance in the field is the characterization of the majority rule given by May (1952).

While originally formulated for preference aggregation, this result can be rephrased in binary aggregation as stating that, for an odd number of agents and one issue, an aggregator satisfies $\mathrm{N}^{\mathcal{D}}$, M^{I} and A if and only if it is the majority rule. Recalling the conclusions of Section 4, we see that the right-to-left direction of the theorem is easily expressible in our framework, as it amounts to $\models \mathsf{Prof}_{\mathrm{IC}}(\mathbb{B}^{n,1}, \mathbb{O}^1) \rightarrow [\mathsf{maj}(\mathbb{B}^{n,1})](\mathsf{A} \wedge \mathsf{M}^{\mathsf{I}} \wedge \mathsf{N}^{\mathcal{D}})$ for any odd n. On the other hand, for the left-to-right direction we find a negative answer: in order to express it, we would need to leave unspecified the program for the aggregation rule to be written inside the axioms in Theorem 3, which would then turn out to be equivalent to the program for majority, something that it is not possible to achieve in DL-PA.

Nevertheless, we can express existing results for the safety of the agenda problem. As an example, the known theorem stating that a constraint is safe for the majority rule if and only if the constraint has the MP, originally proved by Dietrich and List (2007b), can be stated as $\models [\mathsf{prof}_{\mathrm{IC}}(\mathbb{B}, \mathbb{O})\,;\mathsf{maj}(\mathbb{B})]\mathrm{IC} \leftrightarrow \mathsf{MP}_{\mathrm{IC}}$ for an odd number of individuals.

6.3 Automated Reasoning

An essential feature of DL-PA is that this modal logic is grounded on propositional logic, as we have seen in Section 2.2. Our work thus gives direct access to SAT-solvers to enhance research in judgment aggregation, since we now have a chain of translations from aggregation problems to DL-PA, and from DL-PA to propositional logic. Therefore, we can get the best of both formalisms: DL-PA gives us an elegant and compact way to represent the key features of judgment aggregation, and at the same time we do not have to design specific solvers for it since we can deploy the existing ones for propositional logic.

Similar concerns about implementation have been addressed with respect to the Logic for Social Choice Functions as well (Ciná and Endriss, 2016), though in that case the focus was on preference aggregation rather than judgment aggregation. Interestingly, this logic has indeed

been implemented by Troquard (2011) in the Common Lisp language, however focusing on game theoretical notions for positional scoring rules (e.g., strategy-proofness and the presence of Nash or dominance equilibria in given profiles). While the possibility of translating Judgment Aggregation Logic into first-order logic is envisaged by Ågotnes et al. (2011) in the conclusion of their paper, a possible implementation of the setting is left as an issue for future research.

7 Conclusions

In this paper we translated for the first time classical problems in judgment aggregation into an existing logical formalism reducible to propositional logic, in line with similar work in the knowledge representation literature. The core ideas of our translation from judgment aggregation to DL-PA consist in turning profiles of individual ballots into a specific type of valuation, and aggregation rules into DL-PA programs modifying the truth values of a set of outcome variables.

Firstly, we investigated the boundaries of expressibility of aggregation rules in DL-PA. Theorem 1 gave us a positive result, in that any aggregator can be translated into DL-PA, albeit with a program of length exponential in the number of issues and agents. To circumvent this shortcoming we provided more compact translations of some well-known aggregation rules. We then oriented our research towards single and multi-profile axioms for judgment aggregation rules, translating them as propositional or DL-PA formulas. Additionally, we studied the safety of the agenda problem by using the concept of prime implicants to provide DL-PA equivalents of the most common agenda properties.

Our results showcase the flexibility of DL-PA as a useful formalism for judgment aggregation problems, one in which human-readable formulas can be written and later fed to automated solvers to obtain the result of an aggregation rule, check the satisfaction of axiomatic properties, or verify the safety of a given judgment aggregation agenda.

The framework proposed in this paper could be easily generalized to a setting where agents are allowed to abstain on the issues (Dietrich and List, 2008). Specifically, it would be sufficient to consider an additional set of variables for the profile, to keep track of the issues on which the agents abstained. The definitions of the aggregation rules and the axioms would then be easy to adapt. Furthermore, the possibility of defining counters and to formalize the concept of the Hamming distance allows the DL-PA formalism to incorporate the study of strategic judgment aggregation, opening an interesting direction for future work (Dietrich and List, 2007a).

Acknowledgements

We would like to thank the anonymous reviewer for their valuable comments and remarks, as well as Ulle Endriss for the detailed feedback provided on a previous draft of this paper. The present work was partially supported by IDEX UNITI ANR-11-IDEX-0022-02.

References

T. Ågotnes, W. van der Hoek, and M. Wooldridge. On the logic of preference and judgment aggregation. *Autonomous Agents and Multi-Agent Systems*, 22(1):4–30, 2011.

K. J. Arrow, A. K. Sen, and K. Suzumura, editors. *Handbook of Social Choice and Welfare*. North-Holland, 2002.

P. Balbiani, A. Herzig, and N. Troquard. Dynamic logic of propositional assignments: a well-behaved variant of PDL. In *Proceedings of the 28th Annual ACM/IEEE Symposium on Logic in Computer Science, (LICS)*, 2013.

S. J. Brams and P. C. Fishburn. Voting procedures. In *Handbook of Social Choice and Welfare*, volume 1, pages 173–236. Elsevier, 2002.

F. Brandl, F. Brandt, and C. Geist. Proving the incompatibility of efficiency and strategyproofness via SMT solving. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI)*, 2016.

F. Brandt and C. Geist. Finding strategyproof social choice functions via SAT solving. *Journal of Artificial Intelligence Research (JAIR)*, 55:565–602, 2016.

F. Brandt, C. Geist, and D. Peters. Optimal bounds for the no-show paradox via SAT solving. *Mathematical Social Sciences*, 90:18–27, 2017.

G. Ciná and U. Endriss. A syntactic proof of Arrow's theorem in a modal logic of social choice functions. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems, (AAMAS)*, 2015.

G. Ciná and U. Endriss. Proving classical theorems of social choice theory in modal logic. *Autonomous Agents and Multi-Agent Systems*, 30(5):963–989, 2016.

F. Dietrich and C. List. Strategy-proof judgment aggregation. *Economics & Philosophy*, 23(3): 269–300, 2007a.

F. Dietrich and C. List. Judgment aggregation by quota rules: majority voting generalized. *Journal of Theoretical Politics*, 19(4):391–424, 2007b.

F. Dietrich and C. List. Judgment aggregation without full rationality. *Social Choice and Welfare*, 31(1):15–39, 2008.

E. Dokow and R. Holzman. Aggregation of binary evaluations for truth-functional agendas. *Social Choice and Welfare*, 32(2):221–241, 2009.

S. Doutre, A. Herzig, and L. Perrussel. A dynamic logic framework for abstract argumentation. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 2014.

J. van Eijck. Making things happen. *Studia Logica*, 66(1):41–58, 2000.

U. Endriss. Judgment aggregation. In F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. D. Procaccia, editors, *Handbook of Computational Social Choice*. Cambridge University Press, 2016.

U. Endriss and U. Grandi. Binary Aggregation by selection of the most representative voter. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence, (AAAI)*, 2014.

U. Endriss, U. Grandi, and D. Porello. Complexity of judgment aggregation. *Journal of Artificial Intelligence Research*, 45:481–514, 2012.

U. Endriss, U. Grandi, R. de Haan, and J. Lang. Succinctness of languages for judgment aggregation. In *Proceedings of the 15th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 2016.

M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211, 1979.

B. Gaudou, A. Herzig, E. Lorini, and C. Sibertin-Blanc. How to do social simulation in logic: modelling the segregation game in a dynamic logic of assignments. In *Proceedings of the 12th International Workshop on Multi-Agent Systems and Agent-Based Simulation*, 2011.

C. Geist and U. Endriss. Automated search for impossibility theorems in social choice theory: Ranking sets of objects. *Journal of Artificial Intelligence Research*, 40:143–174, 2011.

U. Grandi and U. Endriss. Binary aggregation with integrity constraints. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, 2011.

U. Grandi and U. Endriss. First-order logic formalisation of impossibility theorems in preference aggregation. *Journal of Philosophical Logic*, 42(4):595–618, 2013.

D. Grossi and G. Pigozzi. Judgment aggregation: a primer. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(2):1–151, 2014.

A. Herzig. Belief change operations: a short history of nearly everything, told in dynamic logic of propositional assignments. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 2014.

A. Herzig, E. Lorini, F. Moisan, and N. Troquard. A dynamic logic of normative systems. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, 2011.

J. G. Kemeny. Mathematics without numbers. *Daedalus*, 88(4):577–591, 1959.

L. A. Kornhauser and L. G. Sager. The one and the many: adjudication in collegial courts. *California Law Review*, 81(1):1–59, 1993.

J. Lang and M. Slavkovik. How hard is it to compute majority-preserving judgment aggregation rules? In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI)*, 2014.

C. List. The theory of judgment aggregation: an introductory review. *Synthese*, 187(1):179–207, 2012.

C. List and P. Pettit. Aggregating sets of judgments: an impossibility result. *Economics and Philosophy*, 18(01):89–110, 2002.

J. Marchi, G. Bittencourt, and L. Perrussel. Prime forms and minimal change in propositional belief bases. *Annals of Mathematics and Artificial Intelligence*, 59(1):1–45, 2010.

P. Marquis. Consequence finding algorithms. In *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, pages 41–145. Springer, 2000.

K. O. May. A set of independent necessary and sufficient conditions for simple majority decision. *Econometrica: Journal of the Econometric Society*, pages 680–684, 1952.

M. K. Miller and D. Osherson. Methods for distance-based judgment aggregation. *Social Choice and Welfare*, 32(4):575–601, 2009.

T. Nipkow. Social choice theory in HOL: Arrow and Gibbard-Satterthwaite. *Journal of Automated Reasoning*, 43(3):289–304, 2009.

M. Pauly. Axiomatizing collective judgment sets in a minimal logical language. *Synthese*, 158 (2):233–250, 2007.

T. Perkov. Natural deduction for modal logic of judgment aggregation. *Journal of Logic Language and Information*, pages 1–20, 2016.

D. Porello. Judgement aggregation in non-classical logics. *Journal of Applied Non-Classical Logics*, 27(1-2):106–139, 2017.

V. R. Pratt. Semantical considerations on Floyd-Hoare logic. In *Proceedings of the 17th Annual Symposium on Foundations of Computer Science (FOCS)*, 1976.

A. Rubinstein. The single profile analogues to multi profile theorems: mathematical logic's approach. *International Economic Review*, pages 719–730, 1984.

P. Tang and F. Lin. Computer-aided proofs of Arrow's and other impossibility theorems. *Artificial Intelligence*, 173(11):1041–1053, 2009.

N. Troquard. Logics of social choice and perspectives on their software implementation. Accompanying notes for the Schloss Dagstuhl Seminar 11101, 2011.

N. Troquard, W. van der Hoek, and M. Wooldridge. Reasoning about social choice functions. *Journal of Philosophical Logic*, 40(4):473–498, 2011.

M. B. Wells. Generation of permutations by transposition. *Mathematics of Computation*, pages 192–195, 1961.

F. Wiedijk. Arrow's impossibility theorem. *Formalized Mathematics*, 15(4):171–174, 2007.

A An Example of a Rule Compactly Expressed in DL-PA

We here give a concrete example of how the programs we defined in Section 3 shorten the general program schema given in the proof of Theorem 1. More precisely, we spell out below the program $\mathsf{msa}_{\mathrm{IC}}$ for 3 agents and 3 issues where $\mathrm{IC} = (p_1 \leftrightarrow (p_2 \wedge p_3))$, as in Example 1.

$$\mathsf{msa}_{\mathrm{IC}}(\mathbb{B}^{3,3}) := \mathsf{maj}(\mathbb{B}^{3,3})\,;\mathsf{store}(\mathbb{O}^3)\,;\mathsf{flip}^{\geq 0}(\mathbb{O}^3)\,;\mathrm{IC}?\,;[\mathsf{restore}^1(\mathbb{O}^3)\,;\mathsf{restore}^{\geq 0}(\mathbb{O}^3)]\neg\mathrm{IC}?$$

$$= \mathop{;}_{j\in\mathcal{I}} \big(\mathsf{zero}(\mathsf{pro}\cup\mathsf{con})\,; \mathop{;}_{i\in\mathcal{N}} (\mathsf{if}\ p_{ij}\ \mathsf{then}\ \mathsf{incr}(\mathsf{pro})\ \mathsf{else}\ \mathsf{incr}(\mathsf{con}))\,;\mathsf{if}\ \mathsf{pro} > \mathsf{con}\ \mathsf{do}\ +p_j\big)\,;$$
$$\mathsf{store}(\mathbb{O}^3)\,;\mathsf{flip}^{\geq 0}(\mathbb{O}^3)\,;\mathrm{IC}?\,;[\mathsf{restore}^1(\mathbb{O}^3)\,;\mathsf{restore}^{\geq 0}(\mathbb{O}^3)]\neg\mathrm{IC}?$$

$$= \big(-q_1\,;-q_2\,;-q_1'\,;-q_2'\,;((p_{11}?\,;(\neg(q_1\wedge q_2)?\,;(\neg q_1?\,;+q_1)\cup(\neg q_2?\,;+q_2\,;-q_1)))\cup(\neg p_{11}?\,;$$
$$(\neg(q_1'\wedge q_2')?\,;(\neg q_1'?\,;+q_1')\cup(\neg q_2'?\,;+q_2'\,;-q_1')))\,;(p_{21}?\,;(\neg(q_1\wedge q_2)?\,;(\neg q_1?\,;+q_1)\cup$$
$$(\neg q_2?\,;+q_2\,;-q_1)))\cup(\neg p_{21}?\,;(\neg(q_1'\wedge q_2')?\,;(\neg q_1'?\,;+q_1')\cup(\neg q_2'?\,;+q_2'\,;-q_1')))\,;(p_{31}?\,;(\neg(q_1\wedge q_2)?\,;$$
$$(\neg q_1?\,;+q_1)\cup(\neg q_2?+q_2\,;-q_1)))\cup(\neg p_{31}?\,;(\neg(q_1'\wedge q_2')?\,;(\neg q_1'?\,;+q_1')\cup(\neg q_2'?\,;+q_2'\,;-q_1')))\,;$$
$$((((q_1\wedge\neg q_1')\vee(q_1\leftrightarrow q_1'\wedge q_2\wedge\neg q_2'))?\,;+p_1)\cup(\neg((q_1\wedge\neg q_1')\vee(q_1\leftrightarrow q_1'\wedge q_2\wedge\neg q_2'))?\,;\top?))\,;$$
$$\big(-q_1\,;-q_2\,;-q_1'\,;-q_2'\,;(p_{12}?\,;(\neg(q_1\wedge q_2)?\,;(\neg q_1?\,;+q_1)\cup(\neg q_2?\,;+q_2\,;-q_1)))\cup(\neg p_{12}?\,;(\neg(q_1'\wedge q_2')?\,;$$
$$(\neg q_1'?\,;+q_1')\cup(\neg q_2'?+q_2'\,;-q_1'))\,;(p_{22}?\,;(\neg(q_1\wedge q_2)?\,;(\neg q_1?\,;+q_1)\cup(\neg q_2?\,;+q_2\,;-q_1)))\cup(\neg p_{22}?\,;$$
$$(\neg(q_1'\wedge q_2')?\,;(\neg q_1'?\,;+q_1')\cup(\neg q_2'?+q_2'\,;-q_1')))\,;(p_{32}?\,;(\neg(q_1\wedge q_2)?\,;(\neg q_1?\,;+q_1)\cup$$
$$(\neg q_2?\,;+q_2\,;-q_1)))\cup(\neg p_{32}?\,;(\neg(q_1'\wedge q_2')?\,;(\neg q_1'?\,;+q_1')\cup(\neg q_2'?+q_2'\,;-q_1')))\,;$$
$$((((q_1\wedge\neg q_1')\vee(q_1\leftrightarrow q_1'\wedge q_2\wedge\neg q_2'))?\,;+p_2)\cup(\neg((q_1\wedge\neg q_1')\vee(q_1\leftrightarrow q_1'\wedge q_2\wedge\neg q_2'))?\,;\top?))\,;$$
$$\big(-q_1\,;-q_2\,;-q_1'\,;-q_2'\,;(p_{13}?\,;(\neg(q_1\wedge q_2)?\,;(\neg q_1?\,;+q_1)\cup(\neg q_2?+q_2\,;-q_1)))\cup(\neg p_{13}?\,;(\neg(q_1'\wedge q_2')?\,;$$
$$(\neg q_1'?\,;+q_1')\cup(\neg q_2'?\,;+q_2'\,;-q_1'))\,;(p_{23}?\,;(\neg(q_1\wedge q_2)?\,;(\neg q_1?\,;+q_1)\cup(\neg q_2?+q_2\,;-q_1)))\cup(\neg p_{23}?\,;$$
$$(\neg(q_1'\wedge q_2')?\,;(\neg q_1'?\,;+q_1')\cup(\neg q_2'?\,;+q_2'\,;-q_1')))\,;(p_{33}?\,;(\neg(q_1\wedge q_2)?\,;(\neg q_1?\,;+q_1)\cup$$
$$(\neg q_2?\,;+q_2\,;-q_1))\cup(\neg p_{33}?\,;(\neg(q_1'\wedge q_2')?\,;(\neg q_1'?\,;+q_1')\cup(\neg q_2'?+q_2'\,;-q_1')))\,;$$
$$((((q_1\wedge\neg q_1')\vee(q_1\leftrightarrow q_1'\wedge q_2\wedge\neg q_2'))?\,;+p_3)\cup(\neg((q_1\wedge\neg q_1')\vee(q_1\leftrightarrow q_1'\wedge q_2\wedge\neg q_2'))?\,;\top?))\,;$$
$$(p_1?\,;+p_1')\cup(\neg p_1?\,;-p_1')\,;(p_2?\,;+p_2')\cup(\neg p_2?\,;-p_2')\,;(p_3?\,;+p_3')\cup(\neg p_3?\,;-p_3')\,;$$
$$((+p_1\cup-p_1)\,;(+p_2\cup-p_2)\cup(+p_3\cup-p_3))\,;(p_1\leftrightarrow(p_2\wedge p_3))?\,;$$
$$[(p_1\oplus p_1'?\,;(p_1'?\,;+p_1)\cup(\neg p_1'?\,;-p_1))\,;(p_2\oplus p_2'?\,;(p_2'?\,;+p_2)\cup(\neg p_2'?\,;-p_2))\,;$$
$$(p_3\oplus p_3'?\,;(p_3'?\,;+p_3)\cup(\neg p_3'?\,;-p_3))\,;(\top?\cup(p_1'?\,;+p_1)\cup(\neg p_1'?\,;-p_1))\,;$$
$$(\top?\cup(p_2'?\,;+p_2)\cup(\neg p_2'?\,;-p_2))\,;(\top?\cup(p_3'?\,;+p_3)\cup(\neg p_3'?\,;-p_3))]\neg(p_1\leftrightarrow(p_2\wedge p_3))?$$

As shown above, the full specification in DL-PA of the $\mathsf{msa}_{\mathrm{IC}}$ program, even for a small profile of 3 agents and 3 issues, turns out to be indeed rather long. Nevertheless, observe that the propositional formula expressing this rule, constructed following the proof of Theorem 1, needs to specify the outcome for all the $2^3 \times 2^3 \times 2^3$ possible profiles, which results in a significantly longer formula.