

Constraint Satisfaction Problems:  
Probabilistic Approach and Applications to  
Social Choice Theory

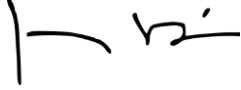
John Livieratos, Ph.D.

December 4, 2020

Department of Mathematics,  
National and Kapodistrian University of Athens

**Committee Members**

**Josep Díaz**, Computer Science Department, Universitat Politècnica de Catalunya, Barcelona.



**Marcel Fernandez**, Department of Network Engineering, Universitat Politecnica de Catalunya, Spain.



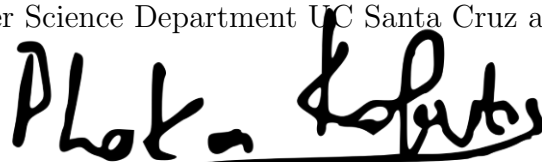
**Dimitris Fotakis**, Division of Computer Science, School of Electrical and Computer Engineering, NTUA.



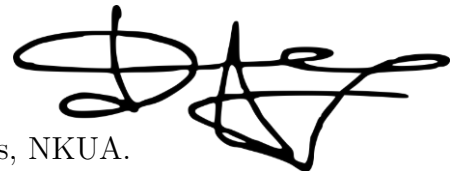
**Lefteris Kirousis** (advisor), Department of Mathematics, NKUA.



**Phokion G. Kolaitis**, Computer Science Department UC Santa Cruz and IBM Research - Almaden.



**Stavros Kolliopoulos**, Department of Informatics and Telecommunications, NKUA.



**Dimitrios M. Thilikos**, Department of Mathematics, NKUA.

Dedicated to  
Fani, Susana, John and Toula,  
who would have certainly  
appreciated all that

and to Sara,  
for continuing to be around.



# Abstract

In this Ph.D thesis, we work in one of the most well studied class of problems in Computer Science, that of *Constraint Satisfaction Problems* (CSPs). In one of their usual formulations, CSPs consist of a set of variables that take values in a common domain set. Groups of variables are tied by constraints that restrict the possible combinations of values that the variables can have in a solution. In such a setting, there are many objectives that one might be interested in: checking if there is a solution, finding or approximating one, or considering how fast an algorithmic procedure can do all that.

The framework of CSPs is broad enough to model a great number of interesting problems in computer science, like the satisfiability of propositional formulas and graph coloring problems. It is also a very developed field on its own accord, with a lot of interesting results that classify the computational complexity of classes of CSPs and delineate the bounds between tractability and NP-hardness. The machinery used to tackle such problems is broad, including polynomial-time algorithms that solve classes of CSPs, randomized ones that find or approximate solutions given some conditions that the CSP in question must satisfy and algebraic manipulations that allow us to relate their computational complexity with structural properties of their sets of constraints.

We begin with an overview of various approaches to CSPs: defining them in the language of Propositional, First or Second Order Logic and via homomorphisms and we consider the subclass of *multi-sorted* CSPs, that is CSPs whose variables are divided in different *sorts* and take values in independent domains. Some of these variations are discussed to show the versatility of CSPs and provide some context to our work, while others are utilized to prove our results.

The first part of our results concerns what is known as the *probabilistic approach*. Here, we devise randomized algorithms that (i) prove conditions

that guarantee the existence of solutions to a given instance of a CSP and (ii) in case a solution exists, find it efficiently. A solution in this setting is usually expressed as a point in a probability space such that no event, from a set of events that are deemed as “undesirable”, occurs. We work with the seminal *Lovász Local Lemma* (LLL) and its variation, Shearer’s Lemma, which, given some bounds concerning the probabilities of undesirable events and the way these events depend on each other, provide conditions that imply all the events can be avoided with positive probability. A solution in this setting, is a point in a probability space such that none of the events occur. All our work is situated in the *variable framework* of Moser and Tardos, where the events are assumed to be defined upon *independent random variables*. Although this is a restriction of the general setting, it is a broad framework that easily translates to the language of CSPs and that is particularly handy for algorithmic purposes.

Specifically, we define two new notions of dependency between the events, the *variable-directed lopsidedependency* (VDL) and the *directed dependency* (*d-dependency*), which are specifically tailored to facilitate the algorithmic manipulation of events that are *negatively correlated*. It is quite common in practice to depict dependencies between the events by a *dependency graph*, where the nodes correspond to the events and unconnected events are considered independent. We thus discuss how the directed dependency graphs that our notions give rise to, relate with other such graphs in the bibliography. Furthermore, we show that the *d-dependency* condition gives rise to a sparser dependency graph than other known such conditions in the variable framework, thus allowing for stronger versions of the LLL to be proven.

We then proceed to prove the simple version of the LLL of the VDL condition. That is, we design an algorithm which, if the probabilities of the events are upper bounded by a common number  $p \in [0, 1)$ , the VDL graph has maximum degree  $d$  and  $epd \leq 1$ , efficiently finds a point in the probability space such that none of the events occur, thus showing at the same time that such a point must exist in the first place. We also prove the more general *asymmetric* version of the LLL for the *d-dependency* graph, where the probability of each event is bounded by a number relating to the probabilities of the events depending on it. We then prove the even stronger Shearer’s lemma for the underlying undirected graph of the *d-dependency* one, which bounds the probabilities of the events by polynomials defined over the independent sets of the graph.

The proofs for these versions of the LLL and Shearer’s lemma employ a

direct probabilistic approach, in which we show that the probability that our algorithms last for at least  $n$  steps is inverse exponential to  $n$ , by expressing it by a recurrence relation which we subsequently solve using advanced analytic tools, such as Bender and Richmond’s *Lagrange Inversion Formula* and *Gelfand’s Formula* for the *spectral radius* of matrices. In contrast, most extant work bounds only the expectation of the steps performed by such algorithms. We believe that this fact is interesting in each own accord. Nevertheless, we have applied our method in two interesting combinatorial problems. First, we show that  $2\Delta - 1$  colors suffice to acyclically color the edges of a graph with maximum degree  $\Delta$ , that is, we want the resulting coloring to contain neither incident edges with the same color, nor bichromatic cycles. We thus match the best possible bound for Moser-like algorithms, as observed by Cai et al. [Acyclic edge colourings of graphs with large girth. *Random Structures & Algorithms*, 50(4):511–533, 2017]. We also show how to explicitly construct binary  $c$ -separating codes whose rate matches the optimal known one.  $c$ -separating codes are  $M \times n$  matrices over some alphabet  $\mathcal{Q}$ , where, in any two sets  $U$  and  $V$  of at most  $c$  rows, there is at least one column such that the set of elements in  $U$  is disjoint with that in  $V$ . Although such codes are very useful for applications, explicit constructions are scarce.

The second part of our results lies in *Social Choice Theory* and, specifically, in *Judgment Aggregation*, where a group of agents collectively decides a set of issues and where, both the individual positions of each agent and the aggregated positions (the *social outcome*) needs to adhere to some restrictions that reflect logical consistency requirements. The aim is to find aggregating procedures that preserve these requirements and do not degenerate to dictatorships, that is aggregators that always output the positions of a specific agent.

Firstly, we consider the case where these restrictions are expressed by a set of  $m$ -ary vectors  $X$  over some finite domain  $\mathcal{D}$ , where  $m$  is the number of issues to be decided. That is,  $m$  contains the allowed combinations of votes over the issues and a vector not in  $X$  is deemed “irrational”. In this setting, we characterize *possibility domains*, that is sets  $X$  where non-dictatorial aggregation is possible, via the types of aggregators they admit. Furthermore, we provide an analogous characterization for a subclass of possibility domains we named uniform possibility domains, which are domains that admit aggregators that are not dictatorial even when restricted to any issue and any binary subset of allowed positions. We also show that uniform possi-

bility domains give rise to tractable multi-sorted CSPs, while any domain that is not uniform, gives rise to NP-complete multi-sorted CSPs, thus tying the possibility of non-dictatorial aggregation with a dichotomy result in the complexity of multi-sorted CSPs.

We then proceed to consider Boolean such domains, that are given as the sets of models of propositional formulas, which, in the bibliography, are called integrity constraints. We provide syntactic characterizations for integrity constraints that give rise to (uniform) possibility domains and also to domains admitting a variety of non-dictatorial aggregators with specific properties that have appeared in the bibliography. We also show how to efficiently identify integrity constraints of these types and how to efficiently construct such constraints given a Boolean domain  $X$  of the corresponding type.

Finally, we turn our attention to the problem of recognizing if a domain admits a (uniform) non-dictatorial aggregator. In case  $X$  is provided explicitly, as a set of  $m$ -ary vectors, we design polynomial-time algorithms that solve this problem. In case  $X$  is Boolean and provided either via an integrity constraint, or, as in the original framework of Judgment Aggregation, as the set of consistent evaluations of a set of propositional formulas, called an agenda, we provide upper and lower complexity bounds in the *polynomial hierarchy*. We extend these results to include the cases where  $X$  admits non-dictatorial aggregators with desirable properties.



# Περίληψη

Σε αυτήν την διδακτορική διατριβή δουλεύουμε σε Προβλήματα Ικανοποίησης Περιορισμών (Π.Ι.Π.), τα οποία αποτελούν μία από τις πιο καλά μελετημένες κατηγορίες προβλημάτων στην Επιστήμη της Πληροφορικής. Στην συνήθη τους μορφή, τα Π.Ι.Π. αποτελούνται από ένα σύνολο μεταβλητών που παίρνουν τιμές από ένα κοινό πεδίο ορισμού. Οι μεταβλητές αυτές υπόκεινται σε διάφορους περιορισμούς, που ορίζουν τους αποδεκτούς συνδυασμούς τιμών μίας λύσης. Σε αυτό το πλαίσιο, υπάρχουν διάφορα ζητήματα που μπορούν να μας απασχολήσουν: ο έλεγχος για το αν υπάρχει λύση, το να βρούμε ή να προσεγγίσουμε μία λύση, ή το πόσο γρήγορα μπορεί να κάνει τα προηγούμενα μία αλγοριθμική διαδικασία.

Πολλά ενδιαφέροντα προβλήματα στην περιοχή της Επιστήμης των Υπολογιστών μπορούν να αναπαρασταθούν ως Π.Ι.Π., όπως αυτό της *ικανοποιησιμότητας προτασιακών τύπων* ή του *χρωματισμού γραφημάτων*. Ως αυτόνομο ερευνητικό πεδίο, τα Π.Ι.Π. έχουν μελετηθεί εκτεταμένα, με αποτέλεσμα να υπάρχει ένα μεγάλο πλήθος ερευνών που τα κατηγοριοποιούν με βάση την υπολογιστική πολυπλοκότητά τους και που διαχωρίζουν τα (πολυωνυμικώς) επιλύσιμα από τα NP-δύσκολα. Πέραν τούτου, έχουν αναπτυχθεί πολλά εργαλεία για αυτά τα προβλήματα, όπως πολυωνυμικού χρόνου αλγόριθμοι για συγκεκριμένες κατηγορίες Π.Ι.Π., πιθανοτικοί αλγόριθμοι που βρίσκουν ή προσεγγίζουν λύσεις σε Π.Ι.Π. που ικανοποιούν ορισμένες συνθήκες και αλγεβρικές προσεγγίσεις μέσω των οποίων συσχετίζουμε την υπολογιστική τους πολυπλοκότητα με την δομή του συνόλου των περιορισμών τους.

Στην παρούσα εργασία, ξεκινάμε με την παρουσίαση διαφόρων προσεγγίσεων στα Π.Ι.Π.: μέσω *Προτασιακής, Πρωτοβάθμιας ή Δευτεροβάθμιας Λογικής* και μέσω *ομομορφισμών*. Μελετούμε επίσης την παραλλαγή των *Πολυ-ειδών Π.Ι.Π.*, των οποίων οι μεταβλητές χωρίζονται σε διαφορετικά είδη και παίρνουν τιμές από διακριτά και ανεξάρτητα πεδία ορισμών. Κάποιες από αυτές τις παραλλαγές και προσεγγίσεις δίνονται ώστε να φανεί η ευρύτητα του πλαισίου μέσα

στο οποίο δουλεύουμε, ενώ άλλες χρησιμοποιούνται στα ίδια τα αποτελέσματά μας.

Το πρώτο μέρος των αποτελεσμάτων μας αφορά την λεγόμενη *πιθανοτική προσέγγιση*. Σχεδιάζουμε αλγορίθμους που (ι) αποδεικνύουν συνθήκες οι οποίες εγγυώνται την ύπαρξη λύσης σε στιγμιότυπα ενός Π.Ι.Π. και (ii) σε περίπτωση που υπάρχει λύση, την βρίσκουν σε πολυωνυμικό χρόνο. Οι λύσεις αυτές συνήθως αναπαριστώνται από σημεία ενός πιθανοτικού χώρου στα οποία δεν ισχύει κανένα από μία σειρά γεγονότα τα οποία κρίνονται ως ‘ανεπιθύμητα’. Δουλεύουμε με το *Τοπικό Λήμμα του Lovász* (Lovász Local Lemma) και την παραλλαγή του, το *Λήμμα του Shearer*. Αυτά τα λήμματα, δοθέντων κάποιων άνω φραγμάτων στην πιθανότητα μη-επιθυμητών γεγονότων να συμβούν, καθώς και στο πλήθος των μεταξύ τους συσχετισμών και εξαρτήσεων, μας δίνουν συνθήκες κάτω από τις οποίες υπάρχει λύση. Ακολουθώντας τους Moser και Tardos, υποθέτουμε ότι όλα τα γεγονότα ορίζονται μέσω *ανεξάρτητων τυχαίων μεταβλητών*. Παρόλο που αυτό είναι ένα πιο περιορισμένο πεδίο σε σχέση με το να δουλεύαμε με γενικούς πιθανοτικούς χώρους, είναι ένα ευρύ πλαίσιο που ταιριάζει με αυτό των Π.Ι.Π. και στο οποίο μπορούμε εύκολα να σχεδιάσουμε αλγορίθμους.

Συγκεκριμένα, εισάγουμε δύο νέες έννοιες εξάρτησης μεταξύ των γεγονότων, την *κατευθυνόμενη ασύμμετρη εξάρτηση μεταβλητής* (variable - directed lopsidedness - VDL) και την *κατευθυνόμενη εξάρτηση*. Και οι δύο αυτές έννοιες είναι σχεδιασμένες ώστε να επιτρέπουν την αλγοριθμική επεξεργασία *αρνητικά συσχετισμένων* γεγονότων. Είναι σύνηθες οι εξαρτήσεις μεταξύ των γεγονότων να αποτυπώνονται στα λεγόμενα *γραφήματα εξάρτησης*, των οποίων οι κορυφές αντιστοιχούν στα γεγονότα και τα γεγονότα που δεν ενώνονται με ακμές θεωρούνται ανεξάρτητα. Ως εκ τούτου, συγκρίνουμε τα κατευθυνόμενα γραφήματα εξάρτησης που προκύπτουν από τις δύο παραπάνω έννοιες που εισαγάγαμε, με άλλα τέτοιου είδους γραφήματα που χρησιμοποιούνται στην βιβλιογραφία. Ειδικότερα, δείχνουμε ότι το γράφημα της κατευθυνόμενης εξάρτησης είναι πιο αραιό από τα περισσότερα τέτοια γραφήματα, κάτι που επιτρέπει να αποδειχθούν πιο ισχυρές μορφές του Τοπικού Λήμματος.

Στην συνέχεια, αποδεικνύουμε την απλή μορφή του λήμματος αυτού για την VDL. Συγκεκριμένα, σχεδιάζουμε έναν αλγόριθμο ο οποίος, δεδομένου ότι η πιθανότητα των γεγονότων φράσσεται από έναν αριθμό  $p \in [0, 1]$ , ότι το VDL γράφημα έχει μέγιστο βαθμό  $d$  και ότι  $epd \leq 1$ , βρίσκει σε πολυωνυμικό χρόνο ένα σημείο στον πιθανοτικό χώρο τέτοιο ώστε κανένα από τα μη επιθυμητά γεγονότα να μην ισχύει. Αυτό φυσικά συνεπάγεται και την ύπαρξη τέτοιου σημείου. Στην συνέχεια, αποδεικνύουμε την *μη-συμμετρική εκδοχή* του λήμ-

ματος για το γράφημα κατευθυνόμενης εξάρτησης. Σε αυτήν την εκδοχή, η πιθανότητα κάθε γεγονότος φράσσεται από έναν αριθμό που σχετίζεται με την πιθανότητα όλων των εξαρτωμένων από αυτό γεγονότων. Τέλος, αποδεικνύουμε και το πιο ισχυρό Λήμμα του Shearer για το μη-κατευθυνόμενο γράφημα που προκύπτει αν αγνοήσουμε τις κατευθύνσεις στο γράφημα κατευθυνόμενης εξάρτησης. Σε αυτό το λήμμα, οι πιθανότητες των γεγονότων φράσσονται από πολυώνυμα επί των ανεξάρτητων συνόλων του γραφήματος.

Οι αποδείξεις αυτών των εκδοχών των λημμάτων αυτών γίνονται μέσω μίας απευθείας πιθανοτικής ανάλυσης του πλήθους των βημάτων που κάνουν οι αλγόριθμοι μας. Για να το επιτύχουμε αυτό, φράσσουμε την πιθανότητα να εκτελέσουν τουλάχιστον  $n$  βήματα από μια σχέση αναδρομής, την οποία στην συνέχεια επιλύουμε με αναλυτικά εργαλεία, όπως την εκδοχή της Φόρμουλας Αντιστροφής του Lagrange που έχουν αποδείξει οι Bender και Richmond, ή την Φόρμουλα του Gelfand για την φασματική νόρμα πινάκων. Αντίθετα, οι μέχρι τώρα αλγοριθμικές προσεγγίσεις υπολογίζουν την αναμενόμενη τιμή των βημάτων των αλγορίθμων. Θεωρούμε πως αυτό το γεγονός από μόνο του κάνει την παραπάνω προσέγγισή ενδιαφέρουσα. Παρόλα αυτά, εφαρμόσαμε τις μεθόδους μας σε δύο ενδιαφέροντα υπολογιστικά προβλήματα. Αρχικά, δείχνουμε ότι  $2\Delta - 1$  χρώματα αρκούν για να βάψουμε τις ακμές ενός γραφήματος με τέτοιων τρόπο ώστε, πρώτον, να μην υπάρχουν προσπίπτουσες ακμές ίδιου χρώματος και, δεύτερον, να μην υπάρχουν διχρωματικοί κύκλοι. Το αποτέλεσμα αυτό είναι βέλτιστο για αλγορίθμους τύπου Moser, όπως παρατήρησαν οι Cai et al. [Acyclic edge colourings of graphs with large girth. *Random Structures & Algorithms*, 50(4):511–533, 2017]. Ακόμη, δείχνουμε πως να κατασκευάσουμε  $c$ -διαχωριστικούς κώδικες των οποίων η πληροφορία σε κάθε ψηφίο είναι η βέλτιστη που χει βρεθεί στην βιβλιογραφία. Οι  $c$ -διαχωριστικοί κώδικες είναι πίνακες διάστασης  $M \times n$ , με στοιχεία από ένα αλφάβητο  $\mathcal{Q}$ , ώστε, για κάθε δύο υποσύνολα  $U, V$  το πολύ  $c$  γραμμών τους, να υπάρχει τουλάχιστον μία στήλη της οποίας τα σύνολα των στοιχείων στο  $U$  και αυτά του  $V$  να είναι ξένα. Παρότι αυτοί οι κώδικες είναι πολύ χρήσιμοι στις εφαρμογές, οι κατασκευές τους είναι πολύ σπάνιες.

Το δεύτερο μέρος της δουλειάς μας είναι στην Θεωρία Κοινωνικών Προτιμήσεων και, πιο συγκεκριμένα, στον Συμψηφισμό Κρίσεων, όπου ένα σύνολο ατόμων θέλει να αποφασίσει συλλογικά ένα σύνολο ζητημάτων, και που οι δυνατοί συνδυασμοί ψήφων, τόσο για το κάθε άτομο ξεχωριστά, όσο και για την συλλογική απόφαση υπακούν σε κάποιους περιορισμούς, οι οποίοι επιβάλουν κάποια έννοια λογικής συνέπειας. Στόχος είναι να βρεθούν κανόνες συμψηφισμού που διατηρούν αυτούς τους περιορισμούς και που δεν εκφυλίζονται σε

δικτατορίες, δηλαδή σε κανόνες που καταλήγουν πάντα στις επιλογές ενός συγκεκριμένου ατόμου.

Αρχικά, εξετάζουμε την περίπτωση που οι περιορισμοί αυτοί μας δίνονται ως ένα σύνολο  $X$   $m$ -αδικών διανυσμάτων με στοιχεία από ένα πεδίο ορισμού  $D$ , όπου  $m$  είναι το πλήθος των ζητημάτων επί των οποίων ψηφίζουν τα άτομα. Το  $X$  λοιπόν περιέχει τους επιτρεπόμενους συνδυασμούς ψήφων επί των θεμάτων και κάθε διάνυσμα εκτός του  $X$  θεωρείται μη λογικά συνεπές. Σε αυτό το πλαίσιο, χαρακτηρίζουμε τα πεδία *δυνατότητας*, τα σύνολα  $X$  δηλαδή όπου μπορούμε να βρούμε μη-δικτατορικούς συμψηφιστές, μέσω των ειδών των συμψηφιστών που δέχονται. Στην συνέχεια, χαρακτηρίζουμε με αντίστοιχο τρόπο τα *ομοιόμορφα πεδία δυνατότητας*, τα οποία δέχονται συμψηφιστές οι οποίοι δεν είναι δικτατορικοί ακόμη κι όταν περιορίζονται σε οποιοδήποτε θέμα και δυαδικό υποσύνολο δυνατών θέσεων ως προς το θέμα αυτό. Δείχνουμε επίσης ότι τα πολυειδή Π.Ι.Π. που ορίζονται πάνω σε ομοιόμορφα πεδία δυνατότητας είναι πολυωνυμικώς επιλύσιμα, ενώ τα Π.Ι.Π. που ορίζονται σε σύνολα που δεν είναι τέτοια, είναι NP-δύσκολα, συνδέοντας έτσι την δυνατότητα μη δικτατορικού συμψηφισμού με μια διχοτομία στην πολυπλοκότητα των πολυειδών Π.Ι.Π.

Στην συνέχεια, ασχολούμαστε με την περίπτωση που το  $X$  ορίζεται πάνω σε δυαδικό πεδίο ορισμού και μας δίνεται ως το σύνολο αληθοτιμών μιας προτασιακής φόρμουλας. Στην βιβλιογραφία, τέτοιες φόρμουλες ονομάζονται *περιορισμοί ακεραιότητας*. Αποδεικνύουμε συντακτικούς χαρακτηρισμούς για φόρμουλες που μας δίνουν (ομοιόμορφα) πεδία δυνατότητας, καθώς και πεδίων που δέχονται ένα πλήθος από μη-δικτατορικούς συμψηφιστές με ενδιαφέρουσες ιδιότητες, που έχουν εμφανιστεί στην βιβλιογραφία. Δείχνουμε επίσης πως να αναγνωρίζουμε αποδοτικά τέτοιους συντακτικούς τύπους αλλά και πως να τους κατασκευάζουμε μέσω ενός πεδίου με τις κατάλληλες ιδιότητες.

Τέλος, ασχολούμαστε με το πρόβλημα της αναγνώρισης ενός πεδίου που δέχεται (ομοιόμορφους) μη δικτατορικούς συμψηφιστές. Στην περίπτωση που το  $X$  μας έχει δοθεί ως σύνολο διανυσμάτων, σχεδιάζουμε πολυωνυμικούς αλγορίθμους που το επιλύουν. Αν το  $X$  δίνεται είτε ως το σύνολο αληθείας μιας προτασιακής φόρμουλας, είτε ως ένα σύνολο λογικά συνεπών κρίσεων ενός συνόλου τέτοιων τύπων, δηλαδή μιας *αντζέντας*, όπως ήταν κι ο αρχικός τρόπος μελέτης της συγκεκριμένης περιοχής, δίνουμε άνω και κάτω φράγματα στην υπολογιστική πολυπλοκότητα αυτών των προβλημάτων, μέσω της πολυωνυμικής ιεραρχίας. Αντίστοιχα αποτελέσματα βρίσκουμε και στις περιπτώσεις που ελέγχουμε για μη δικτατορικούς συμψηφιστές με επιθυμητές ιδιότητες που έχουν μελετηθεί στην βιβλιογραφία.

# Acknowledgements

I would like to thank all the committee members for the time they took to consider this thesis and their useful comments. I especially appreciate the past or still ongoing research collaborations with J. Díaz, M. Fernandez and P. G. Kolaitis, which helped shape this thesis. Most of all, I want to thank Lefteris Kirousis, both for our fruitful research endeavours these past five years and, mainly, for his guidance throughout my post-graduate studies. Without his critical every-day involvement and support, this thesis would not have been realized.

There are many peers to which I owe my gratitude to. Unfortunately I can only name a few here. First, I would like to thank Sofia Kokonezi for a very productive and pleasant cooperation, that led to one of the papers included in this thesis. For our frequent collaborations, both during our studies and afterwards, I would like to state my appreciation to Michalis Samaris. Spyros Maniatis, apart from being one of the main influences I had in order to take my math studies seriously and work in the field of computer science, is also one of the most treasured friendships I formed during my studies. Finally, although our paths have been intertwined in much more profound ways, I would like to thank Phil Mavropoulos for our mathematical talks in some of the weirdest and most unexpected places possible.

Outside the academia, it is impossible to name everyone that supported me all these years. I trust that they know who they are, especially those with which we continue to preserve and grow our friendships. I would be remiss though not to mention Dimitris Birgalias, with whom I share a common household and a strong friendship for many years now; my father, Kostas Livieratos, who has been a point of reference for my whole life and whom I trust to support even my most unorthodox choices; and Maria Petropanagiotaki, whose influence in everything I do these past twelve years is much stronger than she realizes.



# Contents

<b>1 Introduction</b>	<b>19</b>
1.1 Algorithmic approaches . . . . .	20
1.1.1 The algebraic approach . . . . .	21
1.1.2 The Probabilistic Approach . . . . .	24
1.1.3 Graph Coloring and Coding Theory . . . . .	28
1.1.4 Judgment Aggregation . . . . .	30
1.2 Our Results . . . . .	33
1.3 Complementary Material . . . . .	43
1.3.1 Graph Theory . . . . .	44
1.3.2 Propositional Logic . . . . .	45
1.3.3 Algorithms . . . . .	47
1.3.4 Analytic and Algebraic Tools . . . . .	54
<b>2 Constraint Satisfaction Problems</b>	<b>61</b>
2.1 Our Framework . . . . .	61
2.1.1 Multi-Sorted CSP's . . . . .	63
2.2 First-Order Logic . . . . .	65
2.2.1 Transitive Closure Logic . . . . .	71
2.3 Homomorphisms . . . . .	73
2.4 Second-Order Logic . . . . .	74
<b>3 The Probabilistic Method</b>	<b>77</b>
3.1 Probabilistic Framework . . . . .	77
3.2 Dependency graphs . . . . .	82
3.3 The Local Lemma . . . . .	97
3.3.1 Main forms of the local lemma . . . . .	97
3.3.2 The Local Lemma in the variable framework . . . . .	101
3.4 Acyclic Edge Coloring . . . . .	104

3.5	Coding Theory	105
<b>4</b>	<b>Uni. Algebra and Comp. Complexity</b>	<b>109</b>
4.1	Clone Theory	109
4.1.1	Operators on finite domains	110
4.1.2	Clones and co-clones	112
4.1.3	Lattices	117
4.1.4	Polymorphisms and the Galois Connection	122
4.1.5	The Boolean Case: Post's Lattice	131
4.2	Computational Complexity of CSP's	135
4.2.1	Schaefer's Dichotomy Theorem through Post's Lattice	135
4.2.2	Dichotomy Theorems	139
4.2.3	Meta-questions for CSP tractability	142
<b>5</b>	<b>Computational Social Choice Theory</b>	<b>145</b>
5.1	Social Choice Theory	145
5.1.1	Preference Aggregation	145
5.1.2	Judgment Aggregation	147
5.2	Abstract Framework	149
5.3	Integrity Constraints	156
<b>6</b>	<b>Algorithmic Lovász Local Lemma</b>	<b>165</b>
6.1	Symmetric VDL Lovász Local Lemma	165
6.1.1	The Algorithm	166
6.1.2	Forests	167
6.1.3	Analysis of VDL-ALG	169
6.2	Asymmetric d-dependency LLL	172
6.2.1	The Algorithm	173
6.2.2	Recurrence for DD-ALG	176
6.3	Shearer's lemma	178
6.3.1	The MAXSETRES algorithm	179
6.3.2	The validation algorithms	183
6.3.3	The stable set matrix	186
6.4	A new bound on Acyclic Edge Coloring	188
6.4.1	Main Algorithm	188
6.4.2	Analysis of the Algorithm	190
6.4.3	Validation Algorithm	192
6.4.4	Recurrence	194



6.5	Application of the LLL to c-separating codes	197
6.5.1	A lower bound on the rate of c-separating binary codes	197
6.5.2	Explicit constructions	202
<b>7</b>	<b>Aggregating Domains</b>	<b>213</b>
7.1	Characterizations for Abstract Domains	213
7.1.1	Characterization of Possibility Domains	213
7.1.2	Characterization of Total Blockedness	220
7.1.3	Uniform Possibility Domains	227
7.2	Syntactic Characterizations	234
7.2.1	Identifying (local) possibility int. constraints	235
7.2.2	Syntactic Characterization of (local) possibility domains	244
7.2.3	Efficient constructions	254
7.2.4	Other Forms of non-Dictatorial Aggregation	256
7.3	Complexity of Aggregation	272
7.3.1	Tractability of Possibility Domains	272
7.3.2	Expressibility in Transitive Closure Logic	282
7.3.3	Tractability of Uniform Possibility Domains	284
7.3.4	Implicitly given Domains	286
7.3.5	Other types of non-dictatorial aggregation	300



# Chapter 1

## Introduction

*Constraint Satisfaction Problems* (CSPs) are a robust way of expressing various computational problems, where we want to find an object which has properties that can be expressed as constraints over a set of variables. There is a rich theory in place on how to solve them, whether we want exact or approximate solutions, that has been developed both as an independent field of study and also within the various scientific fields that CSPs are applied, like *mathematical logic*, *algorithm design* and *integer programming*. Studying CSPs is reminiscent of learning about algebraic equations in school level mathematics, as a way to express “real life” problems and along with the various techniques and tools in place to solve them.

In this Ph.D thesis, we study CSPs from an algorithmic point of view. In matters of *algorithm design*, we use a probabilistic approach to devise algorithms that solve CSPs efficiently. We use these algorithms to prove two seminal theorems, the *Lovász Local Lemma* and *Shearer’s Lemma*. Furthermore, we apply our approach in two well known problems in the fields of *graph colorability* and *coding theory*, the *acyclic edge coloring* and *2-separating codes* respectively. Apart from the above, we consider some important problems in the theory of *judgment aggregation*, concerning *non-dictatorial aggregation*. There, we take advantage of their relation with CSPs, using various tools from the fields of *universal algebra* and *propositional logic*, in order to obtain interesting complexity theoretic results.

The introduction is structured as follows. We first take, in Sec. [1.1](#), a brief overview of CSPs in the areas that come up in our work. Then, in Sec. [1.2](#), we present our results in these areas. Then, in Sec. [1.3](#), we provide some basic facts in scientific fields that permeate our work: *graph theory* in

Subsec. [1.3.1](#), *propositional logic* in Subsec. [1.3.2](#) and *algorithm design* in Subsec. [1.3.3](#). Finally, in Subsec. [1.3.4](#), we discuss some specialized analytic and algebraic tools we will need to analyse some of our algorithms in Ch. [6](#). The last section of the introduction can be safely skipped, and get back to it if the need arises.

## 1.1 Algorithmic approaches

There are numerous computational problems that can be expressed as CSPs. For example, a system of linear inequalities in the field of real numbers can be expressed by a set of variables, taking values in  $\mathbb{R}$ , constrained by relations, that is, subsets of  $R \subseteq \mathbb{R}^n$ , where  $n$  is the number of variables to which the constraint  $R$  is applied. On the other hand, a coloring of a graph can be expressed by relations that constrain the available colors of neighboring vertices or incident edges. A matrix over the set  $\mathbb{Z}$  of integers can have constraints over the elements of its rows and columns, that force it to have specific algebraic properties, like being invertible. In general, each problem can have various ways to be expressed as a CSP, with each providing a unique perspective on how to deal with it.

Although the study of CSPs includes the case of infinite domains, there is an extensive part of research (this thesis included), that focuses on finite, or even Boolean domains. Quite possibly, the most extensively studied CSP in computer science, is the *satisfiability problem* (SAT). It is the first NP-complete problem, as stated by the seminal “Cook-Levin” Theorem [54](#),[214](#). This means that it forms the base of a long list of computational problems that are believed to be intractable, and where finding an efficient algorithm for one of them, would result in efficiently solving all of them. This list started of with 21 such problems, provided by Karp [137](#) and now contains a huge number of problems and various classifications depending on the computational aspect we are interested in.

Apart from complexity theoretic results, expressing a problem as an instance of a CSP, gives access to a number of algorithms that attempt to find exact or approximating solutions, utilize various approaches such as randomness or parallel computation and so on. One can refer to any introductory book in Algorithm Design or Computational Complexity to find such examples, e.g. [10](#),[148](#),[177](#),[193](#),[199](#). For example, we have algorithms that efficiently solve the satisfiability problem when its input is restricted to Horn

formulas [78], renamable Horn formulas [160] and bijunctive [153] formulas.

A CSP is often defined on a set of relations  $\mathcal{R}$ , from where the constraints of each instance are formed. An approach that plays an important role in our work, is to tie the complexity of  $\text{CSP}(\mathcal{R})$  with the algebraic properties of  $\mathcal{R}$ . The volume of research here is again too big to describe it completely. In the sequel, we mention several works that connect directly or indirectly to ours. The main approach we use, is to analyze the set of relations algebraically, through the set of operations that *preserve* the relations within  $\mathcal{R}$ . Such operations are called *polymorphisms*. This immediately puts us in the field of Universal Algebra and, specifically, *Lattice* and *Clone* Theory (see Davey et al. [64] and Szendrei [208]). Thus, it allows us to use powerful classification results, like Post’s complete description of the lattice of Boolean clones [186] and tools like the “Galois Connection” [29, 102, 136, 184, 185].

### 1.1.1 The algebraic approach

The algebraic approach consists of relating structural properties of the set of constraints of a CSP, with complexity theoretic results. This combination of the fields of universal algebra and computational complexity has been proven quite productive for both scientific areas. The main goal is to delineate classes of constraints that give rise to tractable CSPs and, when possible, to obtain *dichotomy* results, that is to provide criteria that classify a CSP as either tractable or NP-complete. Such results are especially interesting, as they show that many classes of CSPs cannot be NP-*intermediate* problems, that is, problems that are neither polynomial-time solvable nor NP-hard. The existence of such intermediate problems, given that  $\text{P} \neq \text{NP}$ , is known by Ladner [155], although all the currently known problems in that class are highly artificial.

In terms of dichotomy theorems, the first such result was provided by Schaefer [192] for CSPs defined over Boolean domains. After this seminal result, there have been many attempts in obtaining equivalent results for CSPs defined over finite domains of arbitrary cardinality. A long line of research focuses on exploiting *consistency* and *closure* properties of the constraints. The former idea is to find conditions which guarantee that locally consistent solutions can be extended to globally consistent ones, while the latter extracts complexity results based on the set of polymorphisms of the constraints.

A CSP is said to be *k-consistent* if any partial solution with  $k-1$  elements,

can be extended to a partial solution with  $k$  elements. Every CSP has a  $k$ -consistent equivalent one, which can be found by known algorithms, like Cooper's [55]. A CSP is of *width*  $k$  when any instance of it has a solution if and only if the corresponding  $k$ -consistent problem does not have the empty constraint. A CSP is of *bounded width*, if it has a finite width  $k$ . Some of the most referenced works in this are Dechter et al. [67, 68] investigations of structural properties that the set of constraints of a CSP needs to have, in order for local consistent solutions to be extended to global solutions. In the same spirit, Jeavons et al. [131–135] explore algebraic properties of the set of constraints, via their polymorphisms, that guarantee either some form of global consistency or that directly imply tractability of the CSP, whereas Creignou et al. [61] design an efficient algorithm for identifying if a Boolean relation can be expressed as a *conjunctive query* over a given set of relations  $\mathcal{R}$ .

Two variables  $x$  and  $y$  in a CSP are arc-consistent if for every allowed value  $a$  for  $x$ , there is an allowed value  $b$  for  $y$  such that  $(a, b)$  can be extended to a solution of the CSP instance. There is a variety of algorithms that attempt to solve CSPs by first enforcing arc-consistency. In that vein, Chen and Dalmau [50] characterize classes of CSPs that can be solved via an arc-consistency algorithm they designed whereas Dalmau et al. [63] algebraically characterize sets of relations  $\mathcal{R}$  that give rise to CSPs that can be solved via arc consistency algorithms.

Renewed interest in this line of work was shown after Feder and Vardi stated their dichotomy conjecture [90], where they argue in favour of a dichotomy result in the complexity of CSPs defined over finite domains of arbitrary cardinality. Bulatov et al. have obtained tractable classes of constraints using *finite algebras* [34, 39, 41] and have provided efficient algorithms for various classes of constraints [14, 33]. By exploiting the algebras of the constraints, Barto et al. [16] have proved the conjecture of Larose and Zadori [159] that characterizes the CSP's of bounded width. Bulatov has also proved dichotomy theorems for CSPs defined over 3-element domains [36] and *conservative* CSPs [35, 37, 38], that is CSPs where the domains of the variables can be restricted arbitrarily. For overviews of results in this area, see Bulatov and Valeriote's talks [42] and Creignou, Kolaitis and Vollmer's exposition [60]. There is also a new approach to the Feder-Vardi dichotomy conjecture by Kun and Szegedy [154], using the PCP theory.

If the domain is allowed to be infinite, one can obtain complexity results using  $\omega$ -categorical structures, like Bodirsky [26] for general infinite domains,

Bodirsky and Kára [25] for *temporal* CSPs. Apart from the above, Bodirsky and Pinsker [28] generalize Schaefer’s dichotomy theorem for domains defined by formulas in the language of graphs, Idziak et al. [127] explore the tractability and learnability of CSP’s via the subalgebra’s of the constraints, whereas Maroti and McKenzie [169] investigate the class of constraints that admit *weakly symmetric* operations.

Of particular interest is the work of Bodirsky and Mamino [27], where they aim at delineating the bound between tractable and intractable CSPs over the integers, rational, real and complex numbers. Finally, Barto et al. [17] characterize finitely generated *varieties*, that is sets of solutions for systems of polynomial equations, via *Taylor terms*. Closely related is the work of Siggers [198] that finds conditions that define a special class of varieties, which has been conjectured to define tractable CSP’s.

Variants of the usual CSPs have also been studied. Bulatov [40] has shown various tractable classes for *multi-sorted* CSPs, where the constraints are divided into different sorts that have some limited independency, as well as for the the problem of *learnability* [43] of quantified formulas. Furthermore Krousis and Kolaitis have provided dichotomy results both for *minimal* satisfiability problems [139] and *propositional circumscription* [140] (for higher levels of the polynomial hierarchy). In the former problem, we are interested in whether a given solution to a CSP is *minimal* with respect to the component-wise partial order on truth assignments, while in the latter we want to find all minimal models of a given propositional formula.

Atserias, Kolaitis and Severini [12] have also studied CSP’s via *operator assignments* over Hilbert spaces (see [117]). In this framework, they have completely characterized the sets  $\mathcal{R}$  for which  $\text{SAT}(\mathcal{R})$ ’s complexity has a gap between the usual framework and their own.

Finally, important results have been obtained concerning the “meta-questions” of constraint tractability, that is, the ability to discern tractable from intractable constraints. Chen and Larose [51] study the problem of deciding if a given structure admits a polymorphism from a class of operations, while Bessi ere et al. [22] and Carbonnel [48] show how to detect if a constraint is closed under the majority or a conservative Mal’tsev operator. Carbonnel [47] has also showed that one can efficiently discern the tractable classes of conservative CSPs from the intractable ones.

### 1.1.2 The Probabilistic Approach

The probabilistic approach consists of proving that an object with some desired properties exists in a given probability space with positive probability. This approach has been extensively used in various combinatorial problems, in the form of bounds to the parameters of a problem that, when satisfied, guarantee the existence of a solution. In CSPs the desired object is an assignment of values to the variables such that all the constraints are satisfied. There are many tools that can be used to this aim, from plain *linearity of expectation*, to *alterations* and the *Second Moment* method. A great variety of such tools are described by Alon and Spencer [8, 200].

Apart from existence, we are also interested in finding such assignments. This amounts to devising probabilistic algorithms that search the probability space for the required object and using the aforementioned tools to prove that they will terminate fast, by finding the object in question. In his Durango lectures [200], Spencer described this process as “finding a needle in a haystack”. Nevertheless, there are tools we can use for this purpose, with one of the most prominent ones, and the focus of this thesis, being the “Lovász Local Lemma” in its various forms.

The Lovász Local Lemma (LLL) was originally stated and proved in 1975 by Erdős and Lovász [85]. Its original *symmetric* form states that given events  $E_1, \dots, E_m$  in a common probability space, if every event depends on at most  $d$  others, and if the probabilities of all are bounded by  $1/(4d)$ , then

$$\Pr \left[ \bigwedge_{j=1}^m \bar{E}_j \right] > 0,$$

and therefore there exists at least one point in the space where none of the events occurs ( $\bar{E}$  denotes the complement of  $E$ ).

The *asymmetric* version entails an undirected *dependency graph*, i.e. a graph with vertices  $j = 1, \dots, m$  corresponding to the events  $E_1, \dots, E_m$  so that for all  $j$ ,  $E_j$  is mutually independent from the set of events corresponding to vertices not connected with  $j$ . The condition that in this case guarantees that  $\Pr[\bigwedge_{j=1}^m \bar{E}_j] > 0$  (and therefore that there exists at least one point where none of the events occurs) is:

$$\text{for every } E_j \text{ there is a } \chi_j \in (0, 1) \text{ such that } \Pr[E_j] \leq \chi_j \prod_{i \in N_j} (1 - \chi_i), \quad (1.1)$$



where  $N_j$  is the neighborhood of vertex  $j$  in the dependency graph.

Improvements can be obtained by considering, possibly directed, sparser graphs than the dependency graph, that correspond to stronger notions of dependency. For a classic example, the *lopsided* version (LLLL) by Erdős and Spencer [86] entails a *directed* graph  $G$  with vertices corresponding to the events such that for all  $E_j$  and for all  $I \subseteq \{1, \dots, m\} \setminus (N_j \cup \{j\})$  we have that

$$\Pr \left[ E_j \mid \bigcap_{i \in I} \bar{E}_i \right] \leq \Pr [E_j], \quad (1.2)$$

where  $N_j$  is the set of vertices connected with  $j$  with an edge originating from  $j$ . Such graphs are known as *lopsidedependency graphs*. The sufficient condition in this case that guarantees that the undesirable events can be avoided is the same:

$$\text{for every } E_j \text{ there is a } \chi_j \in (0, 1) \text{ such that } \Pr[E_j] \leq \chi_j \prod_{i \in N_j} (1 - \chi_i). \quad (1.3)$$

Note though that in this case, the product is over a possibly smaller neighborhood  $N_j$ .

With respect to ordinary (not lopsided) dependency graphs, a sufficient *but also necessary* condition to avoid all events was given by Shearer [196]. It reads:

$$\text{For all } I \in I(G), \quad q_I(G, \mathbf{p}) := \sum_{J \in I(G): I \subseteq J} (-1)^{|J \setminus I|} \prod_{j \in J} p_j > 0, \quad (1.4)$$

where  $I(G)$  is the set of *independent sets* of  $G$  and  $\mathbf{p} = (p_1, \dots, p_m)$  is the vector of probabilities of the events.

By considering other graphs, and the corresponding to them Condition (1.4), variants of the Shearer lemma are obtained. These variants are in general only sufficient, however they apply to sparser dependency graphs. For example, by proving the sufficiency of Condition (1.4) when applied to the lopsidedependency graph of Erdős and Spencer [86], we get Shearer's lemma for lopsidedependency graphs (actually, for Shearer's lemma in this case it is the underlying undirected graph of the lopsidedependency graph that is considered).

Using the LLL to “efficiently” find a point in the probability space such that no undesirable event occurs has been a long struggle. After several

partially successful attempts that expand over more than three decades (Alon [7], Beck [18], Srinivasan [202] and others), Moser [171] in 2009, initially only for the symmetric LLL, gave an extremely simple randomized algorithm that if and when it stops, it certainly produces a point where all events are avoided. Soon after Moser and Tardos [172] expanded this approach to more general versions including the lopsided one. The algorithms were given in the variable framework, where the space is assumed to be the product space of independent random variables  $X_1, \dots, X_l$ , and each event is assumed to depend on a subset of them, called its *scope*. Their algorithm just samples iteratively the variables of occurring events, until all the events cease to occur. For the analysis, they estimate the *expectation* of the number of times each event will be resampled in a given execution of the algorithm by counting “tree-like” structures they call *witness trees* and by estimating the probability that such a tree occurs in the log of the algorithm’s execution. An alternative proof of Moser’s original result [171] has been provided by Spencer [201]. This approach became known as the “*entropy compression*” method, which is compactly presented in Tao [209]. For the proof of the lopsided LLL, Moser and Tardos [172] defined an *undirected* lopsidedependency graph suitable for the variable framework.

A strengthening of the LLL has been achieved via the notion of *cluster expansion* by Bissacot [24] and Pegden [180], that is closely related with statistical mechanics. This was later generalized to Shearer’s Lemma by Harvey and Vondrak [122].

A direct probabilistic approach has been used by Giotis et al. [104, 107] for both the symmetric and asymmetric versions of the simple LLL. There, instead of computing the expected number of steps of Moser’s algorithm, the authors express the probability that it lasts for at least  $n$  steps by a recurrence relation. They then solve this recurrence relations and prove that this probability is inverse exponential to the number of steps that Moser’s algorithm performs. This implies both that the algorithm will terminate with positive probability and that it will do so fast.

In the variable framework, Harris [118] gave a weaker version for the [1.3] condition, entailing the notion of *orderability*, which takes advantage of the way events are related based on the different values of the variables they depend on. He works with the Moser-Tardos notion of lopsidedependency graph and he proves that his weaker sufficient condition can yield stronger results than the classical Shearer’s lemma. Also, recently He et al. [123] gave a necessary and sufficient condition for LLL in the variable framework, but for

the dependency graph where two events are connected if their scopes share at least one variable.

There are numerous applications of the algorithmic versions of both LLL and its lopsided version, even for problems that do not originate from purely combinatorial issues. For example, the lopsided version has been used in, among others, the satisfiability problem, where the focus is on clauses that are in conflict with each other (i.e. have opposing literals of the same variable). Berman et al. [21] and Gebauer et al. [100], [101] have successfully used this notion along with the lopsided LLL to bound the number of occurrences per variable that can be allowed in a formula, while guaranteeing the existence of a satisfying assignment. Also, let us mention the work of Harris and Srinivasan [119] who apply it in the setting of *permutations*, where the undesirable events are defined over permutations  $\pi_k$  of  $\{1, \dots, n_k\}$ ,  $k = 1, \dots, N$  and that of Harvey and Liaw [120] on *Rainbow Hamiltonian cycles*. Finally, we find Albert and Frieze’s work [5] on Hamilton cycles whose edges can be colored with pair-wise distinct colors.

For the non-lopsided versions we have the problem of *covering arrays*, a problem closely related with software and hardware interaction testing. The objective is to find the *minimum* number  $N$ , expressed as a function  $k, t, v$ , such that there exists an  $N \times k$  array  $A$ , with elements taken from a set  $\Sigma$  of cardinality  $v \geq 2$ , so that every  $N \times t$  sub-array of  $A$  contains as one of its rows every element  $x \in \Sigma^t$ . Sarkar and Colbourn [191] improve on known upper bounds for  $N$ , by using LLL. Notably, they also provide an algorithm that constructs an  $N \times k$  array with the above properties, by using a variant of the Moser-Tardos algorithm [172]. Finally, we have the works of Szabó [205] and Gasarch and Haeupler [98] on the van der Waerden number, that is, the smallest number  $W(n)$  such that an arithmetic progression always exists in some part of  $\{1, \dots, W(n)\}$ .

The lopsided LLL was generalized to the framework of arbitrary probability spaces by Harvey and Vondrák [121], by means of a machinery that they called “resampling oracles”. They introduced, in the framework of arbitrary probability spaces, *directed* lopsided dependency graphs they called *lopsided association* graphs. They proved that a graph is a lopsided association graph if and only if it is a graph along the edges of which resampling oracles can be applied. In the generalized framework and based on their lopsided association condition, they algorithmically proved LLLL. In the same framework, they also proved Shearer’s lemma which has also been independently proven by Kolipaka and Szegedy [150]. Achlioptas and Iliopoulos [2,3] have introduced

a powerful abstraction for algorithmic LLL, which is inherently directed and they prove the lopsided LLL in this framework. In their non-probabilistic framework, they try to avoid *flaws*, which are defined as subsets of a domain set  $\mathcal{D}$  and whose connections are expressed by a directed graph on  $\mathcal{D}$ , by performing *random walks* along the edges of the graph.

Let us mention here that Szegedy [206] gives a comprehensive survey of the LLL, that contains many of the algorithmic results.

### 1.1.3 Graph Coloring and Coding Theory

In this subsection we describe two fields that we have applied our probabilistic approach to, namely *graph colorings* and *coding theory*.

**Acyclic Edge Coloring** Let  $G = (V, E)$  be a (simple) graph with  $l$  vertices and  $m$  edges. The chromatic index of  $G$ , often denoted by  $\chi'(G)$ , is the least number of colors needed to *properly* color its edges, i.e., to color them so that no adjacent edges get the same color. If  $\Delta$  is the maximum degree of  $G$ , it is known that its chromatic index is either  $\Delta$  or  $\Delta + 1$  by Vizing [215].

The *acyclic* chromatic index of  $G$ , often denoted by  $\chi'_a(G)$ , is the least number of colors needed to properly color the edges of  $G$  so that no cycle of even length is *bichromatic*, i.e. no even length cycle's edges are colored by only two colors. Notice that in any properly colored graph, no cycle of odd length can be bichromatic. It has been conjectured, by J. Fiamčík [93] and Alon et al. [9], that the acyclic chromatic index of any graph with maximum degree  $\Delta$  is at most  $\Delta + 2$ .

There is an extensive literature on the acyclic chromatic index. The results include:

- For planar graphs  $\chi'_a(G) \leq \Delta + 6$  by Wang and Zhang [216]. Also, for some constant  $M$ , all planar graphs with  $\Delta \geq M$  have  $\chi'_a(G) = \Delta$ , as shown by Cranston [56].
- Cai et al. [46] showed that, for every  $\epsilon > 0$ , there is a  $g$  such that if the girth of  $G$  is at least  $g$ , then  $\chi'_a(G) = (1 + \epsilon)\Delta + O(1)$ .
- For a random  $d$ -regular graph, it holds, by Nešetřil and Wormald [176],  $\chi'_a(G) = d + 1$ , asymptotically almost surely.

- For  $d$ -degenerate graphs, i.e. graphs whose vertices can be ordered so that every vertex has at most  $d$  neighbors greater than itself,  $\chi'_\alpha(G) \leq \lceil (2 + \epsilon)\Delta \rceil$ , for  $\epsilon = 16\sqrt{(d/\Delta)}$ , as proved by Achlioptas and Iliopoulos [1].

For general graphs, the known upper bounds for the chromatic index are  $O(\Delta)$ . Specifically, Esperet and Parreau [87] proved that  $\chi'_\alpha(G) \leq 4(\Delta - 1)$ . This bound was improved to  $\lceil 3.74(\Delta - 1) \rceil + 1$  by Giotis et al. [105]. Also, an improvement of the  $4(\Delta - 1)$  bound was announced by Gutowski et al. [115] (the specific coefficient for  $\Delta$  is not given in the abstract of the announcement). The best bound until now was recently given by Fialho et al. [92], where it is proved that  $\chi'_\alpha(G) \leq \lceil 3.569(\Delta - 1) \rceil + 1$ . These results are proved by designing randomized algorithms that with high probability halt and produce a proper acyclic edge coloring. Such algorithms, assign at each step a randomly chosen color in a way that properness is not destroyed. This approach, unfortunately, necessitates a palette of more than  $2(\Delta - 1)$  colors to deterministically guarantee properness at each step, and then another  $O(\Delta)$  colors to probabilistically guarantee acyclicity.

**Coding Theory** Coding theory is closely related, but different than, *information theory*. The latter was initiated by Shannon [194] and it forms a branch of probability theory, with extensive application to communication systems, focused on data compression and reliable transmission of information. On the other hand, Coding Theory's main focus is to find explicit methods for efficient and reliable data storage and transmission.

We can represent a code as a matrix of symbols over a finite alphabet. The *length* of a code is the number of its columns, while its *size* that of its rows. The *rate* of a code is the fraction of (the logarithm of) its size over its length, and it measures the information stored in each bit of the code.

We are interested in *c-separating* codes, that is codes where, for any two sets of at most  $c$  disjoint rows each, there is a column where the symbols in the first set are different from the symbols in the second. Separating codes [190] have a long tradition of study in the areas of coding theory and combinatorics. This type of codes have been proven useful in applications in the areas of technical diagnosis, construction of hash functions, automata synthesis and traitor tracing.

For a code to have this property is a really strong requirement. Consequently, although there is a vast research focused on obtaining lower and

upper bounds to the rates of such codes, these bounds are weak. Also, explicit constructions of such codes are very scarce.

For instance, for the already non trivial case of binary 2-separating codes, the best lower bound for the rate obtained so far is 0.064 [13, 190, 203], whereas the best upper bound is 0.2835 given by Korner and Simonyi in [151]. One immediately sees that these bounds are not by any means tight. The lower bound is obtained by the elegant technique of random coding with expurgation. The drawback of the random coding strategy is that it gives no clue whatsoever about how to obtain an explicit code matching the bound.

The LLL has been used in some cases in order to obtain bounds for the rate of codes that satisfy various properties. We have already mentioned the work of Sarkar and Colbourn [191] on covering arrays and another interesting example is that of Deng et al. [70] on perfect and separating hash families.

#### 1.1.4 Judgment Aggregation

Kenneth Arrow initiated the theory of preference aggregation by establishing his celebrated General Possibility Theorem (also known as Arrow's Impossibility Theorem) [11], which asserts that it is impossible, even under mild conditions, to aggregate in a non-dictatorial way the preferences of a society. Judgment aggregation is a related framework that brings together aggregation and logic, since the voters have to choose among logically interconnected propositions and the task is to aggregate the votes in a logically consistent manner. Judgment aggregation, was initially motivated by the *doctrinal paradox*, which is a special case of the *discursive dilemma*. For introductory expositions of the history of judgment aggregation see, e.g., Grossi and Pigozzi [114] or List et al. [162, 165]. Wilson [217] introduced a framework for aggregation on general attributes, rather than just preferences or propositions, and proved an Arrow-like result in this context. Wilson's framework was further investigated by Dokow and Holzman [76, 77]. Similar approaches are described by Grandi & Endriss [112] (see also Endriss [81]), which we will study extensively in the sequel and Nehring & Puppe [175].

In the abstract framework we have a set of  $m$  issues on each of which a population of individuals (voters) of size  $n$  may cast votes, from a set  $\mathcal{D}$  of allowed positions. Furthermore the vector of positions on all issues of every single individual should belong to a fixed set  $X \subseteq \mathcal{D}^m$ . This set  $X$  is called the set of *rational* choices or the set of *feasible* choices or simply the domain. We write  $A := \{\mathbf{a}^1, \dots, \mathbf{a}^k\}$  to denote an element of  $(\mathcal{D}^m)^n$ . It is convenient

to think of such elements  $A$  as an  $n \times m$  matrix, where  $a_j^i$  is the position of voter  $i$  on issue  $j$ .

An  $n$ -ary aggregator is a function  $F : (\mathcal{D}^m)^n \mapsto \mathcal{D}^m$  such that  $X$  is closed under  $F$ , meaning that if  $A = \{\mathbf{x}^1, \dots, \mathbf{x}^n\} \in X^n$ , then  $F(A) \in X$ . If  $n = 2$ , then we talk about a binary aggregator, while if  $n = 3$ , we talk about a ternary aggregator. Requiring  $X$  to be closed under  $F$  reflects the notion of rationality of  $F$ , while requiring  $F$  to be defined on  $(\mathcal{D}^m)^n$  reflects the notion of universality (for a recent presentation of these notions, see, e.g., List [166]).

A main theme of judgment aggregation theory is to relate properties of the domain  $X$  with properties of the aggregator  $F$ . An early class of results in this vein are the so called impossibility theorems, which assert that it is impossible to obtain an aggregator  $F$  with certain desired properties, given that the domain  $X$  satisfies certain minimal conditions. Such a result is the impossibility theorem of List and Pettit [163, 164], which asserts that, in the context of judgment aggregation, if the domain contains two propositional variables  $p$  and  $q$ , and the propositional formulas  $(p \wedge q)$  and  $\neg(p \wedge q)$ , then  $X$  has no aggregator that is universal, anonymous, and systematic. Informally, an aggregator is anonymous if it is invariant under permutations of the columns of the input matrices, while an aggregator is systematic if there is a common aggregation rule for all issues (the formal definitions of these two notions will be given in the next chapters).

Stronger than the impossibility results are the characterization results, where one seeks to find necessary and sufficient conditions for a domain to admit aggregators possessing some desired property. In the abstract framework of Dokow and Holzman [77], it is assumed that aggregators satisfy a much weaker property than systematicity, namely, the property known as *independence of irrelevant alternatives* or *issue-by-issue aggregation* (IIA) or, simply, *independence*. This property is equivalent to the existence of functions  $f_1, \dots, f_m$  such that  $f_j : \mathcal{D}^n \mapsto \mathcal{D}$  and  $F(A = \{\mathbf{a}^1, \dots, \mathbf{a}^n\}) := (f_1(\mathbf{a}_1), \dots, f_m(\mathbf{a}_m))$ , where  $\mathbf{a}_j$  is the  $j$ -th column of  $A$ . Note that systematicity is the special case of IIA in which  $f_1 = f_2 = \dots = f_m$ .

In the Boolean framework, also known as the binary framework, it is assumed that, for each issue, there are only two alternatives to choose from. In this framework, Dokow and Holzman [76] discovered a necessary and sufficient condition for a domain  $X$  to have a non-dictatorial aggregator. This characterization involves the property of *total blockedness*, which was originally introduced by Nehring and Puppe [173, 175]. As Dokow and Holz-

man [76] write “Roughly speaking, it [total blockedness] requires that the limitations on feasibility embodied in the set  $X$  make it possible to deduce any position on any issue from any position on any issue, via a chain of deductions.” The precise definition of total blockedness will be given in the sequel.

Dokow and Holzman [77] subsequently investigated aggregation in the non-Boolean or non-binary framework, where, for some issues, there may be more than two alternatives to choose from. By generalizing the notion of a domain being totally blocked to the non-Boolean framework, they gave a sufficient (but not necessary) condition for non-dictatorial aggregation, namely, they showed that if a domain is not totally blocked, then it is a possibility domain. The non-Boolean case has also been studied by Pauly & Van Hees [179] and Herzberg [125]. In [179], the many-valued votes represent degrees of acceptance; necessary and sufficient conditions for an aggregator to be dictatorial are given, however no characterization of the domains that admit non-dictatorial aggregators is proved. In [125], aggregators having the property of systematicity are characterized as a kind of homomorphism. In both [179] and [125], as well as in [77], it is assumed that the set of possible votes is common to all issues.

Recently, Szegedy and Xu [207] discovered necessary and sufficient conditions for non-dictatorial aggregation. Quite remarkably, their approach relates aggregation theory with universal algebra, specifically with the structure of the space of *polymorphisms*. Observe also that aggregators having the property of systematicity are just polymorphisms of the domain  $X$ , where  $X$  is viewed as an  $m$ -ary relation (compare with [125]). However, the necessary and sufficient conditions of Szegedy and Xu [207] still involve the notion of the domain being totally blocked.

All the aforementioned results are situated in the *abstract* framework, where the domain  $X$  is given explicitly as a set of  $m$ -ary tuples. The historically first approach to judgement aggregation was the *logic-based* approach [163, 165]. In that approach, there is a tuple  $\bar{\phi} = (\phi_1, \dots, \phi_m)$  of propositional formulas, called the *agenda*, and the set  $X$  of feasible evaluations consists of consistent judgements concerning the validity of the formulas. Endriss et al. have studied the computational complexity of three interesting problems based on a given agenda [84]: (i) *winner determination*, where given an aggregator and a propositional formula, we want to decide if the formula is part of the collective decision under this aggregator, (ii) *strategic manipulation*, where the goal is to decide if a given aggregation



procedure is subject to manipulation by insincere voting and (iii) *safety of the agenda*, where we want to decide if a class of aggregators preserves the logical consistency restrictions of the agenda. This last problem is related to our framework, with the difference that we search for at least one aggregator of a given class preserving the logical restrictions of our domain. Also, Terzopoulou et al. have studied the case where the individuals do not need to decide on every issue of the agenda, but can instead provide partial judgments [213].

A variant of this approach is the one used by Grandi and Endriss, where the set of restrictions is provided by a propositional formula  $\phi$ , called an *integrity constraint*, as the set of its satisfying assignments [112]. In [83], Endriss et al. study the relation of this framework with the logic-based one, in terms of *succinctness* and of the computational complexity of the winner determination problem.

## 1.2 Our Results

The results contained in this thesis concern the three fields of study presented in Sec. [1.1]. We begin by presenting them concisely. A more analytic exposition follows immediately after.

- I. In the algorithmic LLL framework, we provide probabilistic algorithms that sample and resample random variables until they reach an assignment of values such that none of the undesirable events occur.
  - (i) For a directed version of lopsidedependency in the variable framework we call *variable directed lopsidedependency*, we provide an algorithm that proves the symmetric version of the LLL under this condition. This work has appeared in [145].
  - (ii) We algorithmically prove the asymmetric version of the LLL and Shearer’s lemma. These results have appeared in [106].
  - (iii) We define a another directed version of lopsidedependency for the variable framework, which we call *d-dependency*. We show that this notion produces sparser dependency graphs than other extant notions of lopsidedependency, which results in stronger version of the LLL. For this version of lopsidedependency, we prove the asymmetric LLL and Shearer’s lemma. These results have been published in [147].

- (iv) We design a probabilistic algorithm that finds an acyclic edge coloring of a graph with maximum degree  $\Delta$ , using  $2\Delta - 1$  colors. This is the best, until today, proven bound for the acyclic chromatic index of a graph. This work is still unpublished and can be found in [146].
- (v) We design a probabilistic algorithm that constructs  $c$ -separating codes of positive rate. To make the algorithm polynomial to the length of the produced code, we are forced to obtain codes of worse rate, but still better than other well known and extensively used ones. The results for the case where  $c = 2$  have appeared in [91], while the generalization presented here is still unpublished.

All these results are presented in Ch. 6.

- II. In the field of judgment aggregation, we characterize domains where various notions of non-dictatorial aggregations are possible and we show how to identify such domains.
  - (i) We characterize possibility domains as domains that admit either binary non-dictatorial aggregators, or majority or minority aggregators, both in the Boolean and non-Boolean framework. We also characterize totally blocked domains as domains that do not admit any binary non-dictatorial aggregators. Then, we define a subclass of possibility domains we call *uniform possibility domains*, which we characterize as domains admitting WNU aggregators. Furthermore, we show that a multi-sorted CSP defined over a uniform possibility domain is tractable, whereas otherwise it is NP-complete. These are the main results in [141] and are included in an extended version published in [144].
  - (ii) We characterize possibility and local possibility domains in the Boolean framework via the syntactic types of propositional formulas that describe them. We also show how to identify such formulas and how to construct them given such a domain. Furthermore we extend these results to domains admitting various types of non-dictatorial aggregators, like anonymous, monotone and systematic aggregators. These results appeared in [71] and are included in an extended version published in [72].

- (iii) We consider the problem of deciding if a domain is a possibility or a uniform possibility domain. In case the domain is given explicitly, we design polynomial time algorithms that solve this problem. We also show that this problem can be expressed in *transitive closure logic* and thus, if the domain is Boolean, it is in NLOG. Furthermore, again in the Boolean domain, we obtain upper and lower complexity bounds in case the domain is given implicitly, either as the truth set of a propositional formula or via an agenda. Finally, we obtain analogous bounds for other types of non-dictatorial aggregation, as in (ii) above. Some preliminary results have been presented in [143], while an extended work including all the above is still unpublished and can be found in [142].

These results are presented in Ch. 7.

First, regarding the probabilistic approach, we work exclusively in the Moser-Tardos variable framework, following the direct probabilistic approach of Giotis et al. [104,107]. Our algorithms progressively sample and resample the random variables in a structured way, until they find a point in the probability space such that no undesirable event occurs, given that one exists. As is the case with all extant algorithmic approaches to the LLL, both the number of events  $m$  and the number of random variables  $l$  are assumed to be constants. Complexity considerations are made with respect to the number of steps the algorithms last.

We begin by defining some novel relations of *directed* dependency that we call *variable directed lopsidedependency (VDL)* and *d-dependency*, which are both stronger than the lopsidedependency relation of Moser and Tardos [172]. We also show that *d-dependency* may generate a strictly sparser dependency graph than other extant ones (and so it leads to weaker sufficient conditions for LLL). We then algorithmically prove that the lopsidedependency condition suffices to avoid all events when applied to the graph defined by our new notions. Thus, in a sense we address the problem “find other simple local conditions for the constraints (in the variable framework) that advantageously translate to some abstract lopsided condition” posed by Szegedy [206]. Specifically, we prove a symmetric version of the LLL, using the VDL dependency graph and the asymmetric LLL over the *d-dependency* graph.

Our approach is based on Moser’s original algorithm [171], which, upon resampling an event, checks its neighborhood for other occurring events.

Like in Giotis et al. [104, 107], we use a witness structure (forest) to depict the execution of our algorithms that, in contrast with those of the “Moser-Tardos-like” proofs, grows “forward in time”, meaning that it is constructed as an execution moves on. Taking advantage of this structure, we express the probability that the algorithm executes for at least  $n$  rounds by a *recurrence relation*. We subsequently solve this recurrence by specialized analytical means, and prove that it diminishes exponentially fast in  $n$ . Specifically, we employ the result of Bender and Richmond [19] on the multivariable Lagrange inversion formula. A positive aspect of this approach is that it provides an exponentially small bound for the probability of the algorithm to last for at least  $n$  steps (including to run intermittently) before it returns the desired result, in contrast to the entropic method that estimates the expected time of the algorithm to return a correct answer. We also note that, in contrast to Harvey and Vondrák [121], our proof for the directed LLL is independent of the one for Shearer’s lemma.

Finally, although we show that our notion of dependency can give stronger results than the classical Shearer’s lemma, we use our forward approach to prove this lemma for the  $d$ -dependency graph. An algorithmic proof for Shearer’s lemma for the ordinary dependency graph was first provided by Kolipaka and Szegedy [150], who actually gave a proof for the general case of arbitrary probability spaces. The latter result was strengthened by Harvey and Vondrák [121] for their notion of association graphs, again for general probability spaces. Our result is for the variable framework, but for the possibly sparser graph of  $d$ -dependency. Also, we give again a direct computation of an exponentially small upper bound to the probability of the algorithm to last for at least  $n$  steps. To carry out the computations in our forward approach, we employ Gelfand’s formula for the spectral radius of a matrix (see [126]).

We subsequently use this direct probabilistic approach to acyclically color the edges of graphs and to construct  $c$ -separating codes. In what concerns coloring graphs, we get rid of the necessity to initially have a separate palette of  $2(\Delta - 1)$  colors that guarantee properness by a simple idea: ignore properness altogether when choosing a color and design the Moser-type algorithm in a way to guarantee that no even-length cycle exists whose edges of the same parity have the same color (same parity edges are edges that are one edge apart in a cycle traversal — cycles in a non-proper coloring whose edges of the same parity have the same color could have all their edges with the same color); then repeat this process anew until a coloring that is proper

is obtained. We show that with high probability the algorithm halts within a polynomial number of steps. Thus we show that  $2\Delta - 1$  colors suffice to properly edge-color a graph so that no two colors cover any cycle. As already observed by Cai et al. [46], this bound is the best possible using a Moser-type algorithm, because such algorithms essentially try independent colorings of the edges, therefore  $2(\Delta - 1)$  differently colored edges may be coincident with an edge. For  $c$ -separating codes, although the straightforward application of those results leads to constructions of exponential complexity, we show how we can explicitly obtain codes better than the current known constructions, by appropriately changing the conditions required in the LLL.

In the field of judgement aggregation, we follow Szegedy and Xu’s idea of deploying the algebraic “toolkit” [207]. All our results assume multi-valued sets of possible votes that may vary from issue to issue.

Firstly, we prove that non-dictatorial aggregation is possible for all societies of some cardinality if and only if a non-dictatorial binary aggregator exists or a non-dictatorial ternary aggregator exists such that on every issue  $j$ , the corresponding component  $f_j$  of the aggregator is a majority operation, i.e., for all  $x$  and  $y$ , it satisfies the equations

$$f_j(x, x, y) = f_j(x, y, x) = f_j(y, x, x) = x,$$

or  $f_j$  is a minority operation, i.e., for all  $x$  and  $y$ , it satisfies the equations

$$f_j(x, x, y) = f_j(x, y, x) = f_j(y, x, x) = y.$$

For additional information about the notions of majority and minority operations, see Szendrei [208, p. 24].

We also show that a domain is totally blocked if and only if it admits no non-dictatorial binary aggregators; this result shows that the notion of a domain being totally blocked is, in a precise sense, a weak form of an impossibility domain and thus it explains why total blockedness appears in several previous characterization results.

After this, we introduce the notion of *uniform* non-dictatorial aggregator, which is an aggregator that on every issue, and when restricted to an arbitrary two-element subset of the votes for that issue, differs from all projection functions. The introduction of this notion was motivated by the fact that an aggregator can be non-dictatorial simply by choosing different dictators for two issues. Actually, this fact has also motivated numerous other

notions stronger than non-dictatorial aggregators. In the Boolean framework, uniform non-dictatorial aggregators coincide with the ones that are locally non-dictatorial. The latter notion was introduced by Nehring and Puppe [175].

We first compare uniform non-dictatorial aggregators to other aggregators with related properties, such as the *anonymous* aggregators [175], the *StrongDem* aggregator of Szegedy and Xu [207], and the *generalized* or *rolling* dictatorship of Grandi and Endriss [111, 113] and of Cariani et al. [49], respectively. Then we characterize the sets of feasible voting patterns that admit uniform non-dictatorial aggregators.

As a corollary of this characterization, we establish that, in the Boolean framework, a set  $X$  of feasible voting patterns admits an aggregator that is locally non-dictatorial of some arity if and only if it admits a ternary anonymous one, a result that, to the best of our knowledge, has not been obtained earlier (note that Nehring and Puppe [175] prove the same result, but with the added hypothesis that the aggregators satisfy *monotonicity*, and without showing, in the “only if” direction, that the anonymous one is ternary).

Furthermore, by using Bulatov’s dichotomy theorem for conservative constraint satisfaction problems [14, 37, 38], we connect social choice theory with the computational complexity of constraint satisfaction. Specifically, we prove that if a set of feasible voting patterns  $X$  has a uniform non-dictatorial aggregator of some arity, then the multi-sorted conservative constraint satisfaction problem on  $X$ , in the sense introduced by Bulatov and Jeavons [40], with each issue representing a sort, is tractable; otherwise it is NP-complete. We believe that the connection of social choice theory with the constraint satisfaction problem may lead to still further characterization results.

Turning our attention exclusively to Boolean domains, we consider the framework of integrity constraints. It is a well known fact from elementary Propositional Logic that for every subset  $X$  of  $\{0, 1\}^m$ ,  $m \geq 1$ , i.e. for every domain, there is a Boolean formula in Conjunctive Normal Form (CNF) whose set of satisfying truth assignments, or models, denoted by  $\text{Mod}(\phi)$ , is equal to  $D$  (see e.g. Enderton [80, Theorem 15B]). Zanuttini and Hébrard [219] give an algorithm that finds such a formula and runs in polynomial-time with respect to the size of the representation of  $X$  as input. Following Grandi and Endriss [112], we call such a  $\phi$  an *integrity constraint* and think of it as expressing the “rationality” of  $X$  (the term comes from databases, see e.g. [79]).

We prove that a domain is a possibility domain, if and only if it admits an integrity constraint of a certain syntactic form to be precisely defined, which we call a *possibility* integrity constraint. Very roughly, possibility integrity constraints are formulas that belong to one of three types, the first two of which correspond to “easy” cases of possibility domains: (i) formulas whose variables can be partitioned into two non-empty subsets so that no clause contains variables from both sets that we call *separable* and (ii) formulas whose clauses are exclusive OR’s of their literals (*affine* formulas). The most interesting third type is comprised of formulas such that if we change the logical sign of some of their variables, we get formulas that have a Horn part and whose remaining clauses contain only negative occurrences of the variables in the Horn part. We call such formulas *renamable partially Horn*, whereas we call *partially Horn*<sup>1</sup> the formulas that belong to the third type without having to rename any variables. Furthermore, we show that the unified framework of Zanuttini and Hébrard [219] for producing formulas of a specific type that describe a given domain, and which entails the notion of prime formulas (i.e. formulas that we cannot further simplify its clauses) works also in the case of possibility integrity constraints. Actually, in addition to the syntactical characterization of possibility domains, we give two algorithms: the first on input a formula decides whether it is a possibility integrity constraint in time linear in the length of the formula (notice that the definition of possibility integrity constraint entails searching over all subsets of variables of the formula); the second on input a domain  $X$  halts in time polynomial in the size of  $X$  and either decides that  $X$  is not a possibility domain or otherwise returns a possibility integrity constraint that describes  $X$ . It should be noted that the satisfiability problem remains NP-complete even when restricted to formulas that are partially Horn. On the other hand, in Computational Social Choice, domains are considered to be non-empty.

We then consider *local possibility domains*, that is, domains admitting IIA aggregators whose components are all different than any projection function. Such aggregators are called *locally non-dictatorial* (see [175]). We show that local possibility domains are described by formulas we call *local possibility integrity constraints* and again, we provide a linear algorithm that checks if a formula is a local possibility integrity constraint and a polynomial algorithm

---

<sup>1</sup>A weaker notion of Horn formulas has appeared before in the work of Yamasaki and Doshita [218]; however our notion is incomparable with theirs, in the sense that the class of partially Horn formulas is neither a subset nor a superset (nor equal) to the class  $\mathbb{S}_0$  they define.

that checks if a domain is a local possibility one and, in case it is, constructs a local possibility integrity constraint that describes it. As a corollary we also obtain a simpler characterization of local possibility domains in the Boolean framework.

There are various notions of non-dictatorial aggregation, apart from the above, that have been introduced in the field of Aggregation Theory. First, we consider domains that admit aggregators which are not *generalized dictatorships*. A  $k$ -ary aggregator is a generalized dictatorship that, on input any  $k$  vectors from a domain  $D$ , always returns one of those vectors as its output. These aggregators are a natural generalization of the notion of dictatorial aggregators, in the sense that they select a possibly different “dictator” for each set of  $k$  feasible voting patterns, instead of a single global one. They were introduced by Cariani et al. [49] as *rolling dictatorships*, under the stronger requirement that the above property holds for any  $k$  vectors of  $\{0, 1\}^n$ . In that framework, Grandi and Endriss [112] showed that generalized dictatorships are exactly those functions that are aggregators for every Boolean domain. In this work, we show that domains admitting aggregators which are not generalized dictatorships are exactly the possibility domains (apart from some trivial cases), and are thus described by possibility integrity constraints.

Then, we consider *anonymous* aggregators, which are aggregators that are not affected by permutations of their input and *monotone* aggregators, which are aggregators that do not change their output if a voter changes his choice in order to agree with it. Both of these types of aggregators have been extensively studied in the bibliography (see e.g. [75, 76, 111, 112, 144, 162, 175]), as they have properties that are considered important, if not necessary, for democratic voting schemes. Here, we show that domains admitting anonymous aggregators are described by local possibility integrity constraints, while domains admitting non-dictatorial monotone aggregators by separable or renamable partially Horn formulas.

We also consider another kind of non-dictatorial aggregator that shares an important property of majority voting. *StrongDem* aggregators are  $k$ -ary aggregators that, on every issue, we can fix the votes of any  $k - 1$  voters in such a way that the  $k$ -th voter cannot change the outcome of the aggregation procedure. These aggregators were introduced by Szegedy and Xu [207]. Here, we show that domains admitting StrongDem aggregators are described by a subclass of local possibility integrity constraints.

Finally, we consider aggregators satisfying *systematicity* (see List [162]).



Aggregators are called systematic when they aggregate every issue with a common rule. This property has appeared also as *(issue-)neutrality* in the bibliography (see e.g. Grandi and Endriss [112] and Nehring and Puppe [175]). By viewing a domain  $D$  as a Boolean relation, systematic aggregators are in fact *polymorphisms* of  $D$  (see Subsec. 7.2.4 and 7.3.5). Polymorphisms are a very important and well studied tool of Universal Algebra. Apart from showing, using known results, that domains admitting systematic aggregators are described by specific types of local possibility integrity constraints, we also examine how our previous results concerning the various kinds of non-dictatorial voting schemes are affected by requiring that the aggregators also satisfy systematicity.

As examples of similar classical results in the theory of Boolean relations, we mention that domains component-wise closed under  $\wedge$  or  $\vee$  have been identified with the class of domains that are models of Horn or dual-Horn formulas respectively (see Dechter and Pearl [68]). Also it is known that a domain is component-wise closed under the ternary sum mod 2 if and only if it is the set of models of a formula that is a conjunction of subformulas each of which is an exclusive OR (the term “ternary” refers to the number of bits to be summed). Finally, a domain is closed under the ternary majority operator if and only if it is the set of models of a CNF formula where each clause has at most two literals. The latter two results are due to Schaefer [192]. The ternary majority operator is the ternary Boolean function that returns 1 on input three bits if and only if at least two of them are 1. It is also known that the respective formulas for each case can be found in polynomial time with respect to the size of  $D$  (see Zanuttini and Hébrard [219]).

Our results can be interpreted as verifying that various kinds of non-dictatorial voting schemes can always be generated by integrity constraints that have a specific, easily recognizable syntactic form. This can prove valuable for applications in the field of judgment aggregation, where relations are frequently encountered in compact form, as the sets of models of integrity constraints. As examples of such applications, we mention the work of Pigozzi [183] in avoiding the *discursive dilemma*, the characterization of *safe agendas* by Grandi and Endriss [111], that of Endriss and de Haan [82] concerning the *winner determination problem* and the work of Endriss et al. [83] in succinct representations of domains. Our proofs draw from results in judgment aggregation theory as well as from results about propositional formulas and logical relations. Specifically, as stepping stones for our algorithmic syntactic characterization we use three results. First, a theorem

implicit in Dokow and Holzman [75] stating that a domain is a possibility domain if and only if it either admits a binary (of arity 2) non-dictatorial aggregator or it is component-wise closed under the ternary direct sum. This result was generalized by Kirousis et al. [144] for domains in the non-Boolean framework. Second, a characterization of local possibility domains proven by Kirousis et al. in [144]. Lastly, the “unified framework for structure identification” by Zanuttini and Hébrard [219].

The aforementioned investigations have characterized possibility domains (in both the Boolean and the non-Boolean frameworks) in terms of *structural* and *syntactical* conditions. We now investigate possibility domains using the *algorithmic lens* and, in particular, we study the following algorithmic problem: given a set  $X$  of feasible evaluations, determine whether or not  $X$  is a possibility domain. Szegedy and Xu give algorithms for this problem [207, Theorem 36], but these algorithms have very high running time; in fact, they run in exponential time in the number of issues and in the number of positions over each issue, even when confined to the Boolean framework.

We design a polynomial-time algorithm that, given a set  $X$  of feasible evaluations (be it in the Boolean or the non-Boolean framework), decides whether  $X$  is a possibility domain. Furthermore, if  $X$  is a possibility domain, then the algorithm produces a binary non-dictatorial or a ternary non-dictatorial aggregator for  $X$ .

The first step towards this result is to show that there is a polynomial-time algorithm that given a set  $X$  of feasible evaluations, decides whether  $X$  admits a binary non-dictatorial aggregator (as mentioned earlier, this amounts to  $X$  not being totally blocked). In fact, we show a stronger result, namely, that this problem is expressible in *Transitive Closure Logic (TCL)*, an extension of *first-order logic* with the *transitive closure* operator; see [161] for the precise definitions. As a consequence, the problem of deciding whether  $X$  admits a binary non-dictatorial aggregator is in NLOGSPACE. Using this result, we then show that the problem of deciding whether a set  $X \subseteq \{0, 1\}^n$  is a possibility domain is in NLOGSPACE.

After this, we give a polynomial-time algorithm for the following decision problem: given a set  $X$  of feasible evaluations (be it in the Boolean or the non-Boolean framework), determine whether or not  $X$  is a uniform possibility domain; moreover, if  $X$  is a uniform possibility domain, then the algorithm produces a suitable uniform non-dictatorial aggregator for  $X$ .

We also study the problems of non-dictatorial and uniform non-dictatorial aggregation in case  $X$  is provided via an integrity constraint or by an agenda.

In both cases, we provide bounds for the computational complexity of deciding if  $X$  is a (uniform) possibility domain. Finally, we extend our results to three types of aggregators that have been used in the bibliography: generalised dictatorships [72,112], anonymous, monotone and systematic aggregators.

These results contribute to the developing field of computational social choice and pave the way for further exploration of algorithmic aspects of vote aggregation. In a sense, the question we investigate is the following: given a specific voting scheme, where some pre-defined rules of logical consistency apply, how difficult is it to decide if it is possible to design an aggregation rule with some desired properties, and construct it in case it is? It should be noted that in the field of Judgment Aggregation, quite frequently the domains that impose the logical consistency restrictions are fixed. In such a setting, the algorithmic approach does not have much to offer, since in this case we have a one-off problem. On the other hand, it is not difficult to imagine groups of people that constantly need to make collective decisions over different sets of issues, where the logical restrictions that apply change according to the dependencies between the issues. In this scenario, algorithms that can quickly decide which aggregation rules can be applied could be useful.

In terms of the applicability of our algorithms in the abstract setting, there is an issue with the size of the input, since a domain  $X$  given explicitly as a set of  $m$ -ary vectors, can be too large for practical purposes. However, as small or large the domain might be, the search space of its possible aggregators is exponentially larger, even provided characterization results like the ones in [75,144,207] that restrict the search to binary or ternary aggregators. Thus, our tractability results in the abstract framework should be interpreted as showing that, given access to the domain, one encounters no more problems in deciding whether non-dictatorial aggregation is possible. Furthermore, the algorithm for finding and producing binary non-dictatorial aggregators (or deciding their lack thereof), is used in obtaining parts of the complexity upper bounds in the cases where the domain is given in compact form.

## 1.3 Complementary Material

We present now various notions in Graph Theory, Propositional Logic, Algorithmic Design and Computational Complexity. Some of these notions will

be used in the sequel and others are meant to provide some context to the reader that is not familiar with these fields. In the last subsection we briefly discuss some advanced analytic and algebraic tools we use. This discussion far exceeds the scope of our work, so we include it here just for completeness.

### 1.3.1 Graph Theory

Graph Theory has a central role in computer science and especially in algorithm design, since we can model various aspects of our problems by graphs and use their reach theory to extract solutions. We provide here a very brief overview of some basic notions in this field and the notation we use. The interested reader can refer to any introductory textbook, such as Diestel [73].

A graph  $G$  is a pair  $(V, E)$ , where  $V$  is a finite set of points, called the vertices of the graph, and  $E \subseteq V \times V := \{(u, v) \mid u, v \in V\}$  is the set of edges. When we consider multiple graphs at the same time, we write  $V(G)$  and  $E(G)$  to denote the vertex and edge set of  $G$  respectively.

$(u, v) \in E$  means that there is a *directed* edge, starting at  $u$  and ending in  $v$ . Such graphs are called directed. If  $G$  is such that  $(u, v) \in E$  if and only if  $(v, u) \in E$ , we say that it is *undirected*, and we denote the edge between  $u$  and  $v$  by  $\{u, v\}$ .

An edge  $\{u, u\}$  is called a *loop*. Also, if  $G$  can have multiple edges between two vertices, we say that it is a *multigraph*. A graph is *simple*, if it has neither loops nor multiple edges. All the graphs we consider here are simple, unless explicitly stated otherwise.

A *path* is a finite sequence of pairwise distinct vertices  $u_1, \dots, u_k$ , such that  $(u_i, u_{i+1}) \in E$ ,  $i = 1, \dots, k - 1$  and it is denoted by  $(u_1, \dots, u_k)$ . A *cycle* is a path whose last vertex has an edge towards the first. The *length* of a path or cycle is the number of its edges. A path (resp. cycle) with length  $k$  is sometimes called  $k$ -path (resp.  $k$ -cycle).

A graph  $G$  is *connected* if, for any  $u, v \in V(G)$ , there is either a path from  $u$  to  $v$  or vice versa. It is *strongly* connected if both these paths exist. In undirected graphs, these notions coincide. A (*strongly*) *connected component* (*scc*) of a graph  $G$  is a maximal (strongly) connected subset of  $V$ . A *tree* is a connected and *acyclic* (i.e. with no cycles) graph. A possibly unconnected, but acyclic graph is a *forest*.

Let  $G = (\{1, \dots, m\}, E)$  be a simple undirected graph. We define  $N_j$  to

be the *neighborhood* of  $j$  in  $G$ , that is:

$$N_j := \{i \mid \{i, j\} \in E\}.$$

Note that since  $G$  is assumed to be simple,  $j \notin N_j$ ,  $j = 1, \dots, m$ . We denote by  $N_j^+ := N_j \cup \{j\}$  the *extended* neighborhood of  $j \in \{1, \dots, m\}$ . The *degree*  $\deg(j)$  of  $j$ , is the size  $|N_j|$  of its neighborhood. We denote by  $\Delta$  the maximum degree of  $G$ .

A *clique* of size  $n$  ( $n$ -clique), is a simple graph on  $n$  vertices, where each pair of distinct vertices are connected by an undirected edge. An *independent set* of size  $n$ , is a simple graph comprised of  $n$  vertices with no edges between them. An edge (resp. vertex) *coloring* is a function  $f : E \mapsto \{1, \dots, K\}$  (resp.  $f : V \mapsto \{1, \dots, K\}$ ), where  $\{1, \dots, K\}$  is the set of available colors. If  $|K| = k$ , we sometimes say that we have a  $k$ -coloring.

### 1.3.2 Propositional Logic

Propositional Logic is the most basic form of mathematical logic, where we consider true/false or “1/0” sentences. It corresponds to CSPs defined on Boolean domains and it is one of the most extensively studied parts of computer science. Again, we provide here some basic definitions and notation that can be found in any introductory textbook, like Enderton [80].

To avoid tedious technicalities, we assume we have a finite set of variables  $\{x_1, \dots, x_m\}$ , where  $m$  is as large as needed. A *propositional formula* (or simply *formula*)  $\phi$  has one of the following three forms:

- $\phi = x_i$ , for some  $i \in \{1, \dots, m\}$ ,
- $\phi = \neg\psi$ , for some already defined formula  $\psi$ , or
- $\phi = \psi * \chi$ , where  $*$   $\in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$  and  $\psi, \chi$  are already defined propositional formulas.

Thus, propositional formulas are defined recursively. We start with plain variables and create more complicated formulas via the *connectives*  $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ . Syntactically, to denote the order of operations, we either use parenthesis or we follow the rule:  $\neg$  comes before any other connective, then  $\wedge, \vee$  and finally  $\rightarrow, \leftrightarrow$ . Thus, the formula:

$$x_1 \wedge \neg x_2 \rightarrow x_3 \vee x_4,$$

is the same with:

$$((x_1 \wedge (\neg x_2)) \rightarrow (x_3 \vee x_4)).$$

We use the usual semantics for our formulas. Given an *assignment of values*  $\mathbf{a} = (a_1, \dots, a_n)$ ,  $a_i \in \{0, 1\}$ ,  $i = 1, \dots, n$  to the variables, we have that:

- if  $\phi = x_i$ , then  $\phi(\mathbf{a}) = 1$  if and only if  $x_i = 1$ ,
- if  $\phi = \neg\psi$ , then  $\phi(\mathbf{a}) = 1$  if and only if  $\psi(\mathbf{a}) = 0$ ,
- if  $\phi = \psi \wedge \chi$ , then  $\phi(\mathbf{a}) = 1$  if and only if  $\psi(\mathbf{a}) = \chi(\mathbf{a}) = 1$ ,
- if  $\phi = \psi \vee \chi$ , then  $\phi(\mathbf{a}) = 0$  if and only if  $\psi(\mathbf{a}) = \chi(\mathbf{a}) = 0$ ,
- if  $\phi = \psi \rightarrow \chi$ , then  $\phi(\mathbf{a}) = 0$  if and only if  $\psi(\mathbf{a}) = 1$  and  $\chi(\mathbf{a}) = 0$  and,
- if  $\phi = \psi \leftrightarrow \chi$ , then  $\phi(\mathbf{a}) = 1$  if and only if  $\psi(\mathbf{a}) = \chi(\mathbf{a})$ .

A formula  $\phi$  *tautologically entails*  $\psi$ , denoted  $\phi \models \psi$ , if any assignment of values that *satisfies*  $\phi$  (i.e.  $\phi$  evaluates to 1), satisfies  $\psi$  too. We extend this definition to sets of formulas  $\Sigma$ , where an assignment of values satisfies  $\Sigma$  if and only if it satisfies every formula in  $\Sigma$ .

We say that  $\phi$  is *tautologically equivalent* (or simply *equivalent*) with  $\psi$ , denoted by  $\phi \equiv \psi$ , if  $\phi \models \psi$  and  $\psi \models \phi$ . Two equivalent formulas are satisfied by exactly the same assignments, thus semantically they do not differ. Syntactically though they are not. For example, the length of  $x_1$  is much smaller than that of the equivalent formula:

$$\underbrace{x_1 \wedge \dots \wedge x_1}_{k\text{-times}}$$

when  $k > 1$ . This might seem like a triviality, but we will see later on that, for algorithmic purposes, the length of a formula can matter a lot.

We write  $\phi(x_1, \dots, x_m)$  to denote that  $\phi$  is defined over the variables  $x_1, \dots, x_m$ . This notation defines the length of the assignments that  $\phi$  admits. That is, the *set of satisfying assignments* or *truth set* of  $\phi(x_1, \dots, x_m)$  is:

$$\text{Mod}(\phi) \subseteq \{0, 1\}^m.$$

Note that writing e.g.  $\phi(x_1, \dots, x_n) = x_1 \wedge x_2$  is not a contradiction, since it is equivalent with the formula:

$$(x_1 \wedge x_2) \wedge \bigwedge_{i=3}^m (x_i \vee \neg x_i).$$

It thus holds that:

$$\text{Mod}(\phi) = \{(1, 1)\} \times \{0, 1\}^{m-2}.$$

### 1.3.3 Algorithms

We provide here a very brief discussion concerning the computational complexity of an algorithm and of a problem. We assume that the reader is already familiar with these concepts, which can be found in any introductory textbook of Algorithm Design or Computational Complexity (see for example [148, 193]).

We loosely refer to an *algorithm*  $A$  as a set of specific instructions, where given an input  $x$ , after a finite number of steps produce the output  $A(x)$ . The corresponding formal model is the *Turing Machine*, but we will not go into any details here. The interested reader is again referred to introductory textbooks (see for example [177, 199]). We assume that the input  $x$  is encoded in some suitable way (e.g. a binary representation), and we let  $x$  stand for both the input and its representation. The length of the input  $x$  is denoted by  $|x|$ .

The complexity of an algorithm  $A$  is the time or memory consumption it needs to produce its output and it is measured in terms of the input's length. We consider, unless explicitly stated otherwise, the time complexity of our algorithms, and we take a "worst-case" approach. This means that we measure the time  $A$  needs to produce its output on the worst possible input  $x$ . The complexity of a problem, is the complexity of the best algorithm that solves it. We make the usual assumption where an algorithm is deemed efficient if the time it needs to produce its output in the worst case is a polynomial over the length of its input. The class of problems admitting such an algorithm is P.

We begin with a very brief exposition of the Complexity Classes we will need. For the interested reader, most introductory textbooks in the field of Computational Complexity would do. We recommend [10, 177].

One way to depict a problem, is as a set of *words*, that is finite sequences of symbols, over a finite alphabet  $\Sigma$ . Let  $\Sigma^*$  be the set of words comprised entirely of symbols from  $\Sigma$ . The words are in fact encodings of the computational objects we study. Thus, a *decision* problem can be seen as a subset  $L \subseteq \Sigma^*$  and the question we want to answer algorithmically is whether a word  $x \in \Sigma^*$  is an element of  $L$  or not. Decision problems have their corre-

sponding function problems, where we want to optimize some quantity. For example, the decision version of the independent set problem is to inquire whether a graph has  $k$  independent vertices. The corresponding function problem is, given a graph, to find an independent set with as many vertices are possible.

**Determinism** The computational complexity of a problem is defined as the resources that the “best” (known) algorithm for that problem spends. Assume again that an algorithm  $A$  is loosely defined as a process comprised of a finite set of well defined and ordered steps, that, on input some  $x \in \Sigma^*$ , produces some output  $A(x)$ , which is usually assumed to be 0 or 1, denoting rejection and acceptance of the input respectively. For the time being, we consider deterministic algorithms, that is, algorithms whose  $i$ -th step is completely and uniquely determined by the input and the state the algorithm is at, after completing the  $(i - 1)$ -th step.

An algorithm  $A$  solves the problem  $L$  if and only if, for all  $x \in \Sigma^*$ ,  $A$  accepts  $x$  if and only if  $x \in L$ . The complexity of  $A$  is measured in terms of the length  $|x|$  of  $x$ , i.e. the number of its symbols. Although in practice every detail counts, for theoretical purposes, we do not care about the constants in the complexity of an algorithm. Thus, we say for example that algorithm  $A$  has  $O(N^2)$  time complexity and  $O(N \log N)$  space complexity, if on input  $x$  of length  $|x| = N$ ,  $A$  needs  $c_1 N^2$  steps to terminate and  $c_2 N \log N$  places in memory to perform its operations, where  $c_1$  and  $c_2$  are constants. This is usually referred to as “Big-O notation”, and can be found in any introductory textbook in this scientific area, like [177].

We express the time/space consumption of an algorithm  $A$  by non decreasing and *computable* functions. A computable function is one that can be produced by an algorithm. We will not get into details here. For our purposes, it suffices to know that logarithms, polynomials and exponentials are all functions that can express the complexity of an algorithm.

Let  $\text{DTIME}(f(N))$  be the complexity class that contains all problems  $L$  that can be solved by a deterministic algorithm  $A$  in  $O(f(N))$  time. The class of *polynomially solvable* or *tractable* problems is defined as:

$$\mathbf{P} := \bigcup_{k \in \mathbb{N} \setminus \{0\}} \text{DTIME}(N^k).$$

We consider problems in  $\mathbf{P}$  as the ones that can be solved efficiently, with respect to the time an algorithm needs to solve them. Analogously, we can



define problems that can be solved by exponential-time algorithms as:

$$\text{EXP} := \bigcup_{k \in \mathbb{N} \setminus \{0\}} \text{DTIME}(2^{N^k}).$$

Easily, any problem in  $\text{P}$  also belongs in  $\text{EXP}$ .

We can define analogous complexity classes concerning memory consumption. Let  $\text{DSPACE}(f(N))$  be the complexity class that contains all problems  $L$  that can be solved by a deterministic algorithm  $A$  in  $O(f(N))$  space. The class of problems that can be solved in logarithmic space is

$$\text{LOG} = \text{DSPACE}(\log(N)),$$

where  $\log$  is assumed to have base 2, unless specifically stated otherwise.  $\text{LOG}$  is often denoted as  $\text{LOGSPACE}$  in the bibliography. The class of problems that can be solved in polynomial space is:

$$\text{PSPACE} := \bigcup_{k \in \mathbb{N} \setminus \{0\}} \text{DSPACE}(N^k).$$

It holds that:

$$\text{LOG} \subseteq \text{P} \subseteq \text{PSPACE} \subseteq \text{EXP} \tag{1.5}$$

and, although we do not know anything more than that  $\text{LOG} \subsetneq \text{PSPACE}$  and  $\text{P} \subsetneq \text{EXP}$ , we suspect that all containments are proper.

**Non-determinism** Non-determinism is a computational model that cannot be implemented in the real world. Nevertheless, it is quite useful for theoretical purposes and has been widely studied. There are many ways that have been used to describe non-deterministic machines, e.g. as machines that guess what is the right thing to do or that do everything at the same time. We believe that to avoid the confusion arising by comparing non-determinism with other computational models, like oracle machines or parallel computation, the best way is to say that a non-deterministic machine creates “alternate universes” to go through different computational scenarios.

In contrast with deterministic algorithms, a non-deterministic one can have more than one choices to proceed, when in a given state and reading a particular symbol of its input. Such an algorithm accepts its input if at least one of its computational paths leads to success and rejects otherwise. The

time/space consumption of such an algorithm is measured as the time/space consumption of its longest computational path.

A simple example is the following. There is a non-deterministic algorithm that solves SAT, where, on input a formula  $\phi(x_1, \dots, x_n)$  proceeds as follows:

1. Non-deterministically choose an assignment  $\mathbf{a}$ .
2. If  $\phi(\mathbf{a}) = 1$ , accept; else, reject.

This might seem either nonsensical or extremely powerful (or both). For the latter, we will see in the sequel that in some cases, non-determinism does not provide any more computational power. Interestingly, even in the cases it presumably adds computational power, we cannot yet prove it. For the former, we will see some alternative definitions of non-deterministic classes that make both its theoretical and practical purposes clear.

Let  $\text{NTIME}(f(n))$  be the complexity class that contains all problems  $L$  that can be solved by a non-deterministic algorithm  $A$  in  $O(f(n))$  time. The class of problems that can be solved by a polynomial time non-deterministic algorithm is defined as:

$$\text{NP} := \bigcup_{k \in \mathbb{N} \setminus \{0\}} \text{NTIME}(N^k).$$

Analogously, we can define problems that can be solved by non-deterministic exponential-time algorithms as:

$$\text{NEXP} := \bigcup_{k \in \mathbb{N} \setminus \{0\}} \text{NTIME}(2^{N^k}),$$

which is obviously a super-class of NP. Again, we can define complexity classes concerning memory consumption. Let  $\text{NSPACE}(f(N))$  be the complexity class that contains all problems  $L$  that can be solved by a non-deterministic algorithm  $A$  in  $O(f(N))$  space. The class of problems that can be solved in non-deterministic logarithmic space is

$$\text{NLOG} = \text{NSPACE}(\log(N)),$$

where  $\log$  is assumed to have base 2, unless specifically stated otherwise. The class of problems that can be solved in non-deterministic polynomial space is:

$$\text{NSPACE} := \bigcup_{k \in \mathbb{N} \setminus \{0\}} \text{NSPACE}(N^k).$$

It holds that:

$$\text{NLOG} \subseteq \text{NP} \subseteq \text{NPSpace} \subseteq \text{NEXP}. \quad (1.6)$$

The deterministic and non-deterministic classes relate to each other as follows:

$$\text{LOG} \subseteq \text{NLOG} \subseteq \text{P} \subseteq \text{NP} \subseteq \text{PSPACE} = \text{NPSpace} \subseteq \text{EXP} \subseteq \text{NEXP}. \quad (1.7)$$

For any computational class  $\mathcal{C}$ , let its *complementary* class be defined as follows:

$$\text{co}\mathcal{C} := \{L \mid \bar{L} \in \mathcal{C}\},$$

where, for any subset  $L \subseteq \Sigma^*$ ,  $\bar{L} := \Sigma^* \setminus L$ .

It is easy to see that all the deterministic polynomial classes are equal their complementary ones. Indeed assume that  $L \in \text{DTIME}(f(N))$ . Then, there is an algorithm  $A$  whose running time/space is  $O(f(N))$ , such that  $A(x) = 1$  if and only if  $x \in L$ . We can now design an algorithm  $B$  that performs the exact same steps as  $A$ , but in the end, it outputs 1 if and only if  $A(x) = 0$ . Then, it is immediate to see that  $\bar{L} \in \text{DTIME}(f(N))$  too.

With non-determinism though, things aren't that simple. One can see that by the definition of acceptance in a non-deterministic algorithm. It is known that:

$$\text{P} \subseteq \text{coNP} \cap \text{NP} \text{ and } \text{EXP} \subseteq \text{NEXP} \cap \text{coNEXP}.$$

The biggest question in computational complexity is where  $\text{P}$  equals  $\text{NP}$  or not, closely followed by whether  $\text{NP} = \text{coNP}$  or not and whether  $\text{P} = \text{NP} \cap \text{coNP}$ . A positive answer to the first question, implies positive answers to the other two as well. We suspect though that none of these equalities hold.

**Randomized Computation** Another computational model that permeates our work, is using randomness in the design of our algorithms. We will not get into formal details here. In general, a randomized algorithm is one that relies on “coin tosses” in order to perform some of its steps. An easy example of such an algorithm would again be for **SAT**, where, on input a formula  $\phi(x_1, \dots, x_n)$  proceeds as follows:

1. Choose an assignment  $\mathbf{a}$  uniformly at random.
2. If  $\phi(\mathbf{a}) = 1$ , accept; else, reject.

According to the random assignment that step 1 produces, the algorithm takes a different computational path. But, in contrast with non-determinism, here there are no alternative universes. We simply say that, if there are  $k$  assignments satisfying  $\phi$ , the probability of succeeding is  $\frac{k}{2^n}$ .

Usually, we allow randomness in an algorithm in two ways. Either we allow some chance of the algorithm to produce a wrong result, or we allow its execution time to be random and compute the *expected* running time of the algorithm. These methods give rise to the following computational classes.

A language  $L$  is in the class *Bounded-error Probabilistic Polynomial-time* (BPP), if there is a probabilistic algorithm  $A$  such that:

$$\Pr[A(x) = L(x)] \geq \frac{2}{3},$$

where  $L(x) = 1$  if and only if  $x \in L$ . If we allow only one-sided errors, we get the classes RP and its complement coRP, where  $L \in \text{RP}$  if there is a probabilistic algorithm  $A$  such that:

$$\begin{aligned} \text{if } x \in L \text{ then } \Pr[A(x) = 1] &\geq \frac{2}{3}, \\ \text{if } x \notin L \text{ then } \Pr[A(x) = 0] &= 1. \end{aligned}$$

Thus, RP contains problems that admit algorithms that can get it wrong in case  $x \in L$  only.

Finally,  $L$  is in ZPP if it admits a randomized algorithm such  $A$ , such that  $\Pr[A(x) = L(x)] = 1$  and its expected running time is polynomial. Thus, such algorithms make no errors, but may have longer execution times. The algorithms we present in Ch. 6 fall in fact in this class.

It holds that:

$$\text{ZPP} = \text{RP} \cap \text{coRP}, \quad \text{RP} \cup \text{coRP} \subseteq \text{BPP}.$$

Finally, we know that  $\text{BPP} \subseteq \text{EXP}$ , but nothing more. That is, we cannot even prove containment in NP. There are many researchers who believe that any randomized algorithm can be derandomized and thus that  $\text{P} = \text{BPP}$ . An interesting discussion as to why that may be true can be found in [10, Chapters 19,20].

**Polynomial Hierarchy** Going above P but remaining within the bounds of PSPACE, the *polynomial hierarchy* consists of the complexity classes  $\Sigma_k^P$ ,  $\Pi_k^P$  and  $\Delta_k^P$ ,  $k \in \mathbb{N}$ , which are recursively defined as follows:

- $\Sigma_0^P = \Sigma_0^P = \Delta_0^P = P$  and
- $\Sigma_{k+1}^P$  is NP with oracle  $\Sigma_k^P$ ,  $\Pi_{k+1}^P$  is coNP with oracle  $\Sigma_k^P$  and  $\Delta_{k+1}^P$  is P with oracle  $\Sigma_k^P$ ,  $k \in \mathbb{N}$ .

It is known that

$$\Sigma_k^P \cup \Pi_k^P \subseteq \Delta_{k+1}^P \subseteq \Sigma_{k+1}^P \cap \Pi_{k+1}^P, \quad \forall k \in \mathbb{N}.$$

Furthermore, if for some  $k \in \mathbb{N}$ , we have  $\Sigma_k^P = \Pi_k^P$ , then PH collapses to that level, in the sense that  $\Sigma_l^P = \Pi_l^P$ , for all  $l \geq k$ . For example, if  $\text{NP} = \text{coNP}$ , then  $\Sigma_k^P = \Pi_k^P = \text{NP}$ , for all  $k \geq 1$ . For a more in depth discussion of the polynomial hierarchy, we refer the interested reader to Stockmeyer's work [204]. Finally, there is the following alternative description of the polynomial hierarchy, via certificates:

$$\begin{aligned} \Sigma_{k+1}^P &= \exists \Pi_k^P = \{x \in \{0, 1\}^* \mid \exists w \in \{0, 1\}^{p(|x|)} : \langle x, w \rangle \in \Pi_k^P\}, \\ \Pi_{k+1}^P &= \forall \Sigma_k^P = \{x \in \{0, 1\}^* \mid \forall w \in \{0, 1\}^{p(|x|)} : \langle x, w \rangle \in \Pi_k^P\}, \end{aligned}$$

where  $p(|x|)$  is polynomial to the size of  $|x|$ . We will use both definitions of these complexity classes to derive our results.

**Hardness and Completeness** We end this subsection by defining hardness and completeness of a problem with respect to a computational class. We first need the central definition of a *reduction*. We say that a problem  $L$  *reduces* to a problem  $L'$  and write  $L \leq L'$ , if there is a function  $f : L \mapsto L'$  such that  $w \in L$  if and only if  $f(w) \in L'$ . Again we need  $f$  to be computable, but we will not go into such details here.  $L$  *polynomially* reduces to  $L'$ , denoted by  $L \leq_P L'$ , if there is a reduction  $f : L \mapsto L'$  that can be constructed in polynomial time. Intuitively, a reduction is a way to transform an instance of a given problem to one of another, while preserving membership to each problem.

Let  $L$  be a problem and  $\mathcal{C}$  a computational class.  $L$  is  $\mathcal{C}$ -*hard* if every problem of  $\mathcal{C}$  polynomially reduces to  $L$ . It is  $\mathcal{C}$ -*complete* if it is  $\mathcal{C}$ -hard and, furthermore,  $L \in \mathcal{C}$ . In general, when trying to bound the complexity of a given problem  $L$ , membership in a class corresponds to an upper bound, hardness in another class to a lower bound and completeness in a class to a tight bound.

Every level of PH contains complete problems that are generalizations of SAT. For example,  $\Sigma_k$ -SAT is complete for  $\Sigma_k^P$ , where  $\Sigma^k$ -SAT is the following decision problem: given an expression of the form

$$\exists \mathbf{x}_1 \forall \mathbf{x}_2 \dots Q \mathbf{x}_k \varphi(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k),$$

where  $\varphi$  is a Boolean formula, is this expression true when the quantifiers vary over the set  $\{0, 1\}$ ? (Here,  $Q = \exists$  if  $k$  is odd, while  $Q = \forall$  if  $k$  is even.)

### 1.3.4 Analytic and Algebraic Tools

In this section, we present some more advanced analytic and algebraic notions that we will need in the following chapters. We begin with some preliminaries concerning *ordinary* and *multivariate generating functions*, which we will use, along with Bender and Richmond’s “Lagrange Inversion Formula” [19] in order to analyse the algorithm presented in Ch. 6, Sec. 6.2. We then proceed with some information concerning *norms* defined on matrices, we define the *spectral radius* of a matrix and consider “Gelfand’s formula” [126], which we need for the algorithm of Sec. 6.3.

**Generating Functions** Generating functions are widely used in combinatorics in order to count the number of various objects, like formulas or graphs with given properties. The main reason they are so useful is that they can be viewed as analytic objects, allowing us to implement a wide and robust theory in order to analyze them. The results we need here can all be found in Flajolet and Sedgewick’s [94]. Let  $(a_n) := (a_n)_{n \geq 0}$  be an infinite sequence. The *Ordinary Generating Function* of  $(a_n)$  is:

$$A(x) := \sum_{n=0}^{\infty} a_n x^n. \quad (1.8)$$

Intuitively, if  $(a_n)$  is the *counting sequence* of a class of combinatorial objects, that is, if  $a_n$  denotes the number of objects of size  $n$ , then  $x$  *marks* that size in the OGF, since  $x^n$  appears as many times in Eq. (1.8) as there are objects of size  $n$ , for all  $n \in \mathbb{N}$ .

The coefficient of  $x^n$  in  $A(x)$  is denoted by  $[x^n]A(x)$ , in the sense that:

$$[x^n]A(x) = [x^n] \sum_{n=0}^{\infty} a_n x^n = a_n.$$

The first known appearance of OGFs was in 1751, in a letter by Euler to Goldbach [88], where he provides the OGF for  $n$ -gon *triangulations*, that is, the number of ways one can divide a convex  $n$ -gon to triangles. It is unclear however if Euler had a formal proof for his result.

A problem of interest that can be solved using formal power series is the *inversion problem*, where, given the equation  $z = h(y)$ , one wants to express  $y$  in terms of  $z$ . Assuming that  $[z^0]h(z) = 0$  and  $[z^1]h(z) \neq 0$ , the problems can be expressed as:

$$\phi(y) = \frac{y}{h(y)}.$$

The *Lagrange inversion formula* of 1970 (see [53, 124]) provides a non-elementary solution to this problem.

**Theorem 1.3.1** (Lagrange inversion). *Let:*

$$\phi(u) = \sum_{k \geq 0} \phi_k u^k$$

be a formal power series, with  $\phi_0 \neq 0$ . Then, the equation  $y = z\phi(y)$  admits a unique solution whose coefficients are given by:

$$y(z) = \sum_{n=1}^{\infty} y_n z^n, \text{ where } y_n = \frac{1}{n} [u^{n-1}] \phi(u)^n.$$

Furthermore, for each  $k > 0$ , the Bürmann form is:

$$y(z)^k = \sum_{n=1}^{\infty} y_n^{(k)} z^n, \text{ where } y_n^{(k)} = \frac{k}{n} [u^{n-k}] \phi(u)^n.$$

Finally, by linearity, we have:

$$[z^n]H(y(z)) = \frac{1}{n} [u^{n-1}] (H'(u) \phi(u)^n),$$

where  $H$  is an arbitrary function.

We have already mentioned that one of the main features of generating functions is that we can view them as analytic objects. Assume that we have a function  $f$  defined on a *region*  $\Omega$  of the *complex plane*, that is an *open* and *connecte* subset of  $\mathbb{C}$ .  $f$  is *analytic* at a point  $z_0 \in \Omega$  if, for all  $z$  in some

open disc of  $\Omega$  centered at  $z_0$ , it is representable by a convergent power series expansion:

$$f(z) = \sum_{n \geq 0} c_n (z - z_0)^n.$$

The  $c_n$  are referred to as *Taylor coefficients*, since by differentiating  $f$   $n$  times we obtain:

$$c_n = f^{(n)}(z_0) n!.$$

We now have the following result.

**Proposition 1.3.1** (Prop. IV.5 [94]). *Let  $\phi$  be a function analytic at 0, having non-negative Taylor coefficients, such that  $\phi(0) \neq 0$ . Let  $R \leq +\infty$  be the radius of convergence of the series representing  $\phi$  at 0. If:*

$$\lim_{R^-} \frac{x\phi'(x)}{x} > 1,$$

*there exists a unique solution  $\tau \in (0, R)$  of the characteristic equation:*

$$\frac{\tau\phi'(\tau)}{\phi(\tau)} = 1.$$

*Then, the formal solution  $y(z)$  of  $y(z) = z\phi(y(z))$  is analytic at 0 and its coefficients satisfy:*

$$\limsup_{n \rightarrow +\infty} ([z^n](y(z))) = \left(\frac{1}{\rho}\right),$$

*where:*

$$\rho = \frac{\tau}{\phi(\tau)} = \frac{1}{\phi'(\tau)}.$$

We return now to the Lagrange inversion formula, but in the case of multivariate generating functions. This case has been extensively studied: Bergeron et al. [20], Gessel [103], Goulden and Kulkarni [110], Haiman and Schmitt [116]. Consider now the following definitions.

The *Krocker delta* is the function:

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{otherwise.} \end{cases}$$

Let  $A$  be an  $n \times n$  matrix, where  $a_j^i$  is the element of the  $i$ -th row and  $j$ -th column,  $i, j \in \{1, \dots, n\}$ . Let also  $A_j^i$  be  $A$  without its  $i$ -th row and  $j$ -th



column. The *determinant*  $\det(A)$  of  $A$  can now be computed by the *Laplace expansion*:

$$\det(A) = \sum_{j=1}^n (-1)^{i+j} a_j^i \det(A_j^i), \text{ for } i = 1, \dots, n,$$

$$\det(A) = \sum_{i=1}^n (-1)^{i+j} a_j^i \det(A_j^i), \text{ for } j = 1, \dots, n,$$

where, for a  $2 \times 2$  matrix  $A$ , it holds that  $\det(A) = a_1^1 a_2^2 - a_2^1 a_1^2$ .

Let  $\mathbf{x} = (x_1, \dots, x_k)$  be a vector of length  $k$  and  $\mathbf{n} = (n_1, \dots, n_k)$  a vector of  $k$  natural numbers. Then:

$$\mathbf{x}^{\mathbf{n}} := x_1^{n_1} \cdots x_k^{n_k}.$$

The *multivariate* OGF of  $(a_{\mathbf{n}})$  is:

$$A(\mathbf{x}) := \sum_{\mathbf{n}=0}^{\infty} a_{\mathbf{n}} \mathbf{x}^{\mathbf{n}}. \quad (1.9)$$

We now have the usual formulation of the multivariate Lagrange inversion.

**Theorem 1.3.2** (Multivariate Lagrange inversion). *Assume that  $g(\mathbf{x})$  and  $\mathbf{f}(\mathbf{x}) := (f_1(\mathbf{x}), \dots, f_k(\mathbf{x}))$  are formal power series in  $\mathbf{x}$  such that  $f_i(\mathbf{0}) \neq 0$ ,  $i = 1, \dots, k$ . Then, the set of equations  $w_i = t_i f_i(\mathbf{w})$ ,  $i = 1, \dots, k$  uniquely determine the  $w_i$  as formal power series in  $\mathbf{t}$  and:*

$$[\mathbf{t}^{\mathbf{n}}]g(\mathbf{w}(\mathbf{t})) = [\mathbf{x}^{\mathbf{n}}] \left\{ g(\mathbf{x}) \mathbf{f}(\mathbf{x})^{\mathbf{n}} \det \left( \delta_{ij} - \frac{x_i}{f_j(\mathbf{x})} \frac{\partial f_j(\mathbf{x})}{\partial x_i} \right) \right\}.$$

The disadvantage of this formulation is that the determinant vanishes near the point where the integrand is maximized. Bender and Richmond's alternate formulation avoids this issue. Assume  $G = (V, E)$  is a directed graph on  $V = \{0, 1, \dots, k\}$  and:

$$\frac{\partial \mathbf{f}}{\partial G} := \prod_{j \in V} \left\{ \left( \prod_{(i,j) \in E} \frac{\partial}{\partial x_i} \right) f_j(\mathbf{x}) \right\}.$$

The proof of the following formulation can be found in [\[19\]](#).

**Theorem 1.3.3** (Bender and Richmond [19]). Assume that  $g(\mathbf{x})$  and  $\mathbf{f}(\mathbf{x}) := (f_1(\mathbf{x}), \dots, f_k(\mathbf{x}))$  are formal power series in  $\mathbf{x}$  such that  $f_i(\mathbf{0}) \neq 0$ ,  $i = 1, \dots, k$ . Then, the set of equations  $w_i = t_i f_i(\mathbf{w})$ ,  $i = 1, \dots, k$  uniquely determine the  $w_i$  as formal power series in  $\mathbf{t}$  and:

$$[\mathbf{t}^{\mathbf{n}}]g(\mathbf{w}(\mathbf{t})) = \frac{1}{\prod_{i=1}^k n_i} [\mathbf{x}^{\mathbf{n}-1}] \sum_{\mathcal{T}} \frac{\partial(g, f_1^{n_1}, \dots, f_k^{n_k})}{\partial \mathcal{T}},$$

where  $\mathbf{1} = (1, \dots, 1)$ ,  $\mathcal{T}$  contains all trees on  $\{0, 1, \dots, k\}$  with edges directed towards 0 and  $\partial/\partial \mathcal{T}$  is indexed from 0 to  $d$ .

**Matrix norms** We now proceed with some preliminaries concerning matrix norms. Our aim is to present *Gelfand's formula* on the spectral radius of matrices, which we will need in the analysis of the algorithm in Sec. 6.3. Here, we follow the Horn's "Matrix Analysis" [126].

We begin with the definition of a *norm*.

**Definition 1.3.1.** Let  $V$  be a vector space over a field  $F$  (usually  $\mathbb{R}$  or  $\mathbb{C}$ ). A function  $\|\cdot\| : V \mapsto \mathbb{R}$  is a *norm*, if the following conditions hold:

1.  $\|\mathbf{x}\| \geq 0$ , for all  $\mathbf{x} \in V$ ,
2.  $\|\mathbf{x}\| = 0$ , if and only if  $\mathbf{x} = \mathbf{0}$ ,
3.  $\|c\mathbf{x}\| = |c|\|\mathbf{x}\|$ , for all  $\mathbf{x} \in V$  and  $c \in F$  and
4.  $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ , for all  $\mathbf{x}, \mathbf{y} \in V$ .

Note that from any vector  $\mathbf{x}$ , we can take a *unit vector*  $\mathbf{y}$ , such that  $\|\mathbf{y}\| \leq 1$ , by taking  $\mathbf{y} = \frac{\mathbf{x}}{\|\mathbf{x}\|}$ . Some of the most common norms are the *absolute value*  $|\cdot|$  on  $\mathbb{R}$ , where, for all  $x \in \mathbb{R}$ :

$$|x| = \begin{cases} x, & \text{if } x > 0, \\ -x, & \text{else,} \end{cases}$$

the *Euclidean norm*  $\|\cdot\|_2$  on  $\mathbb{R}^n$ , where:

$$\|\mathbf{x} = (x_1, \dots, x_n)\|_2 = \sqrt{x_1^2 + \dots + x_n^2}$$

and the 1-norm (or *taxicab* norm or *Manhattan* norm)  $\|\cdot\|_1$  on  $\mathbb{R}^n$ , where:

$$\|\mathbf{x}\|_1 = \sum_{i=1}^n |x_i|.$$

Let  $M_n$  be the vector space of all  $n \times n$  matrices with elements from the field  $F$  (usually  $\mathbf{R}$  or  $\mathbf{C}$ ).

**Definition 1.3.2.** A function  $\|\cdot\| : M_n \mapsto \mathbb{R}$  is a (matrix-)norm (or ring-norm, if the following conditions hold:

1.  $\|A\| \geq 0$ , for all  $A \in M_n$ ,
2.  $\|A\| = 0$ , if and only if  $A$  is the all-zero matrix,
3.  $\|cA\| = |c|\|A\|$ , for all  $A \in M_n$  and  $c \in F$ ,
4.  $\|A + B\| \leq \|A\| + \|B\|$ , for all  $A, B \in M_n$  and
5.  $\|AB\| \leq \|A\|\|B\|$ , for all  $A, B \in M_n$ .

By any norm  $\|\cdot\|$  on  $\mathbf{R}^n$  (or on  $\mathbf{C}^n$ ), we can define the corresponding matrix-norm, called the *induced* norm, in the following way:

$$\|A\| := \sup_{\mathbf{x} \neq \mathbf{0}} \frac{\|Ax\|}{\|x\|}.$$

For the induced norm we have the following properties.

**Theorem 1.3.4.** Let  $\|\cdot\|$  be a norm on  $\mathbf{R}^n$ . We use the same notation for its induced norm on  $M_n$ . The following hold:

- (i)  $\|I\| = 1$ , where  $I$  is the identity matrix,
- (ii)  $\|Ax\| \leq \|A\|\|x\|$ , for all  $A \in M_n$  and  $\mathbf{x} \in \mathbf{R}^n$ .

The *spectral radius* of  $A \in M_n$  is:

$$\rho(A) := \max\{|\lambda| \mid \lambda \text{ is an eigenvalue of } A\}.$$

We can now state *Gelfand's formula*.

**Theorem 1.3.5** (Gelfand's formula). Let  $\|\cdot\|$  be a matrix norm on  $M_n$  and  $A \in M_n$ . Then:

$$\rho(A) = \lim_{k \rightarrow +\infty} \|A^k\|^{1/k}.$$



# Chapter 2

## Constraint Satisfaction Problems

In this Chapter, we provide a general overview of *constraint satisfaction problems* (CSPs), in the various ways that they appear in the bibliography. We begin by presenting the framework we use in Section 2.1 and, in Subsection 2.1.1, discuss a particular variant of CSP's, the *multi-sorted* CSP's. We then discuss CSP's in the field of *First Order Logic*, in Section 2.2 and, in Subsection 2.2.1, present an extension of First-Order Logic, the *Transitive Closure-Logic* (TCL). We proceed by discussing CSP's via *homomorphisms*, in Section 2.3 and Second-Order Logic in Section 2.4. All the relevant material is provided in the text. The reader is assumed to have some familiarity with Propositional and First-Order Logic (see e.g. Enderton [80]).

### 2.1 Our Framework

Let  $\mathcal{D}$  be a (possibly infinite) set, which we call the *domain*. A *relation*  $R$  with *arity*  $n$  is a subset of  $\mathcal{D}^n$ . In general, we allow both trivial relations, the empty relation  $\emptyset$  and the whole domain  $\mathcal{D}^n$ . The arity of  $R$  is sometimes denoted by  $\alpha(R)$ .

Let  $\mathcal{R}$  be a set of relations over  $\mathcal{D}$ . In the bibliography,  $\mathcal{R}$  is referred to as a (*constraint*) *language*. Let also  $V$  be a set of *variables*, taking values in  $\mathcal{D}$ . To avoid tedious technicalities, one usually assumes  $V$  is finite, but as large as needed. A *Constraint Satisfaction Problem* over  $\mathcal{R}$ , denoted by  $\text{CSP}(\mathcal{R})$  is defined as follows:

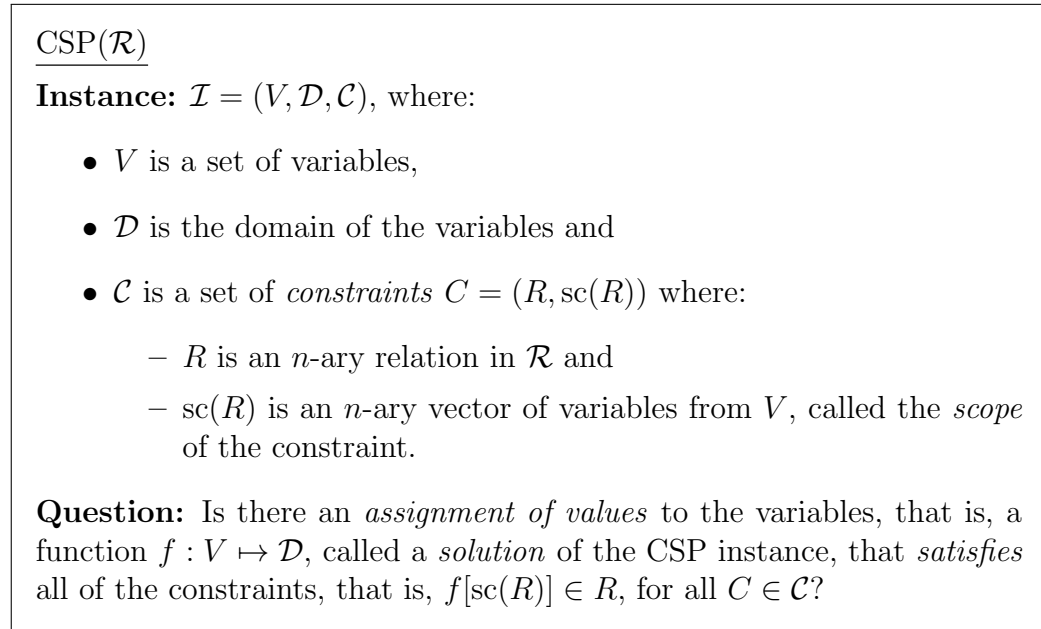


Figure 2.1: Constraint Satisfaction Problem over a set of relations  $\mathcal{R}$

The framework of CSP's is broad enough so that many computational problems of interest can be represented as a CSP problem. To illustrate this, we provide a toy example from the field of Graph Theory.

**Example 2.1.1.** Let  $G = (V, E)$  be a simple (no loops or multiple edges) and undirected graph. Suppose we are interested in coloring its edges with a palette of  $K \in \mathbb{N}_{>1}$  colors  $\{1, \dots, K\}$  and we require the resulting coloring to be proper, that is, no adjacent edges of the graph receive the same color.

Using a more formal terminology, we search for a function  $f : E \mapsto K$ , such that  $f(\{u, v\}) \neq f(\{z, w\})$ , for all edges  $\{u, v\}, \{z, w\}$  such that  $z$  or  $w \in \{u, v\}$ . Such a function can be obtained as a solution to an instance of  $\text{CSP}(\{\neq_K\})$ , where:

- $V = \{x_e \mid e = \{u, v\} \in E\}$ ,
- $\mathcal{D} = \{1, \dots, K\}$ ,
- $\mathcal{C}$  contains all constraints of the form  $(\neq_K, (x_{\{u,v\}}, x_{\{z,w\}}))$ , such that exactly one of  $z$  or  $w$  are in  $\{u, v\}$  and

- $\neq_K \subseteq K^2$  is the inequality relation  $\neq_K = \{(a, b) \in K^2 \mid a \neq b\}$ .

A solution to  $\text{CSP}(\neq_K)$  easily corresponds to a proper coloring with at most  $K$  colors.  $\diamond$

### 2.1.1 Multi-Sorted CSP's

We proceed with discussing a variant of CSP's, the *multi-sorted* CSP, or MCSP. The difference with the original definition is that, in an MCSP, there are different “sorts”, each being a “semi-autonomous” CSP, in the sense that it contains distinct variables that take values from a distinct domains. On the other hand, the constraints are defined over every sort, inducing dependencies between the solutions of each sort. Here, we follow the terminology of A. Bulatov and P. Jeavons [40].

Let  $\mathfrak{D} = \{\mathcal{D}_i \mid i \in I\}$  be a (possibly infinite) collection of (possibly infinite) sets. A *multi-sorted* relation  $R$  over  $\mathfrak{D}$ , with *signature*  $\sigma(R) = (i_1, \dots, i_k)$ , is a subset of  $\prod_{j=1}^k \mathcal{D}_{i_j}$ , where  $i_1, \dots, i_k \in I$  are not necessarily distinct. Note that the signature  $\sigma(R)$  provides the information of the domains that  $R$  is defined upon. The arity of  $R$  will again be denoted by  $\alpha(R)$ , and it is now defined as the *length* of  $\sigma(R)$ , i.e. the number of indices  $i_j$  in the signature of  $R$ . If  $i_1 = i_2 = \dots = i_k$ , then  $R$  is called a *one-sorted* relation over  $\mathfrak{D}$ . Note that if  $\mathfrak{D}$  is a singleton, then all relations over  $\mathfrak{D}$  are one-sorted.

**Example 2.1.2.** Let  $\mathfrak{D} = \{D_1, D_2, D_3, D_4\}$ , where  $D_1 = \{a, b, c\}$ ,  $D_2 = \{0, 1\}$ ,  $D_3 = \{2, 3, 4, 5\}$  and  $D_4 = \{d\}$ . Then, we can see:

$$R_1 = \{(0, a, 0, d), (0, b, 1, d), (1, a, 0, d), (1, b, 1, d)\}$$

either as a multi-sorted relation over  $\mathfrak{D}$  with signature  $\sigma(R_1) = (2, 1, 2, 4)$ , or as a one-sorted relation over:

$$D_1 \cup D_2 \cup D_4 = \{a, b, c, 0, 1, d\}.$$

On the other hand,  $R_2 = \{(0, a), (b, 1)\}$  is necessarily a one-sorted relation over  $D_1 \cup D_2$ , since only values of the same sort can be permuted.  $\diamond$

Let  $\Gamma$  be a set of multi-sorted relation over  $\mathfrak{D}$  over  $\mathcal{D}$ . A *Multi-Sorted Constraint Satisfaction Problem* over  $\Gamma$ , denoted by  $\text{MCSP}(\Gamma)$  is defined as follows:

MCSP( $\Gamma$ )

**Instance:**  $\mathcal{I} = (V, \mathcal{D}, \delta, \mathcal{C})$ , where:

- $V$  is a set of variables,
- $\delta : V \mapsto I$  is the *domain function* and
- $\mathcal{C}$  is a set of *multi-sorted constraints*  $C = (R, \text{sc}(R))$  where:
  - $\text{sc}(R) = (v_{i_1}, \dots, v_{i_n})$  is an  $n$ -ary vector of variables from  $V$ , called the *scope* of  $R$  and
  - $R$  is a multi-sorted relation in  $\Gamma$ , with signature  $\sigma(R) = (\delta(v_{i_1}), \dots, \delta(v_{i_n}))$ .

**Question:** Is there an *assignment of values* to the variables, that is, a function

$$f : V \mapsto \bigcup_{\mathcal{D} \in \mathcal{D}} D,$$

called a *solution* of the MCSP instance, that respects the variables' domains, i.e.  $f(v) \in \mathcal{D}_{\delta(v)}$  for all  $v \in V$  and that *satisfies* all of the constraints, that is,  $f[\text{sc}(R)] \in R$ , for all  $C \in \mathcal{C}$ ?

Figure 2.2: Multi-Sorted CSP

It is evident we can immediately convert an MCSP( $\Gamma$ ) problem to a CSP( $\Gamma$ ), by considering all  $R \in \Gamma$  as one-sorted relations over  $\bigcup_{\mathcal{D} \in \mathcal{D}} D \in \mathcal{D}$  and taking the corresponding constraints for each instance. We will see later on though that this approach does not preserve the tractability or the intractability of the problem and is thus of little practical use.

Returning to (one-sorted) CSP's, there are many ways to express them across the literature. Here, we present several of them, along with the connections between them. For another such general overview, see M. Bordinsky's essay [25].



## 2.2 First-Order Logic

We begin with some basic concepts and notation, but we do assume some familiarity with them. Our main point of reference is Enderton's work [80].

For our purposes, a *language* is any, (possibly infinite) set, containing all the usual parenthetical and sentential connective symbols  $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ , with their standard semantic meaning and a (possibly infinite) set  $V$  of variables. The set of its *parameters* contains the usual quantifier symbols  $(\forall, \exists)$ , constant symbols, usually denoted as  $c_0, c_1, c_2, \dots$  and  $n$ -ary predicate and functional symbols,  $n \in \mathbb{N}$ . In all that follows, we freely include any symbols we need in our languages.

We call any finite sequence of symbols of a language an *expression* or a *word*. A *term*  $t$  is either a single variable or constant, or an expression of the form:

$$f(t_1, \dots, t_k),$$

where  $f$  is a  $k$ -ary functional symbol and  $t_1, \dots, t_n$  are terms. An *atomic formula* is an expression of the form:

$$P(t_1, \dots, t_n),$$

where  $P$  is an  $n$ -ary predicate symbol and  $t_1, \dots, t_n$  are terms. Finally, the (*well-formed*) *formulas* are quantified atomic formulas connected by the connective symbols. For example:

$$\phi(x, y) = \forall z(P(x, z) \rightarrow (Q(x, y, z) \vee P(c_0, z)))$$

is a formula on two *free* variables  $x, y$ , that is variables not in the scope of any quantifier. A formula with no free variables is called a *sentence* or *proposition*. We use the usual ordering between the logical connectives when we omit the parenthesis.

A *structure*  $\mathfrak{A}$  is a function, whose domain is the set of parameters, such that:

- $|\mathfrak{A}|$  is a non-empty set called the *universe* or *domain* of  $\mathfrak{A}$ , which is assigned to the quantifiers,
- each constant symbol  $c$  takes a value in the universe  $|\mathfrak{A}|$ , denoted by  $c^{\mathfrak{A}}$ ,

- each  $n$ -ary predicate symbol gives rise to an  $n$ -ary relation  $P^{\mathfrak{A}} \subseteq |\mathfrak{A}|^n$  and
- each  $k$ -ary functional symbol is assigned a function  $f^{\mathfrak{A}} : |\mathfrak{A}|^k \rightarrow |\mathfrak{A}|$ .

A structure  $\mathfrak{A}$  is called *relational*, if it consists only of a universe and relations. Unless explicitly stated otherwise, all structures from now on will be relational. Also, when there is no confusion, we will omit the superscript  $\mathfrak{A}$ , thus writing e.g.  $P$  to denote both the symbol of the language and the relation assigned to it by the structure.

Given a formula  $\phi(x_1, \dots, x_n)$ , a structure  $\mathfrak{A}$  and a function  $s : V \rightarrow |\mathfrak{A}|$ , called an *assignment of values* (to the variables),  $\phi[s]$  denotes the assignment of value  $s(x_i)$  to each free variable  $x_i$  of  $\phi$ ,  $i = 1, \dots, n$ . We write:

$$\models_{\mathfrak{A}} \phi[s]$$

when  $\phi(s(x_1), \dots, s(x_n)) = 1$ , that is,  $\phi$  is true in  $\mathfrak{A}$  under assignment  $s$ , with the standard Tarski's definition of truth. We write

$$\models_{\mathfrak{A}} \phi$$

when  $\phi$  is true under any assignment in  $\mathfrak{A}$ . In such a case, we say that  $\mathfrak{A}$  is a *model* of  $\phi$ . Furthermore, we write

$$\models \phi$$

when  $\phi$  is *valid*, i.e. true in every structure and under any assignment of values.

Given some initial structures, we can construct new ones by combining them via the usual set-theoretic operations. Specifically, assume  $\mathfrak{A}$  and  $\mathfrak{B}$  are structures over a common language. Then:

- the intersection  $\mathfrak{A} \cap \mathfrak{B}$  is the structure whose universe is  $|\mathfrak{A} \cap \mathfrak{B}| = |\mathfrak{A}| \cap |\mathfrak{B}|$ , and where, for each predicate  $P$ ,  $P^{\mathfrak{A} \cap \mathfrak{B}} = P^{\mathfrak{A}} \cap P^{\mathfrak{B}}$ ,
- the union  $\mathfrak{A} \cup \mathfrak{B}$  is the structure whose universe is  $|\mathfrak{A} \cup \mathfrak{B}| = |\mathfrak{A}| \cup |\mathfrak{B}|$ , and where, for each predicate  $P$ ,  $P^{\mathfrak{A} \cup \mathfrak{B}} = P^{\mathfrak{A}} \cup P^{\mathfrak{B}}$  and
- the disjoint union  $\mathfrak{A} \uplus \mathfrak{B}$  is the structure whose universe is  $|\mathfrak{A} \uplus \mathfrak{B}| = |\mathfrak{A}| \uplus |\mathfrak{B}|$ , and where, for each predicate  $P$ ,  $P^{\mathfrak{A} \uplus \mathfrak{B}} = P^{\mathfrak{A}} \uplus P^{\mathfrak{B}}$ .

We can define analogously the intersection, union and disjoint union structures of any countable number of structures.

A set of formulas  $\Gamma$  is *satisfiable* if there is a structure  $\mathfrak{A}$  and an assignment of values  $s : V \rightarrow \mathfrak{A}$  such that all members of  $\Gamma$  are simultaneously true under  $s$  in  $\mathfrak{A}$ . Finally

$$\Gamma \models \phi$$

if whenever a  $\Gamma$  is satisfied by an  $s : V \rightarrow \mathfrak{A}$  in a structure  $\mathfrak{A}$ , then so is  $\phi$ . When  $\Gamma$  is a singleton  $\{\psi\}$ , we write  $\psi \models \phi$  instead of  $\{\psi\} \models \phi$  and write  $\phi \equiv \psi$  whenever both  $\phi \models \psi$  and  $\psi \models \phi$  hold.

**$\tau$ -formulas** A set  $\tau$  of relational symbols of finite arity is called a *relational signature*. We say that a structure  $\mathfrak{A}$  has a *finite relational signature* if it can be applied to a language with predicates of finite arities. A first order formula is called a  $\tau$ -formula if all its relational symbols are in  $\tau$ . A  $\tau$ -formula  $\phi(x_1, \dots, x_n)$  is called *primitive positive* if its of the form:

$$\exists x_{n+1}, \dots, \exists x_m (\psi_1 \wedge \dots \wedge \psi_l),$$

where  $\psi_j = P(y_{j_1}, \dots, y_{j_{k_j}})$ ,  $P \in \tau$ ,  $y_{j_i} \in \{x_1, \dots, x_m\}$ ,  $i = 1 \dots k_j$ ,  $j = 1, \dots, l$ . The conjuncts  $\psi_j$  are called the *constraints* of  $\phi$ . Don't be confused by the indices of the quantified variables. Under the notation we use,  $\phi(x_1, \dots, x_n)$  means that  $x_1, \dots, x_n$  are free in  $\phi$ .

Accordingly to CSP's over sets of relations  $\mathcal{R}$ , we can define a CSP over a structure  $\mathfrak{A}$  with a finite relational signature  $\tau$  as follows:

<p><u>CSP(<math>\mathfrak{A}, \tau</math>)</u>  <b>Instance:</b> Primitive positive <math>\tau</math>-sentence <math>\phi</math>.  <b>Question:</b> Is <math>\phi</math> true?</p>
--

Figure 2.3: Constraint Satisfaction Problem over a  $\tau$ -structure  $\mathfrak{A}$

By taking  $\mathcal{R}$  to contain exactly the predicates of  $\mathfrak{A}$  and dropping the existential quantifiers, we immediately obtain the CSP definition of Figure 2.1. In a way, the first definition of CSP's does not give any notice to the syntactic properties of the language, but simply assumes that we have some

way of writing down all the necessary information. The FO approach is certainly more systematic, but can become tiresome in practice.

Going back to Ex. [2.1.1](#), we use the terminology of First Order Logic to express the property of a graph admitting a proper coloring with  $K$  colors.

**Example 2.2.1.** *Assume first that our language contains a variable  $x_e$ , for each  $e = \{u, v\} \in E$  and a binary predicate  $P$ . Assume also a structure  $\mathfrak{A}$ , with universe  $|\mathfrak{A}| = \{1, \dots, K\}$ , such that  $P^{\mathfrak{A}} = \{(a, b) \in K^2 \mid a \neq b\} \neq \emptyset$ . Now, a proper coloring exists if and only if*

$$\models_{\mathfrak{A}} \exists y_1 \exists y_2 \cdots \exists y_m \left( \bigwedge_{\substack{e, e' \in E: \\ e \cap e' \neq \emptyset}} P(x_e, x_{e'}) \right),$$

where  $y_1, \dots, y_m$  is some arbitrary ordering of the variables  $x_e$ ,  $e \in E$ .  $\diamond$

**$\tau$ -theories** We can also define CSP's based on sets of sentences, rather than structures. A first-order *theory*  $T$  is a set of first order sentences. It is a  $\tau$ -theory if all sentences are over the same signature  $\tau$ . A theory  $T$  is satisfiable if there is a structure  $\mathfrak{A}$  such that every sentence in  $T$  is true in  $\mathfrak{A}$ . For a  $\tau$ -theory  $T$ , we can define  $\text{CSP}(T)$  as follows:

CSP( $T$ )

**Instance:** Primitive positive  $\tau$ -sentence  $\phi$ .

**Question:** Is  $T \cup \phi$  satisfiable?

Figure 2.4: Constraint Satisfaction Problem over a  $\tau$ -theory  $T$

Note that for  $\text{CSP}(T)$  to be interesting,  $T$  must be satisfiable. In terms of the expressibility, defining CSP's using structures and theories is equivalent. We begin by showing that a CSP defined on a theory  $T$  can also be expressed over a suitable structure  $\mathfrak{A}$ .

**Proposition 2.2.1.** *Let  $T$  be a satisfiable  $\tau$ -theory. Then, there exists a  $\tau$ -structure  $\mathfrak{A}$  such that  $\text{CSP}(\mathfrak{A}, \tau)$  and  $\text{CSP}(T)$  are the same problem.*

*Proof.* Let  $\mathfrak{T}$  be the set of models for  $T$  and define the union structure:

$$\mathfrak{A} = \bigcup_{\mathfrak{B} \in \mathfrak{T}} \mathfrak{B}.$$

Observe that  $T$  is satisfied in  $\mathfrak{A}$ . Indeed, any first-order primitive positive formula  $\psi \in T$  is satisfied by all  $\mathfrak{B} \in \mathfrak{T}$  and thus by  $\mathfrak{A}$  too. We now show that  $\text{CSP}(\mathfrak{A}, \tau)$  and  $\text{CSP}(T)$  are the same problem.

Let  $\phi$  be a first-order primitive positive  $\tau$ -sentence. If  $\phi \in T$ ,  $T \cup \phi$  is satisfiable and  $\phi$  is true in  $\mathfrak{A}$ . Thus, we can assume that  $\phi \notin T$ . Now let  $\phi$  be true in  $\mathfrak{A}$ . Since  $\mathfrak{A}$  is a model of  $T$ , it follows that  $T \cup \{\phi\}$  is satisfiable. Finally, if  $T \cup \{\phi\}$  is satisfiable, then there is some  $\mathfrak{B} \in \mathfrak{T}$  that satisfies it. It follows that  $\mathfrak{A} \supseteq \mathfrak{B}$  satisfies it too.  $\square$

We now turn our attention to the opposite result. Namely, given a CSP over a structure  $\mathfrak{A}$ , we show how we can find a theory  $T$  whose CSP expresses the same problem.

**Proposition 2.2.2.** *Let  $\mathfrak{A}$  be a  $\tau$ -structure. Then, there exists a  $\tau$ -theory  $T$  such that  $\text{CSP}(T)$  and  $\text{CSP}(\mathfrak{A}, \tau)$  are the same problem.*

*Proof.* Given  $\mathfrak{A}$ , we construct a first-order primitive positive  $\tau$ -sentence  $\phi$ , such that its only model is  $\mathfrak{A}$ . Then, it follows immediately that  $\text{CSP}(\mathfrak{A}, \tau)$  and  $\text{CSP}(\{\phi\})$  are the same problems.

Let  $|\mathfrak{A}| := \{a_1, \dots, a_m\}$  and assume that the predicates of  $\tau$ , in some arbitrary ordering, are  $P_1, \dots, P_s$ ,  $s \geq 1$ . Finally, let:

$$\begin{aligned} \phi(\mathfrak{A}) := \phi = & \exists x_1 \exists x_2 \exists \dots \exists x_m \left( \left( \forall y \bigvee_{i=1}^m (y = x_i) \right) \wedge \right. \\ & \wedge \left( \bigwedge_{\substack{j=1,2,\dots,s \\ (d_{j_1}, \dots, d_{j_{k_j}}) \in P_j^{\mathfrak{A}}}} P_j(x_{j_1}, \dots, x_{j_{k_j}}) \right) \wedge \\ & \left. \left( \bigwedge_{\substack{j=1,2,\dots,s \\ (d_{j_1}, \dots, d_{j_{k_j}}) \notin P_j^{\mathfrak{A}}} } \neg P_j(x_{j_1}, \dots, x_{j_{k_j}}) \right) \right). \end{aligned} \quad (2.1)$$

That  $\phi$  is such that its only model is  $\mathfrak{A}$  is straightforward.  $\square$

**The Boolean Satisfiability Problem (SAT)** It is known that for any Boolean  $n$ -ary relation  $R$ , there is a propositional formula  $\phi(x_1, \dots, x_n)$  such that the set of its satisfying assignments (or models)  $\text{Mod}(\phi) = R$ . Thus, if the universe  $|\mathfrak{A}|$  of a given structure  $\mathfrak{A}$  is Boolean, by dropping the existential quantifiers of a primitive positive  $\tau$ -sentence  $\phi$ , we obtain the Boolean *satisfiability* problem:

<p><u>SAT</u></p> <p><b>Instance:</b> Propositional formula <math>\phi</math>.</p> <p><b>Question:</b> Is <math>\phi</math> satisfiable?</p>
--

Figure 2.5: The Boolean Satisfiability Problem

SAT is quite possibly the most extensively studied problem in the field of Computational Complexity. We introduce here some terminology that we will be useful in the following chapters.

A propositional formula  $\phi(x_1, \dots, x_n)$  is in *Conjunctive Normal Form*, if it is a conjunction of terms, comprised from disjunctions of variables or negations of variables. Thus, we can write  $\phi$  as follows:

$$\phi(x_1, \dots, x_n) = \bigwedge_{j=1}^m C_j,$$

where the terms  $C_j$  are called *clauses*, and are of the form:

$$C_j = l_{j_1}, \dots, l_{j_{k_j}}, \quad j = 1, \dots, m,$$

where the  $l_{j_i}$ 's are called *literals* and:

$$l_{j_i} \in \{x_1, \dots, x_n, \neg x_1, \dots, \neg x_n\}, \quad i = 1, \dots, k_j, \quad j = 1, \dots, m.$$

For each variable  $x_i$ ,  $x_i$  is identified with its *positive* literal and  $\neg x_i$  is its *negative* literal,  $i = 1, \dots, n$ . We denote by  $\text{vbl}(C_j)$  the variables of clause  $C_j$ . Finally, the set of models of a formula  $\phi$  is denoted by  $\text{Mod}(\phi)$ .

If  $k_j = k$ ,  $j = 1, \dots, m$ , we say that  $\phi$  is a  $k$ -CNF formula. It is known that every propositional formula  $\phi$  has an equivalent 3-CNF one.

A CNF formula  $\phi$  is *Horn* (resp. *dual-Horn*) if each clause has at most one position (resp. negative) literal and 2 – SAT if each clause contains at

most two literals. Allowing formulas with clauses of a *generalized form*, we say that  $\phi$  is affine if each clause is of the form:

$$C_j = (l_{j_1} \oplus \cdots \oplus l_{j_{k_j}}),$$

where  $\oplus$  denotes the binary addition. Note that for example, the clause  $(x \oplus y \oplus z)$  is equivalent to:

$$(\neg x \vee \neg y \vee \neg z) \wedge (\neg x \vee \neg y \vee z) \wedge (\neg x \vee y \vee \neg z) \wedge (x \vee \neg y \vee \neg z),$$

but in practice, it is much more convenient to use the generalized clauses.

### 2.2.1 Transitive Closure Logic

It is well known that first-order logic has rather limited expressive power on finite structures. In particular, there is no formula of first-order logic that expresses *connectivity* on finite graphs (see Fagin [89] and Aho and Ullman [4]); this means that there is no formula  $\psi$  of first-order logic such that a finite graph  $\mathbf{G}$  satisfies  $\psi$  if and only if  $\mathbf{G}$  is connected. Intuitively one can see that to express connectivity, one would need to express a sequence  $z_1, \dots, z_l$ , such that  $(x, z_1), (z_l, y)$  and  $(z_i, z_{i+1}) \in R$ ,  $i = 1, \dots, l - 1$ . The problem here is that the number  $l$  can be arbitrarily large and we have no way of knowing it. Given that our formulas need to be finite in length, and thus something like:

$$\bigvee_{l \in \mathbb{N}} \left( R(x, z_1) \wedge R(x, z_2) \wedge \cdots \wedge R(z_{l-1}, z_l) \wedge R(z_l, y) \right)$$

is not allowed, it seems rather plausible that there is no way to express transitive closure in FO Logic. Proving it far exceeds the scope of this exposition. Moreover, the same holds true for other properties of finite graphs of algorithmic significance, such as *acyclicity* and *2-colorability*; for details, see, e.g., [161]. Intuitively, the reason for these limitations of first-order logic is that first-order logic on finite structures lacks a recursion mechanism.

Least Fixed-Point Logic (LFP) augments first-order logic with a recursion mechanism in the form of least fixed-points of *positive* first-order formulas. More formally, one considers first-order formulas of the form  $\varphi(x_1, \dots, x_n, R)$ , where  $\varphi(x_1, \dots, x_n, R)$  is a first-order formula over a relational schema with an extra  $n$ -ary relation symbol  $R$  such that every occurrence of  $R$  is within

an even number of negation symbols. Every such formula has a *least fixed-point*, that is, for every relational structure  $\mathcal{R}$ , there is a smallest relation  $R^*$  such that  $R^* = \{(a_1, \dots, a_n) \in R^n : \mathcal{R} \models \varphi(a_1, \dots, a_n, R^*)\}$ . We use the notation  $\varphi^\infty(x, y)$  to denote a new formula that expresses the least fixed-point of  $\varphi(x_1, \dots, x_n, S)$ .

A binary relation  $R \subseteq \mathcal{D}^2$  is transitive if, for all  $x, y, z \in \mathcal{D}$ ,  $(x, y), (y, z) \in R$  implies that  $(x, z) \in R$ . For a binary relation  $R \subseteq \mathcal{D}^2$ , its *transitive closure* is defined as the set:

$$R^{cl} := \bigcap \{Q \subseteq \mathcal{D}^2 \mid R \subseteq Q \text{ and } Q \text{ is transitive}\},$$

that is  $R^{cl}$  is the smallest transitive relation that contains  $R$ .

For example, if  $\varphi(x_1, x_2, R)$  is the formula

$$E(x_1, x_2) \vee \exists z (E(x_1, z) \wedge R(z, x_2)),$$

then, for every graph  $\mathbf{G} = (V, E)$ , the least fixed-point  $\varphi^\infty(x_1, x_2)$  of this formula defines the *transitive closure* of the edge relation  $E$ . Consequently, the expression  $\forall x_1 \forall x_2 \varphi^\infty(x_1, x_2)$  is a formula of least fixed-point logic LFP that expresses *connectivity*. For a different example, let  $\psi(x, S)$  be the formula  $\forall y (E(y, x) \rightarrow T(y))$ , where  $T$  is a unary relation symbol. It can be verified that, for every finite graph  $\mathbf{G} = (V, E)$ , the least fixed-point  $\psi^\infty(x)$  defines the set of all nodes  $v$  in  $V$  such that no path containing  $v$  leads to a cycle. Consequently, the expression  $\forall x \psi^\infty(x)$  is a formula of least fixed-point logic LFP that expresses *acyclicity* on finite graphs.

Transitive Closure Logic (TCL) is the fragment of LFP that allows for the formation of the transitive closure of first-order definable relations. Thus, if  $\theta(x_1, \dots, x_k, x_{k+1}, \dots, x_{2k})$  is a first-order formula, then we can form in TCL the least fixed point of the formula:

$$\theta(x_1, \dots, x_k, x_{k+1}, \dots, x_{2k}) \vee \exists z_1 \cdots \exists z_k (\theta(x_1, \dots, x_k, z_1, \dots, z_k) \wedge S(z_1, \dots, z_k, x_{k+1}, \dots, x_{2k})).$$

As regards their expressive power, it is known that  $\text{FO} \subset \text{TCL} \subset \text{LFP}$  on the class of all finite graphs. In other words, FO is strictly less expressive than TLC, while TLC is strictly less expressive than LFP on the class of all finite graphs. As regards connections to computational complexity, it is known FO is properly contained in LOGSPACE, TLC is properly contained



in NLOGSPACE, and LFP is properly contained in PTIME on the class of all finite graphs.

The situation, however, changes if *ordered* finite graphs are considered, that is, finite structures of the form  $\mathbf{G} = (V, E, \leq)$ , where  $E$  is a binary relation on  $V$  and  $\leq$  is a total order on  $V$  that can be used in LFP and in TCL formulas. In this setting, it is known that  $\text{TLC} = \text{NLOGSPACE}$  and that  $\text{LFP} = \text{PTIME}$  (the latter result is known as the Immerman-Vardi Theorem); thus, separating TLC from LFP on the class of all ordered graphs is equivalent to showing that NLOGSPACE is properly contained in PTIME, which is an outstanding open problem in computational complexity. Furthermore, similar results hold for the class of all finite structures and the class of all ordered finite structure over a relational schema containing at least one relation symbol of arity at least 2. These results have been established in the context of *descriptive complexity theory*, which studies the connections between computational complexity and expressibility in logics on finite structures. We refer the reader to the monographs [129, 161] for detailed information.

## 2.3 Homomorphisms

Another way to define CSP's is via the notion of *homomorphisms*. This framework has been used quite productively, and has led to various results in the field. Assume we have two structures  $\mathfrak{A}$  and  $\mathfrak{B}$ , with the same signature  $\tau$  (we call them also  $\tau$ -structures). A *homomorphism* from  $\mathfrak{A}$  to  $\mathfrak{B}$  is a function  $h : |\mathfrak{A}| \rightarrow |\mathfrak{B}|$  such that, for each  $m$ -ary relation  $P \in \tau$ , if  $(a_1, \dots, a_m) \in P^{\mathfrak{A}}$ , then  $(h(a_1), \dots, h(a_m)) \in P^{\mathfrak{B}}$ .

We can define a CSP problem over a structure  $\mathfrak{A}$  with finite signature  $\tau$ , using homomorphisms, as follows:

H CSP( $\mathfrak{A}, \tau$ )

**Instance:** Structure  $\mathfrak{B}$  with finite signature  $\tau$ .

**Question:** Is there a homomorphism  $h : |\mathfrak{B}| \rightarrow |\mathfrak{A}|$ ?

Figure 2.6: CSP over a  $\tau$ -structure  $\mathfrak{A}$  via homomorphisms

Quite often, we consider  $\text{HCSP}(\mathfrak{A}, \tau)$  as the *class* of  $\tau$ -structures that homomorphically map to  $\mathfrak{A}$ . To see the correspondence between Definitions [2.1](#), [2.2](#) and [2.3](#), first assume we have a finite  $\tau$ -structure  $\mathfrak{A}$ , whose universe  $|\mathfrak{A}| = \{d_1, \dots, d_m\}$  and let  $\tau = P_1, P_2, \dots, P_s$ , such that  $\alpha(P_j) = k_j$ ,  $i = 1, \dots, s$ . Let  $x_i$  be a variable corresponding to  $d_i$ ,  $i = 1, \dots, m$  and define the *canonical conjunctive query* of  $\mathfrak{A}$  as:

$$Q(\mathfrak{A}) = \exists x_1 \exists x_2 \cdots \exists x_m \bigwedge_{\substack{j=1,2,\dots,s \\ (d_{j_1}, \dots, d_{j_{k_j}}) \in P_j^{\mathfrak{A}}}} P_j(x_{j_1}, \dots, x_{j_{k_j}}). \quad (2.2)$$

Thus, the canonical conjunctive query over  $\mathfrak{A}$ , is the existential quantification over all predicates of  $\tau$  and over all vectors of the relations of  $\mathfrak{A}$ . It is not difficult now to prove the following result, that shows that any  $\text{HCSP}(\mathfrak{A}, \tau)$ , can be expressed also as a  $\text{CSP}(\mathfrak{A}, \tau)$ , and thus as a  $\text{CSP}(\mathcal{R})$  too.

**Proposition 2.3.1.** *Let  $\mathfrak{A}$  be a structure with finite relational signature  $\tau$  and  $\mathfrak{B}$  a finite  $\tau$ -structure. Then, there exists a homomorphism  $h : |\mathfrak{B}| \rightarrow |\mathfrak{A}|$  if and only if  $Q(\mathfrak{B})$  is true in  $\mathfrak{A}$ .*

*Proof.* For the forward direction, let  $x_i = d_i$ ,  $i = 1, \dots, m$  and consider an arbitrary conjunct  $P_j(x_{j_1}, \dots, x_{j_{k_j}})$  of  $\mathfrak{B}$ ,  $j \in \{1, \dots, s\}$ . Since this conjunct appears in  $Q(\mathfrak{B})$ , it must be the case that  $(d_{j_1}, \dots, d_{j_{k_j}}) \in P_j^{\mathfrak{B}}$ . Thus  $P_j(x_{j_1}, \dots, x_{j_{k_j}})$ , and by extension,  $Q(\mathfrak{B})$ , are true in  $\mathfrak{B}$ . Now, since  $h$  is a homomorphism,  $(h(d_{j_1}), \dots, h(d_{j_{k_j}})) \in P_j^{\mathfrak{A}}$ , thus  $Q(\mathfrak{B})$  is true in  $\mathfrak{A}$  too.

For the converse assume that  $Q(\mathfrak{B})$  is true in  $\mathfrak{A}$ . Then, there are values  $a_1, \dots, a_m \in |\mathfrak{A}|$ , such that, for  $x_i = a_i$ ,  $i = 1, \dots, m$ ,  $Q(\mathfrak{B})$  is true in  $\mathfrak{A}$ . Define  $h : |\mathfrak{B}| \rightarrow |\mathfrak{A}|$ , where  $h(d_i) = a_i$ ,  $i = 1, \dots, m$ . Let  $(d_{j_1}, \dots, d_{j_{k_j}}) \in P_j^{\mathfrak{B}}$ . By the definition of  $q(\mathfrak{B})$ , for some  $j \in \{1, \dots, s\}$ , we have that  $P = P_j$  and that  $P_j(x_{j_1}, \dots, x_{j_{k_j}})$  is true for  $x_{j_i} = d_{j_i}$ ,  $i = 1, \dots, k_j$ . By the definition of  $h$ ,  $P_j(x_{j_1}, \dots, x_{j_{k_j}})$  is true in  $\mathfrak{A}$  for  $x_{j_i} = a_i$ ,  $i = 1, \dots, k_j$ , thus  $(a_{j_1}, \dots, a_{j_{k_j}}) \in P_j^{\mathfrak{A}}$ .  $\square$

## 2.4 Second-Order Logic

Second-Order Logic (SO) is a richer and more expressive extension of FO. The difference with FO is that here, we can use quantifiers over predicates and

function symbols. Consider the setting of FO in Sec. [2.2](#). To avoid possible mix-ups, we say the the variables of  $V$  are *individual* and we introduce two additional kinds of variables:

- $n$ -place *predicate* variables  $X_1^n, X_2^n, \dots$  and
- $n$ -place *function* variables  $F_1^n, F_2^n, \dots$

We denote the set of all variables by  $\bar{V}$ . The definitions for terms and atomic formulas remain the same as before. The (well-formed) formulas are now extended to include formulas  $\psi$  of the form:

$$\forall X_i^n \phi, \forall F_i^n \phi, \exists X_i^n \phi, \exists F_i^n \phi,$$

where  $i \in \mathbb{N}$  and  $\phi$  is a (well-formed) formula.  $\psi$  is a sentence if no individual, predicate or function variable occurs free.

Now, given a structure  $\mathfrak{A}$ , an assignment of values to  $\psi$  is as in FO, where additionally, for any  $n$ -place predicate variable and any  $n$ -place function variable,  $s(X^n)$  is an  $n$ -ary relation over  $|\mathfrak{A}|$  and  $s(F^n)$  is an  $n$ -ary function on  $\mathfrak{A}$ , if  $t_1, \dots, t_n$  are terms,  $X$  and  $n$ -ary predicate variable and  $F$  an  $n$ -ary function variable:

$$\bar{s}(F(t_1, \dots, t_n)) = s(F)(\bar{s}(t_1), \dots, \bar{s}(t_n))$$

and

$$\models_{\mathfrak{A}} X(t_1, \dots, t_n) \text{ if and only if } (\bar{s}(t_1), \dots, \bar{s}(t_n)) \in s(X).$$

The truth of a formula  $\phi$  with quantified predicate or function variables is defined in the obvious way.

It is easy to observe that FO is indeed a part of SO. Nevertheless, we provide here a more explicit relation between the two logics.

**Theorem 2.4.1** (Skolem Normal Form). *For any first-order formula  $\phi$ , there is an equivalent second order formula  $\psi$  such that:*

- (i)  $\psi$  begins with a string of existential individual and function quantifiers,
- (ii) is followed by a string of universal individual quantifiers and
- (ii) ends with a quantifier-free formula.

The strings in (i) and (ii) can be empty. For a proof, see Enderton [\[80\]](#).

**Definition 2.4.1.** Let  $\psi$  be a second-order formula. We say that  $\psi$  is an SNP-sentence if it is of the form:

$$\exists \bar{X} \forall \mathbf{x} \phi(\bar{X}, \mathbf{x}, \mathfrak{A}, \mathfrak{B}),$$

where:

- $\bar{X}$  is a string of predicate variables,
- $\mathbf{x}$  is a string of individual variables and
- $\phi$  is a quantifier-free first-order formula with relational symbols from  $\mathfrak{A}$ .
- $\mathfrak{B}$  is a structure such that  $|\mathfrak{B}| = |\mathfrak{A}|$ .

The satisfiability problem for the second order formulas of Def. 2.4.1 amounts in checking if there is a structure  $\mathfrak{B}$ , with the same universe as  $\mathfrak{A}$ , such that, by assigning values to the predicate variables of  $\bar{X}$  from  $\mathfrak{B}$ ,  $\phi$  is true for all assignments  $\mathbf{a} \in |\mathfrak{A}| (= |\mathfrak{B}|)$  to the individual variables of  $\mathbf{x}$  (see [89, 149, 178]). Formally:

SNP( $\mathfrak{A}$ )

**Instance:** SNP-sentence  $\psi$ .

**Question:** Is there a structure  $\mathfrak{B}$  such that  $|\mathfrak{A}| = |\mathfrak{B}|$ , that makes  $\psi$  true?

Figure 2.7: SNP over structure  $\mathfrak{A}$

SNP is a broader class than CSP. Assume we have an instance of CSP( $\mathcal{R}$ ) (recall Fig. 2.1). For each constraint  $\mathcal{C}$ , put the constraint relation  $\mathcal{R}$  in  $\mathfrak{A}$ . For each variable  $x \in V$ , let  $X$  be a predicate variable of arity 1 and let  $\mathfrak{B}$  contain all unary predicates  $\{a\}$ , for each  $a \in |\mathfrak{A}| (= |\mathfrak{B}|)$ . Now, the SNP-sentence contains, for each constraint  $\mathcal{C} = (\mathcal{R}, \text{sc}(\mathcal{R}))$ , where  $\text{sc}(\mathcal{R}) := (x_1, \dots, x_n)$ , the sentence:

$$\exists X_1 \cdots \exists X_n \forall x_1 \cdots \forall x_n \mathcal{R}(x_1, \dots, x_n) \rightarrow \bigvee_{i=1}^n X_i.$$

We will talk more about it in Ch. 4, where we discuss the “Feder-Vardi conjecture”.

# Chapter 3

## The Probabilistic Method

In this Chapter, we present a general probabilistic framework for solving CSP's and discuss the main tools to which we focus. In Sec. [3.1](#), we discuss the Probabilistic Approach and introduce the *variable framework*. In Sec. [3.2](#), we consider *dependency graphs*, which constitute an efficient way to encode the dependencies between a set of probabilistic events and have been used extensively throughout the bibliography. We also present various dependency notions between events that are specifically tailored for the various theorems and algorithms that are used in this line of work. In Sec. [3.3](#) we discuss the principle tool we utilize for analyzing our algorithms, the “Lovász Local Lemma” and its variations. In Sec. [3.4](#) we provide some preliminaries concerning the acyclic edge coloring of graphs and in Sec. [3.5](#), we introduce some preliminary notions in Coding Theory, where some of our results are situated.

### 3.1 Probabilistic Framework

It is well known that if the set of relations  $\mathcal{R}$  is left unrestricted,  $\text{CSP}(\mathcal{R})$  is NP-complete. This means that although we can quickly verify if a given assignment of values to the variables is a solution or not, there is no efficient algorithm that can find such an assignment, unless the  $\text{P} \neq \text{NP}$  conjecture turns out to be false (something that seems highly unlikely). We will discuss aspects of the computational complexity of CSP's in Ch. [4](#) (see also Subsec. [1.3.3](#) in Ch. [1](#)).

As usual when one encounters NP-complete or harder problems, he or

she turns to other techniques, like searching for not optimal (approximate) solutions by relaxing the constraints or finding assignments that only satisfy a subset of the constraints. A very popular approach is to use randomness, allowing a chance for obtaining a wrong solution or for the algorithm to run for too long. Here, we consider the last technique and specifically, we employ what is known as the “Probabilistic Method” [8, 200]. Using randomness in a CSP amounts to producing random assignments of values to the variables. In the satisfiability setting of Fig. 2.2, we could for example flip a fair coin independently for each variable of a given propositional formula. Simply doing this directly gives us no advantage though. Searching in a random way for an assignment that might not even exist is of course no better than searching for it deterministically, by checking every possible assignment one by one (except if we are lucky!).

What we do instead, is to search for conditions that guarantee the existence of a solution. For example, consider a  $k$ -CNF formula:

$$\phi(x_1, \dots, x_l) = \bigwedge_{j=1}^m C_j,$$

where all the scopes  $\text{vbl}(C_j)$  of the constraints are pairwise distinct. We can easily argue that such a formula is necessarily satisfiable. Indeed, take any of the  $2^k - 1$  assignments that satisfy each clause, and combine them to produce a satisfying assignment for the whole formula. We show that straightforward result by an informal probabilistic fashion, to become somewhat familiar with this approach.

Assume that, in order to produce an assignment for  $\phi$ , we flip a fair coin, once for each variable  $x_i$  independently. Thus the probability for each  $x_i$  to be 0 (which is equal to that of being 1), is  $1/2$ . To compute the probability of an assignment to be satisfying, we argue as follows. Consider a clause  $C_j$ . Since there is only one assignment to the variables in  $\text{vbl}(C_j)$  that does not satisfy it, it holds that the probability of  $C_j$  to be satisfied is:

$$1 - \frac{1}{2^k} = \frac{2^k - 1}{2^k}.$$

Since we have  $m$  clauses with no common variables, it follows that the probability of  $\phi$  to be satisfied is:

$$\left(\frac{2^k - 1}{2^k}\right)^m,$$

which is greater than 0, since  $k \geq 1$ . Now, since  $\phi$  is satisfied with positive probability, a satisfying assignment must exist, thus our proof is complete.

The question of how quickly such an assignment can be found, will concern us in Ch. 6. For now, let us note that this is in fact the principle of the “Probabilistic Method”. When trying to prove the existence of an object with some desirable properties, build a probability space and show that a randomly chosen element in this space has the desired property with positive probability.

The criterion we used above to guarantee the satisfiability of  $\phi$  was of course not very useful. Firstly, most interesting formulas do not contain independent clauses (although, see the *separable* formulas of Ch. 5). Secondly, in this case, the probabilistic approach had no essential advantage than a purely analytical one would have. To see some more interesting examples, we begin with some formalism, that will enable us to deal with CSP’s in a probabilistic way.

Let  $\mathcal{D}$  be a finite domain. We define a probability space over  $\mathcal{D}$  in the standard way, using a *probability mass* function  $p : \mathcal{D} \rightarrow [0, 1]$ , such that:

$$\sum_{d \in \mathcal{D}} p(d) = 1$$

and where, for all  $A \subseteq \mathcal{D}$ :

$$p[A] = \sum_{d \in A} p(d).$$

We work with the *product probability space*  $(\mathcal{D}^l, p^l)$ , where  $\mathcal{D}^l$  is the *Cartesian product*:

$$\mathcal{D}^l := \underbrace{\mathcal{D} \times \cdots \times \mathcal{D}}_{l\text{-times}} := \{(d_1, \dots, d_l) \mid d_1, \dots, d_l \in \mathcal{D}\},$$

$p^l$  is the product probability measure:

$$p^l((d_1, \dots, d_l)) := \prod_{i=1}^l p(d_i)$$

and  $l \in \mathbb{N}$  is a positive constant. To simplify things, the reader can safely assume that  $p$ , and by extension  $p^l$ , is the *uniform* probability measure, where:

$$p[A] := \frac{|A|}{|\mathcal{D}|},$$

for all  $A \subseteq \mathcal{D}$ . However, what we do here works for any probability mass function  $p$ .

A subset  $E \subseteq \mathcal{D}^l$  is an *event*, whose probability is denoted by  $\Pr[E]$ . Its *complementary event*  $\mathcal{D}^l \setminus E$  is denoted by  $\overline{E}$  and it holds that:

$$\Pr[\overline{E}] = 1 - \Pr[E].$$

Given two events  $E$  and  $F$ , the *conditional probability* of  $E$  given  $F$ , is:

$$\Pr[E \mid F] = \frac{\Pr[E \cap F]}{\Pr[F]}.$$

Two events  $E$  and  $F$  are *independent*, if:

$$\Pr[E \cap F] = \Pr[E] \cdot \Pr[F], \quad (3.1)$$

or, equivalently, if  $\Pr[E \mid F] = \Pr[E]$ . Finally,  $E$  and  $F$  are *strictly negatively correlated*, if:

$$\Pr[E \mid \overline{F}] > \Pr[E]. \quad (3.2)$$

In what follows, we assume we have  $m$  events  $E_1, \dots, E_m$ , where  $m \in \mathbb{N}$  is a positive constant. We use  $\mathcal{E}$  to denote the set  $\{E_1, \dots, E_m\}$ . We consider these events as undesirables, in the sense that we want to find conditions that guarantee the existence of a point in  $\mathcal{D}^l$  such that none of them occurs. That is, we want to find conditions that guarantee:

$$\Pr \left[ \bigcap_{j=1}^m \overline{E}_j \right] > 0.$$

We thus always assume that  $\Pr[E_j] < 1$ ,  $j = 1, \dots, m$ , lest there is no way to avoid all the events.

We begin with two easy such probabilistic criteria.

**Observation 3.1.1.** *Let  $E_1, \dots, E_m$ ,  $m \geq 1$ , be events on  $\mathcal{D}^l$ . If:*

$$\sum_{j=1}^m \Pr[E_j] < 1,$$

*then:*

$$\Pr \left[ \bigcap_{j=1}^m \overline{E}_j \right] > 0.$$



The proof is immediate by the well known *union-bound* inequality:

$$\Pr \left[ \bigcup_{j=1}^m E_j \right] \leq \sum_{j=1}^m \Pr[E_j]. \quad (3.3)$$

The benefit of the union-bound is that adding probabilities is far easier than computing the probability of a union of events, for example via the *inclusion-exclusion* principle, that can be found in any introductory book in discrete mathematics or probabilities (see e.g. [6]) and is attributed to de Moivre [66], which is often not feasible. Nevertheless, the criterion is too restrictive to the point of being impractical. For example, if we have two fair coins, and the undesirable events are for each of them to be tails, the union-bound cannot guarantee us that we can get two heads.

**The variable framework** To apply the probabilistic approach to CSP's, we work in a slightly more restricted framework, the so called *variable framework*, which was first used by Moser and Tardos [172]. Most of the theorems and tools we use work in the general framework as well. However, the variable framework is both easily compatible with our CSP framework and very well suited for designing probabilistic algorithms, which are our main contributions.

Let  $X_i, i = 1, \dots, l$  be mutually independent random variables defined on a common probability space and taking values in  $\mathcal{D}$ . An assignment of values to the random variables is an  $l$ -ary vector  $\mathbf{a} = (a_1, \dots, a_l)$ , where  $a_i \in \mathcal{D}, i = 1, \dots, l$ . Observe that such a vector is, by definition, an element of  $\mathcal{D}^l$ . We assume that the events  $E_1, \dots, E_m$  depend on these random variables. The *scope*  $\text{sc}(E_j)$  of an event  $E_j$  is the minimal subset of variables such that one can determine whether  $E_j$  occurs or not knowing only their values.

By considering the definition of CSP's in Fig. 2.1, it is easy to see that an assignment  $\mathbf{a}$  to the random variables such that none of the events occur, corresponds to finding a solution to the instance  $(V, \mathcal{D}, \mathcal{C})$  of  $\text{CSP}(\bar{\mathcal{E}})$ , where  $\bar{\mathcal{E}} := \{\bar{E}_1, \dots, \bar{E}_m\}$  and:

- $V = \{X_1, \dots, X_l\}$ ,
- $\mathcal{C} = \{(\bar{E}_1, \text{sc}(E_1)), \dots, (\bar{E}_m, \text{sc}(E_m))\}$ .

On the other hand, given any instance  $(V, \mathcal{D}, \mathcal{C})$  of a  $\text{CSP}(\mathcal{R})$ , where  $V = \{x_1, \dots, x_l\}$  and  $\mathcal{C} = \{(C_1, \text{sc}(C_1)), \dots, (C_m, \text{sc}(C_m))\}$ , we can use the variable

framework in a straightforward way. We have a random variable  $X_i$  for each variable  $x_i \in V$ , taking values in  $\mathcal{D}$ , and the event  $E_j$  is that the constraint  $C_j$  is not satisfied, where  $\text{sc}(E_j) = \{X_i \mid x_i \in \text{sc}(C_j)\}$ .

Note that in the example of the CNF-formula we discussed, we would define the event  $E_j$  to mean that the  $j$ -th clause of  $\phi$  is not satisfied. Due to the hypothesis that  $\text{vbl}(C_j)$  were pairwise distinct, all the events would have pairwise distinct scopes.

Finally, observe that our assumption that  $\Pr[E_j] < 1$ ,  $j = 1, \dots, m$ , corresponds to excluding constraints whose constraint relation is  $\emptyset$ . This is a natural requirement for the CSP to be non-trivial, since otherwise the instance is obviously unsatisfiable. Note, on the other hand, that each constraint being satisfiable does not imply satisfiability of the whole instance.

There is a variation of the variable framework that has been used by Kolipaka and Szegedy [150] to prove Shearer’s Lemma [196]. We discuss this lemma in Sec. 3.3 and provide an alternative proof in Ch. 6. In their framework, they “invert” the variable framework and use the notion of *indicator variables* to define random variables through the events and not the other way around.

Given the set of events  $\mathcal{E} = \{E_1, \dots, E_m\}$ , the indicator variables of the events are  $\omega_1, \dots, \omega_m$ , where:

$$\omega_j = \begin{cases} 1, & \text{if } E_j \text{ occurs,} \\ 0, & \text{else.} \end{cases}$$

The only drawback of this approach is that the (random) indicator variables are not independent, something that induces complexities in analyzing their behaviour. Nevertheless, this framework is clearly broader than the variable one.

## 3.2 Dependency graphs

Whether one works in the variable framework, or in the general one, the events  $E_1, \dots, E_m$  to be avoided usually have at least some limited dependency on each other. We begin with two notions of dependency: the usual one of Eq. (3.1) and the *lopsidedependency*, which, in the framework of dependency graphs, first appeared in a paper of Erdős and Spencer [86] and takes into account negative correlation of Eq. (3.2).

**Mutual independence** The events  $E_1, \dots, E_m$  are *mutually independent*, if for every subset  $J \subseteq \{1, \dots, m\}$ :

$$\Pr \left[ \bigcap_{j \in J} E_j \right] = \prod_{j \in J} \Pr[E_j]. \quad (3.4)$$

Also,  $E_j$  is mutually independent from all the events  $E_i$  such that  $i \in I$ , if for every subset  $J \subseteq I$ :

$$\Pr \left[ E_j \mid \bigcap_{i \in J} E_i \right] = \Pr[E_j]. \quad (3.5)$$

Before proceeding, we can immediately obtain another criterion that guarantees that all the events can be avoided, which is in fact what we used in our SAT example, where the formula had clauses which pairwise contained different variables.

**Observation 3.2.1.** *Let  $E_1, \dots, E_m$ ,  $m \geq 1$ , be mutually independent events on  $\mathcal{D}^l$ . If  $\Pr[E_j] < 1$ ,  $j = 1, \dots, m$ , then:*

$$\Pr \left[ \bigcap_{j=1}^m \bar{E}_j \right] > 0.$$

Again, we can easily prove this observation, since by (3.4), we have that:

$$\Pr \left[ \bigcap_{j=1}^m \bar{E}_j \right] = \prod_{j=1}^m \Pr[\bar{E}_j]$$

and since  $E_1, \dots, E_m$  are mutually independent if and only if  $\bar{E}_1, \dots, \bar{E}_m$  are. Again, most interesting examples do not have mutually independent events exclusively.

We now proceed with the definition of a dependency graph, for a set of events  $\mathcal{E}$ .

**Definition 3.2.1.** *Given the set of events  $\mathcal{E} = \{E_1, \dots, E_m\}$ , let  $G[\mathcal{E}]$  be a graph whose vertex set is  $\{1, \dots, m\}$ .  $G[\mathcal{E}]$  is a dependency graph for  $\mathcal{E}$ , if for all  $j \in \{1, \dots, m\}$ ,  $E_j$  is mutually independent from all  $E_i$  such that  $i \notin N_j$ .*

Thus, given access to  $G[\mathcal{E}]$ , we can find the set of dependent events of some  $E_j$ , by checking its neighborhood  $N_j$ . Notice that mutual independence is a *symmetric* relation, thus  $G[\mathcal{E}]$  is an undirected graph.

**Lopsidedependency** We turn now to a notion of dependency that produces a sparser graph than that of Definition 3.2.1. This turns out to be beneficial for the algorithms we design in the sequel.

**Definition 3.2.2** (Erdős and Spencer [86]). *Given the set of events  $\mathcal{E} = \{E_1, \dots, E_m\}$ , let  $G[\mathcal{E}]$  be a graph whose vertex set is  $\{1, \dots, m\}$ .  $G[\mathcal{E}]$  is a lopsidedependency graph for  $\mathcal{E}$ , if for all  $j \in \{1, \dots, m\}$ , and for all  $I \subseteq \{1, \dots, m\}$  such that  $I \cap (N_j \cup \{j\}) = \emptyset$ :*

$$\Pr \left[ E_j \mid \bigcap_{i \in I} \overline{E}_i \right] \leq \Pr[E_j].$$

Observe that if two events are (strictly) negatively correlated, in the sense of Eq. (3.2), they should be connected by an edge in the lopsidedependency graph. Since the notion of negative correlation is symmetric, the lopsidedependency graph is again undirected. Also, it is immediate by the Definitions 3.2.1 and 3.2.2, that if  $G[\mathcal{E}]$  is a dependency graph for  $\mathcal{E}$ , then it is also a lopsidedependency graph. Finally, assume that  $\mathcal{E}$  contains two events  $E_i$  and  $E_j$ , such that  $\Pr[E_j \mid \overline{E}_i] < \Pr[E_j]$ . Then any lopsidedependency graph  $G(\mathcal{E}) = (V, E)$  for  $\mathcal{E}$ , such that  $\{i, j\} \notin E$ , is not a dependency graph for  $\mathcal{E}$ . Thus, a lopsidedependency graph for a set of events  $\mathcal{E}$  can be sparser than their dependency graph.

**Resampling Oracles** Harvey and Vondrak [121] introduced a notion they named *resampling oracles*. Suppose that each element  $\mathbf{a} \in \Omega$  is distributed according to  $\mu^{\mathbf{a}}$ . Given a graph  $G[\mathcal{E}]$ , a resampling oracle is a routine  $r_j : \mathcal{D}^l \mapsto \mathcal{D}^l$ ,  $j = 1, \dots, m$ , such that:

- (i) if  $\mathbf{a}$  follows the distribution  $\mu^{\mathbf{a}}|_{E_j}$ , that is  $\mu^{\mathbf{a}}$  conditional on  $E_j$ , then  $r_j(\mathbf{a})$  follows  $\mu^{\mathbf{a}}$ , that is,  $r_j$  removes the conditioning on  $E_j$  and
- (ii) if  $\mathbf{a} \notin E_i$ , where  $E_i$  is an event that is not in the extended neighborhood of  $E_j$ , then  $r_j(\mathbf{a}) \notin E_i$ .

Intuitively, resampling oracles are procedures that remove the dependencies on occurring events without causing a non-neighboring event to occur, given that it did not occur before the resampling took place. We will see in the sequel that this goes to the heart of the LLL-like algorithms. For now, let us state, without proving, that Harvey and Vondrak [121] showed that for such procedures to exist, the graph  $G[\mathcal{E}]$  must be a *lopsided association graph*.

**Definition 3.2.3.** Let  $G[\mathcal{E}]$  be a graph on  $\mathcal{E}$ .  $G[\mathcal{E}]$  is a lopsided association graph, denoted by  $G^A[\mathcal{E}]$ , if for all  $j \in \{1, \dots, m\}$  and for all  $E \in \mathcal{F}_j$ :

$$\Pr[E_j \cap E] \geq \Pr[E_j] \cdot \Pr[E], \quad (3.6)$$

where  $\mathcal{F}_j$  contains all events  $E$  whose indicator variable is a monotone non-decreasing function of the indicator variables of all  $E_i$  such that  $i \notin N_j^+$ .

$G^A[\mathcal{E}]$  can be sparser than a dependency graph, but it is a lopsidedependency graph. The proof for both these facts can be found again in [121].

**A non-probabilistic framework** Achlioptas and Iliopoulos [2, 2] introduced a novel framework that, although it is not probabilistic, can be used for the same purposes as the frameworks we discuss in this chapter. We provide a very brief overview here for completeness.

Assume that  $D$  is a domain set, and let  $G^{AI}[D]$  be a directed graph on  $D$ . A *flaw*  $f$  is a subset of  $D$  and our aim is to find flawless objects, that is a point in  $D$  that does not belong in any flaw, by performing *random walks* on  $G$ . An edge  $\sigma \rightarrow \tau$  on  $G[D]$  is called a *state transformation*. Thus, our aim is to randomly follow the edges of  $G^{AI}[D]$  in order to reach a state  $\sigma$  that is not an element of any flaw  $f$ .

**Definition 3.2.4.** Let  $D$  be a finite domain. We define the multi-digraph  $G^{AI}[D]$  on  $D$  as follows. Firstly, for each  $\sigma \in D$ , let  $U(\sigma) = \{f \text{ is a flaw} \mid \sigma \in f\}$ . For each  $\sigma \in D$  and  $f \in U(\sigma)$ , let  $\{\sigma\} \neq A(f, \sigma) \subseteq D$  be the set of possible actions for addressing  $f$  on  $\sigma$ .

For each  $\sigma \in D$ , each flaw  $f \in U(\sigma)$  and each  $\tau \in A(f, \sigma)$ , let  $\sigma \xrightarrow{f} \tau$  be an edge of  $G^{AI}[D]$ .

The correspondence to the usual framework is straightforward.  $D$  corresponds to  $\mathcal{D}^l$ , flaws correspond to events and  $G^{AI}[D]$  corresponds to the (lopsi)dependency graph of the events. The advantage here is that  $D$  does not need to have any specific structure or symmetry like  $\mathcal{D}^l$  and the edges of  $G^{AI}[D]$  can be autonomously, according to each flaw.

We proceed now with some dependency notions, specifically tailored for the variable framework. All of them define lopsidedependency graphs for a set of events  $\mathcal{E}$  that are not necessarily minimal with respect to sparseness. However, they can be used easily to design fast algorithms for solving CSP's. Intuitively, the sparser the dependency graph, the less the dependencies a given algorithm has to take care of.

**Simple dependency** The easiest way to define a dependency graph in the variable framework, is by connecting events whose scopes share at least one variable.

**Definition 3.2.5.** Two events  $E_i, E_j, i, j \in \{1, \dots, m\}$ , are  $s$ -dependent if  $\text{sc}(E_i) \cap \text{sc}(E_j) \neq \emptyset$ .

Let now  $G^s(\mathcal{E})$  be the graph whose:

- vertex set is  $\{1, \dots, m\}$  and
- $E = \{\{i, j\} \mid E_i \text{ and } E_j \text{ are } s\text{-dependent}\}$ .

Since  $s$ -dependency is a symmetric notion and since each event depends on itself,  $G^s(\mathcal{E})$  is an undirected graph with loops. For technical reasons that will become apparent in the sequel, we can safely ignore the loops.

**Example 3.2.1.** Suppose we have four independent Bernoulli trials, each represented by a random variable  $X_i, i = 1, 2, 3, 4$ , with probability of success ( $X_i = 1$ ) equal to  $x \in [0, 1]$ .

Consider the set of events  $\mathcal{E} = \{E_1, E_2, E_3\}$ , where:

$$\begin{aligned} E_1 &= \{(X_1 = 0 \vee X_2 = 1) \wedge X_3 = 0\}, \\ E_2 &= \{X_2 = 1 \wedge X_4 = 0\} \text{ and} \\ E_3 &= \{X_3 = 0 \wedge X_4 = 1\}. \end{aligned}$$

It is trivial to observe that  $E_1, E_2$  and  $E_3$  are pairwise  $s$ -dependent and, consequently, that  $G^s[\mathcal{E}]$  is the 3-cycle (or 3-clique):

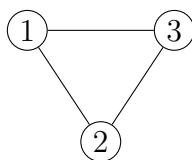


Figure 3.1:  $s$ -dependency graph of Def. 3.2.5

That the graph of Fig. 3.2.1 is a (lospi)dependency graph follows immediately by Def. 3.2.1 and 3.2.2.  $\diamond$

We now show that  $G^s[\mathcal{E}]$  is always a dependency graph and thus a lo-sidependency graph too.

**Lemma 3.2.1.** *Let  $\mathcal{E}$  be a set of events. Then,  $G^s[\mathcal{E}]$  is a dependency graph for  $\mathcal{E}$ .*

*Proof.* Let  $E_j$ ,  $j \in \{1, \dots, m\}$  be an arbitrary event from  $\mathcal{E}$ , and  $I \subseteq \{1, \dots, m\}$  such that  $I \cap (N_j \cup \{j\}) = \emptyset$ . It suffices to prove that:

$$\Pr \left[ E_j \mid \bigcap_{i \in I} \overline{E}_i \right] = \Pr[E_j] \quad (3.7)$$

To that end, assume  $E = \bigcap_{i \in I} \overline{E}_i$  and note that  $E$  is not necessarily in  $\mathcal{E}$ . In fact, if it is, then there is not assignment of values such that all the events are avoided.

To prove Eq. (3.7), we can equivalently show that:

$$\Pr[E_j \cap E] \geq \Pr[E_j] \cdot \Pr[E], \quad (3.8)$$

$$\Pr[E_j \cap E] \leq \Pr[E_j] \cdot \Pr[E]. \quad (3.9)$$

Assume  $\mathbf{a} = (a_1, \dots, a_l)$  and  $\mathbf{b} = (b_1, \dots, b_l)$  are two assignments of values obtained by independently sampling the random variables twice, once to get  $\mathbf{a}$  and once to get  $\mathbf{b}$ . Construct the assignments  $\mathbf{a}' = (a'_1, \dots, a'_l)$ ,  $\mathbf{b}' = (b'_1, \dots, b'_l)$  by swapping the values of the variables in  $\text{sc}(E)$ :

- $a'_i = b_i$ , for all  $i$  such that  $X_i \in \text{sc}(E)$ ,  $a'_i = a_i$  for the rest,
- $b'_i = a_i$ , for all  $i$  such that  $X_i \in \text{sc}(E)$  and  $b'_i = b_i$  for the rest.

Observe that  $\mathbf{a}'$  and  $\mathbf{b}'$  are both independent samplings of the variables, since  $\mathbf{a}$  and  $\mathbf{b}$  were, and we only swapped values in some positions.

Now, for Eq. (3.8), let  $Q$  be the event that  $E_j$  occurs under  $\mathbf{a}$  and  $E$  occurs under  $\mathbf{b}$  and let  $R$  be the event that both  $E_j$  and  $E$  occur under  $\mathbf{a}'$ . Since  $E_j$  and  $E$  have disjoint scopes and by the definition of  $\mathbf{a}'$ , it holds that  $Q$  implies  $R$ . Thus:

$$\Pr[E_j] \cdot \Pr[E] = \Pr[Q] \leq \Pr[R] = \Pr[E_j \cap E].$$

For Eq. (3.9), let  $S$  be the event that both  $E_j$  and  $E$  occur under  $\mathbf{a}$  and  $T$  the event that  $E_j$  occurs under  $\mathbf{a}'$  and  $E$  occurs under  $\mathbf{b}'$ . Again, since  $E_j$  and  $E$  have disjoint scopes and by the definitions of  $\mathbf{a}'$  and  $\mathbf{b}'$ ,  $S$  implies  $T$ . Thus:

$$\Pr[E_j \cap E] = \Pr[S] \leq \Pr[T] = \Pr[E_j] \cdot \Pr[E].$$

Thus, since  $\mathcal{E}$  was arbitrary, we conclude that  $s$ -dependency always defines a dependency graph and thus a lopsidedependency graph too.  $\square$

**Moser-Tardos lopsidedependency** The first variation of lopsidedependency in the variable framework was provided by Moser and Tardos [172]. Intuitively, two events are *MT-lopsidedependent* if by trying to avoid one of them, we end up with the other one occurring, although it did not before. This property expresses some kind of negative correlation between the events, although, as we shall see in the sequel, it can be refined to obtain sparser dependency graphs.

**Definition 3.2.6** (Moser and Tardos [172]). *Let  $E_i, E_j$  be events,  $i, j \in \{1, \dots, m\}$ . We say that  $E_i, E_j$  are MT-lopsidedependent if there exist two assignments  $\mathbf{a}, \mathbf{b}$ , that differ only on variables in  $\text{sc}(E_i) \cap \text{sc}(E_j)$ , such that:*

1.  $\mathbf{a}$  makes  $E_i$  occur and  $\mathbf{b}$  makes  $E_j$  occur and
2. either  $\bar{E}_i$  occurs under  $\mathbf{b}$  or  $\bar{E}_j$  occurs under  $\mathbf{a}$ .

Obviously, an event  $E_j$  is never lopsidedependent on itself, nor on any event whose scope shares no variables with  $\text{sc}(E_j)$ . Let now  $G^{MT}(\mathcal{E})$  be the graph whose:

- vertex set is  $\{1, \dots, m\}$  and
- $E = \{\{i, j\} \mid E_i \text{ and } E_j \text{ are MT-lopsidedependent}\}$ .

Since MT-lopsidedependency is a symmetric notion,  $G^{MT}(\mathcal{E})$  is again an undirected graph.

**Example 3.2.2.** *Consider the setting of Example [3.2.1], where the events are:*

$$\begin{aligned} E_1 &= \{(X_1 = 0 \vee X_2 = 1) \wedge X_3 = 0\}, \\ E_2 &= \{X_2 = 1 \wedge X_4 = 0\} \text{ and} \\ E_3 &= \{X_3 = 0 \wedge X_4 = 1\}. \end{aligned}$$

*We've seen that  $G^s[\mathcal{E}]$  was a 3-cycle. We now prove that the graph  $G^{MT}[\mathcal{E}]$  is sparser. Consider the assignments  $\mathbf{a} = (0, 0, 0, *)$  and  $\mathbf{b} = (0, 1, 0, *)$ , where '\*' means that the exact value in this coordinate can be arbitrary. Under  $\mathbf{a}$ ,  $E_1$  occurs and  $E_2$  does not. Also,  $\mathbf{b}$  differs from  $\mathbf{a}$  only in the value of  $X_2$ , which belongs in  $\text{sc}(E_1) \cap \text{sc}(E_2)$  and, under it,  $E_2$  occurs. By Def. [3.2.6],  $E_1$  and  $E_2$  are lopsidedependent. Analogously, under  $\mathbf{a}' = (*, 1, 0, 0)$ ,  $E_2$  occurs and  $E_3$  does not, whereas under  $\mathbf{b}' = (*, 1, 0, 1)$   $E_3$  occurs and  $E_2$  does not.*



Also, these two assignments differ only in  $\text{sc}(E_2) \cap \text{sc}(E_3)$ , thus, by Definition [3.2.6](#),  $E_2$  and  $E_3$  are lopsidedependent. Finally, under any assignment of the form  $(*, *, 1, *)$ , neither  $E_1$  nor  $E_3$  occur. Since  $\text{sc}(E_1) \cap \text{sc}(E_3) = \{X_3\}$ , it follows that there are no assignments that satisfy the conditions of Def. [3.2.6](#). Thus,  $E_1$  and  $E_3$  are not lopsidedependent. This means that  $G^{MT}[\mathcal{E}]$  is as follows:

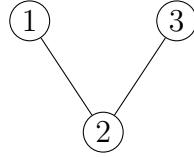


Figure 3.2: MT-lopsidedependency graph of Def. [3.2.6](#)

Easily:

$$\Pr[E_1] = \left( \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} \right) \cdot \frac{1}{2} = \frac{3}{8}$$

and

$$\Pr[E_1 \mid \bar{E}_3] = \frac{\Pr[E_1 \cap \bar{E}_3]}{\Pr[\bar{E}_3]} = \frac{\frac{3}{16}}{\frac{3}{4}} = \frac{1}{4} < \Pr[E_1].$$

This shows that the graph of Fig. [3.2.2](#) is a lopsidedependency graph, but not a dependency one.  $\diamond$

We end this paragraph by showing that for any set of events  $\mathcal{E}$ , the graph  $G^{MT}[\mathcal{E}]$  is a lopsidedependency graph.

**Lemma 3.2.2.** *Let  $\mathcal{E}$  be a set of events. Then,  $G^{MT}[\mathcal{E}]$  is a lopsidedependency graph for  $\mathcal{E}$ .*

*Proof.* We closely follow that proof of Lemma [3.2.1](#). Let  $E_j$ ,  $j \in \{1, \dots, m\}$  be an arbitrary event from  $\mathcal{E}$ ,  $I \subseteq \{1, \dots, m\}$  such that  $I \cap (N_j \cup \{j\}) = \emptyset$  and set  $E = \bigcap_{i \in I} \bar{E}_i$ . We show that:

$$\Pr[E_j \mid E] \leq \Pr[E_j],$$

or, equivalently, that:

$$\Pr[E_j \cap E] \leq \Pr[E_j] \cdot \Pr[E].$$

Again, let  $\mathbf{a} = (a_1, \dots, a_l)$  and  $\mathbf{b} = (b_1, \dots, b_l)$  be two independent random samplings of the variables and obtain  $\mathbf{a}' = (a'_1, \dots, a'_l)$  and  $\mathbf{b}' = (b'_1, \dots, b'_l)$  by swapping the values of the variables in  $\text{sc}(E) \setminus \text{sc}(E_j)$ . As before,  $\mathbf{a}'$  and  $\mathbf{b}'$  are both independent samplings of the variables.

Finally, let  $S$  be the event that both  $E_j$  and  $E$  occur under  $\mathbf{a}$  and  $T$  the event that  $E_j$  occurs under  $\mathbf{a}'$  and  $E$  occurs under  $\mathbf{b}'$ . Since  $a_i = a'_i$ , for all  $i$  such that  $X_i \in \text{sc}(E_j)$ ,  $E_j$  occurs under  $\mathbf{a}'$ . Assume now that  $E$  does not occur under  $\mathbf{b}'$  and let  $\mathbf{c}$  be the assignment of values such that:

$$c_i = \begin{cases} b'_i, & i : X_i \in \text{sc}(E) \\ a_i, & \text{else.} \end{cases}$$

Since  $c_i = b'_i$  for all  $i$  such that  $X_i \in \text{sc}(E)$ ,  $E$  occurs under  $\mathbf{c}$ . Also,  $c_i \neq a_i$  if and only if  $i$  such that  $X_i \in \text{sc}(E_j) \cap \text{sc}(E)$ . Thus,  $E_j$  and  $E$  are lopsidedependent. Contradiction.

It follows that  $S$  implies  $T$ , and thus:

$$\Pr[E_j \cap E] = \Pr[S] \leq \Pr[T] = \Pr[E_j] \cdot \Pr[E].$$

Thus, since  $\mathcal{E}$  was arbitrary, we conclude that MT-lopsidedependency always defines a lopsidedependency graph too.  $\square$

**Variable-directed lopsidedependency** In [145], we presented a non symmetric version of MT-lopsidedependency we named *Variable-dependent Directed Lopsidedependency (VDL)* (depending on the context, VDL may also stand for “Variable-dependent Directed Lopsidedependent” –an adjective rather than a noun).

**Definition 3.2.7** (Kirousis and Livieratos [145]). *Let  $E_i, E_j$  be events,  $i, j \in \{1, \dots, m\}$ . We say that  $E_j$  is VDL on  $E_i$  if:*

1. *there exists an assignment  $\mathbf{a}$  under which  $E_i$  and  $\overline{E_j}$  occur and*
2. *the values of the variables in  $\text{sc}(E_i)$ , can be changed so that  $E_j$  occurs.*

Intuitively,  $E_j$  is VDL on  $E_i$  if the effort to undo the undesirable event  $E_i$  results in  $E_j$  occurring, although it did not before. Notice that an event  $E_j$  can never be VDL on itself, nor on any event whose scope shares no variables with  $\text{sc}(E_j)$ .

Let now  $G^{VDL}(\mathcal{E})$  be the graph whose:

- vertex set is  $\{1, \dots, m\}$  and
- $E = \{(i, j) \mid E_j \text{ is VDL on } E_i\}$ .

Since VDL is not a symmetric relation,  $G^{VDL}(\mathcal{E})$  is a directed graph. We first show the connection between VDL and MT-lopsidependency.

**Lemma 3.2.3.** *Two events  $E_i, E_j$ ,  $i, j \in \{1, \dots, m\}$ , are MT-lopsidependent if and only if  $E_i$  is VDL on  $E_j$  or  $E_j$  is VDL on  $E_i$ .*

*Proof.* ( $\Rightarrow$ ) By Definition [3.2.6](#), there exist assignments  $\mathbf{a}, \mathbf{b}$  that differ only on variables in  $\text{sc}(E_i) \cap \text{sc}(E_j)$  such that  $E_i$  occurs under  $\mathbf{a}$ ,  $E_j$  occurs under  $\mathbf{b}$  and either  $\bar{E}_j$  occurs under  $\mathbf{a}$  or  $\bar{E}_i$  occurs under  $\mathbf{b}$ . It is immediate to see that if  $\bar{E}_j$  occurs under  $\mathbf{a}$ ,  $E_j$  is VDL on  $E_i$  and that, if  $\bar{E}_i$  occurs under  $\mathbf{b}$ ,  $E_i$  is VDL on  $E_j$ .

( $\Leftarrow$ ) Assume  $E_j$  is VDL on  $E_i$ . Then, there are two assignments  $\mathbf{a} = (a_1, \dots, a_i)$  and  $\mathbf{b} = (b_1, \dots, b_i)$  that differ only in  $\text{sc}(E_i)$ , such that  $E_i, \bar{E}_j$  occur under  $\mathbf{a}$  and  $E_j$  occurs under  $\mathbf{b}$ . If assignments  $\mathbf{a}, \mathbf{b}$  differed only in  $\text{sc}(E_i) \cap \text{sc}(E_j)$ , there would be nothing to prove.

Let the assignment  $\mathbf{b}' = (b'_1, \dots, b'_i)$  be such that:

- $b'_i = a_i$ , for all  $i \in \{1, \dots, n\}$  such that  $X_i \notin \text{sc}(E_i) \cap \text{sc}(E_j)$  and
- $b'_i = b_i$ , for all  $i \in \{1, \dots, n\}$  such that  $X_i \in \text{sc}(E_i) \cap \text{sc}(E_j)$ .

Since  $\mathbf{a}$  differs from  $\mathbf{b}$  only on variables in  $\text{sc}(E_i)$ , it follows that  $\mathbf{b}'$  differs from  $\mathbf{b}$  only on variables in  $\text{sc}(E_i) \setminus \text{sc}(E_j)$ . Now, since  $E_j$  holds for  $\mathbf{b}$  and does not depend on variables not in its scope,  $E_j$  holds under  $\mathbf{b}'$  also. Thus, the assignments  $\mathbf{a}$  and  $\mathbf{b}'$  fulfill the requirements of Definition [3.2.6](#). The analogous arguments go through if  $E_i$  is VDL on  $E_j$ .  $\square$

A direct consequence of Lemma [3.2.3](#) is that, for any set of events  $\mathcal{E}$ ,  $G^{MT}[\mathcal{E}]$  is the *undirected* graph that underlies  $G^{VDL}[\mathcal{E}]$ .

**Example 3.2.3.** *Consider again the setting of Ex. [3.2.2](#). We show that  $G^{VDL}[\mathcal{E}]$  is sparser than  $G^{MT}[\mathcal{E}]$ . By Lemma [3.2.3](#),  $E_1$  and  $E_3$  cannot be connected in  $G^{VDL}[\mathcal{E}]$  since they are not connected in  $G^{MT}[\mathcal{E}]$ . Also, by using the exact same assignments as in Ex. [3.2.2](#), we get that  $E_2$  is VDL on  $E_1$  and that  $E_2$  and  $E_3$  are VDL on each other.*

On the other hand,  $E_1$  is not VDL on  $E_2$ . To see this, consider the four assignments under which  $E_2$  occurs, namely:

$$(0, 1, 0, 0), (0, 1, 1, 0), (1, 1, 0, 0), (1, 1, 1, 0).$$

Under the first and third of the assignments above,  $E_1$  occurs too. Thus, only the second and fourth assignments satisfy the first condition of Definition [3.2.7](#). But, under both these assignments,  $X_3 = 1$ , which means that  $E_1$  does not occur. Since  $X_3 \notin \text{sc}(E_2)$ , there is no way to make  $E_1$  occurring by changing the values of the variables  $\in \text{sc}(E_2)$ , thus the second condition of Definition [3.2.7](#) does not hold.

This means that  $G^{VDL}[\mathcal{E}]$  is as follows:

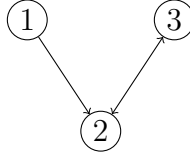


Figure 3.3: VDL-lopsidependency graph of Def. [3.2.7](#)

Since  $G^{MT}[\mathcal{E}]$  is not a dependency graph, neither is  $G^{VDL}[\mathcal{E}]$ . Finally, given the calculation in Ex. [\(3.2.2\)](#) and that:

$$\Pr[E_2 \mid \bar{E}_1] = \frac{\frac{1}{8}}{\frac{5}{8}} = \frac{1}{5} \leq \frac{1}{4} = \Pr[E_2],$$

it follows that  $G^{MT}[\mathcal{E}]$  is a lopsidependency graph.  $\diamond$

We end this paragraph by showing that  $G^{VDL}[\mathcal{E}]$  is always a lopsidependency graph.

**Lemma 3.2.4.** *Let  $\mathcal{E}$  be a set of events. Then,  $G^{VDL}[\mathcal{E}]$  is a lopsidependency graph for  $\mathcal{E}$ .*

*Proof.* Same as that of Lemma [3.2.2](#), by noting that  $\text{sc}(E_j) \cap \text{sc}(E) \subseteq \text{sc}(E_j)$ .  $\square$

**$d$ -dependency** We end this section by further refining the lopsidedependency notions we have discussed.

**Definition 3.2.8** (Kirousis et al. [147]). *Let  $E_i, E_j$  be events in  $\mathcal{E}$ . We say that  $E_j$  is  $d$ -dependent on  $E_i$  if:*

1. *there exists an assignment  $\mathbf{a}$  under which  $E_i$  and  $\bar{E}_j$  occur and*
2. *the values of the variables in  $\text{sc}(E_i)$ , can be changed so that  $E_j$  occurs and  $E_i$  ceases occurring.*

Intuitively,  $E_j$  is  $d$ -dependent on  $E_i$  if it is possible that some *successful* attempt to avoid the occurrence of  $E_i$  may end up with  $E_j$  occurring, although initially it did not. Again,  $E_j$  can never be  $d$ -dependent on itself, nor on any event whose scope shares no variables with  $\text{sc}(E_j)$ .

Let now  $G^d(\mathcal{E})$  be the graph whose:

- vertex set is  $\{1, \dots, m\}$  and
- $E = \{(i, j) \mid E_j \text{ is } d\text{-dependent on } E_i\}$ .

Since  $d$ -dependency is not a symmetric relation,  $G^d(\mathcal{E})$  is a directed graph. For the connection of  $d$ -dependency to VDL and MT-dependency, we have the following result.

**Lemma 3.2.5.** *Let  $E_i$  and  $E_j$  be events,  $i, j \in \{1, \dots, m\}$  and assume  $E_j$  is  $d$ -dependent on  $E_i$ . Then  $E_j$  is VDL on  $E_i$  and thus  $E_i$  and  $E_j$  are MT-lopsidedependent.*

*Proof.* Immediate by Definition [3.2.7] and Lemma [3.2.3]. □

In the following example, we show that the  $d$ -dependency graph can be strictly sparser than other dependency graphs that have been used in the literature.

**Example 3.2.4.** *Consider the setting of Ex. [3.2.3]. It is straightforward to observe that  $G^d[\mathcal{E}] = G^{\text{VDL}}[\mathcal{E}]$ . Thus,  $G^d[\mathcal{E}]$  is sparser than  $G^{\text{MT}}[\mathcal{E}]$ , it is not a dependency graph and it is a lopsidedependency one. To show that it can be sparser than the VDL graph in some cases, consider the following setting.*

*Suppose we have  $l \geq 3$  independent Bernoulli trials  $X_1, X_2, \dots, X_l$ , where  $X_i = 1$  denotes the event that the  $i$ -th such trial is successful,  $i = 1, \dots, l$ . Consider also the set  $\mathcal{E}$  containing  $n$  events:*

$$E_j = \{X_j = 1 \vee X_{j+1} = 1\},$$

where  $X_{l+1} = X_1$  and assume also that each of the Bernoulli trials succeeds with probability  $p \in [0, 1)$ . Thus:

$$\Pr[E_j] = p + (1 - p)p = 2p - p^2.$$

We begin by showing that for any two distinct  $E_i, E_j$ , neither one of them is  $d$ -dependent on the other. Without loss of generality, let  $i = 1$  and  $j = 2$ .

Since  $\text{sc}(E_1) \cup \text{sc}(E_2) = \{X_1, X_2, X_3\}$ , both  $E_1$  and  $E_2$  are affected only by the first three coordinates of an assignment of values. We will thus restrict the assignments to those coordinates.

Suppose  $E_1$  and  $\bar{E}_2$  occur under an assignment  $\mathbf{a}$ . Then,  $\mathbf{a} = (1, 0, 0)$  and there is no way to change the first two coordinates in order for  $\bar{E}_1$  and  $E_2$  to occur. Thus  $E_2$  is not  $d$ -dependent on  $E_1$ . Furthermore, for  $\bar{E}_1$  and  $E_2$  to occur under an assignment  $\mathbf{b}$ ,  $\mathbf{b} = (0, 0, 1)$  and there is no way to change the last two coordinates of  $\mathbf{b}$  in order for  $E_1$  and  $\bar{E}_2$  to occur. Thus  $E_1$  is not  $d$ -dependent on  $E_2$ .

We can analogously prove the same things for all pairs of  $E_j, E_{j+1}$ ,  $j = 1, \dots, n$ , where  $E_{n+1} := E_1$ . Furthermore, it is easy to see that for any  $i, j \in \{1, \dots, n\}$ :  $i < j$  and  $j \neq i + 1$ , neither  $E_j$  is  $d$ -dependent on  $E_i$  nor vice versa, since  $E_i, E_j$  have no common variables they depend on. Thus, the  $d$ -dependency graph of the events has no edges and it is trivial to observe that we can avoid all the events if and only if  $p < 1$ .

On the other hand, consider assignments  $\mathbf{a}' = (1, 0, 0)$  and  $\mathbf{b}' = (1, 1, 0)$ . Under  $\mathbf{a}'$ ,  $E_1, \bar{E}_2$  occur, under  $\mathbf{b}'$   $E_2$  occurs and the assignments differ only on  $X_2 \in \text{sc}(E_1) \cap \text{sc}(E_2)$ . By Definition 3.2.7,  $E_2$  is VDL on  $E_1$ , and thus,  $E_1$  and  $E_2$  are MT-lopsidependent.

Given the above, it is not difficult to see that  $G^{MT}[\mathcal{E}] = G^{VDL}[\mathcal{E}]$  is an  $n$ -cycle, whereas  $G^d$  is an independent set on  $n$  vertices. Interestingly, by interpreting Harvey and Vondrák's [121] definition of resampling oracles in the variable setting as the resampling of the variables in the scope of an event, we get that  $G^A$  is a directed graph, whose underlying graph is again  $C_n$ . The same is true for  $G^{AI}$ , where we interpret an arc  $f \rightarrow g$  between flaws  $f, g$ , again in the variable framework, as being able to obtain flaw  $g$  by resampling the variables in the scope of flaw  $f$ .

Finally, by performing some trivial computations, we have that:

$$\Pr[E_j] = 2p - p^3$$

and

$$\Pr[E_j \mid \bar{E}_{j+1}] = \frac{p(1-p)^2}{(1-p)^2} = p,$$

$j = 1, \dots, n-1$ . Since  $0 \leq p < 1$ ,  $\Pr[E_j \mid \bar{E}_{j+1}] < \Pr[E_j]$  and thus  $G^d[\mathcal{E}]$  is a lopsidedependency graph, although not a dependency one.  $\diamond$

We now show that the  $d$ -dependency graph is a lopsidedependency graph.

**Lemma 3.2.6.** *Let  $\mathcal{E}$  be a set of events. Then,  $G^d[\mathcal{E}]$  is a lopsidedependency graph for  $\mathcal{E}$ .*

*Proof.* As usual, let  $E = \bigcap_{i \in I} \bar{E}_i$ . Now, in order to obtain a contradiction, suppose that:

$$\Pr[E_j \mid E] > \Pr[E_j]$$

or, equivalently, that:

$$\Pr[E_j \cap E] > \Pr[E_j] \cdot \Pr[E].$$

Then, it holds that:

$$\begin{aligned} \Pr[\bar{E}_j \cap E] &= \Pr[E] - \Pr[E_j \cap E] < \Pr[E] - \Pr[E_j] \cdot \Pr[E] = \\ &\Pr[E](1 - \Pr[E_j]) = \Pr[\bar{E}_j] \cdot \Pr[E]. \end{aligned} \quad (3.10)$$

Since  $E_i$  is not  $d$ -dependent on  $E_j$ , for all  $i \in I$ , it holds that for any assignment  $\mathbf{a}$  that makes  $E_j$  and  $E$  hold, there is no assignment  $\mathbf{b}$  that differs from  $\mathbf{a}$  only in  $\text{sc}(E_j)$  that makes  $\bar{E}_j$  hold.

To obtain a contradiction, it suffices to show that:

$$\Pr[E_j \mid E] \leq \Pr[E_j] \Leftrightarrow \Pr[E_j \cap E] \leq \Pr[E_j] \cdot \Pr[E].$$

Suppose now  $\mathbf{a} = (a_1, \dots, a_l)$ ,  $\beta = (b_1, \dots, b_l)$  are two assignments obtained by independently sampling the random variables twice, once to get  $\mathbf{a}$  and once to get  $\mathbf{b}$ . It holds that:

$$\underbrace{\Pr[E_j \cap E \text{ occurs under } \mathbf{a} \text{ and } \bar{E}_j \text{ occurs under } \mathbf{b}]}_{\text{event } S} = \Pr[E_j \cap E] \cdot \Pr[\bar{E}_j].$$

Let now  $\mathbf{a}' = (a'_1, \dots, a'_l)$ ,  $\mathbf{b}' = (b'_1, \dots, b'_l)$  be two assignments obtained by  $\mathbf{a}$ ,  $\mathbf{b}$  by swapping values in variables in  $\text{sc}(E_j)$ :

- $a'_i = b_i$ , for all  $i$  such that  $X_i \in \text{sc}(E_j)$ ,  $a'_i = a_i$  for the rest,
- $b'_i = a_i$ , for all  $i$  such that  $X_i \in \text{sc}(E_j)$  and  $b'_i = b_i$  for the rest.

Obviously  $\mathbf{a}'$ ,  $\mathbf{b}'$  are two independent samplings of all variables, since all individual variables were originally sampled independently, and we only changed the positioning of the individual variables. Also, under  $\mathbf{a}'$ ,  $\overline{E}_j$  occurs. Since none of the  $E_i$ 's is  $d$ -dependent on  $E_j$ ,  $E$  occurs under  $\mathbf{a}'$ . Also, under  $\mathbf{b}'$ ,  $E_j$  occurs. Thus, it holds that:

$$\Pr[\underbrace{\text{under } \mathbf{a}', \overline{E}_j \cap E \text{ occurs and under } \mathbf{b}', E_j \text{ occurs}}_{\text{event } T}] = \Pr[\overline{E}_j \cap E] \cdot \Pr[E_j] \\ < \Pr[\overline{E}_j] \cdot \Pr[E] \cdot \Pr[E_j],$$

where the last inequality holds by (3.10). Now, by the hypothesis and the construction of  $\mathbf{a}'$ ,  $\mathbf{b}'$ , it also holds that  $S$  implies  $T$ . Thus:

$$\Pr[S] \leq \Pr[T] \Leftrightarrow \Pr[E_j \cap E] \cdot \Pr[\overline{E}_j] \leq \Pr[\overline{E}_j] \cdot \Pr[E] \cdot \Pr[E_j] \\ \Leftrightarrow \Pr[E_j \cap E] \leq \Pr[E_j] \cdot \Pr[E].$$

The last inequality provides the contradiction and the proof is complete.  $\square$

As we have seen, all the lopsidedependency notions in the variable framework we examined give rise to lopsidedependency graphs. The converse is not true. That is, there are sets of events  $\mathcal{E}$  where we can find sparser lopsidedependency graphs than  $G^d[\mathcal{E}]$ . Furthermore, even though we have already seen examples where the lopsidedependency graphs we defined are sparser than a dependency graph, we can also find examples where we can define sparser dependency graphs than  $G^d[\mathcal{E}]$ . We end this section with an example that attests to that.

**Example 3.2.5.** *Let  $X_1$  and  $X_2$  be two independent random variables taking values, uniformly at random, in  $\{0, 1\}$ . Let also  $E_1 = \{X_1 \neq X_2\}$  and  $E_2 = \{X_2 = 0\}$ .*

*First, observe that  $E_1$  is  $d$ -dependent on  $E_2$ . Indeed, let  $\mathbf{a} = (0, 0)$ . Under  $\mathbf{a}$ ,  $E_1$  does not occur and  $E_2$  does. By changing the value of  $X_2 \in \text{sc}(E_2)$ , we obtain the assignment  $\mathbf{b} = (0, 1)$ , under which  $E_1$  occurs and  $E_2$  doesn't. That  $E_2$  is  $d$ -dependent on  $E_1$  follows by taking the assignment  $\mathbf{b}$  and changing the value of  $X_2 \in \text{sc}(E_1)$  to obtain  $\mathbf{a}$ .*



On the other hand, notice that:

$$\Pr[E_1] = \frac{1}{2} = \Pr[E_1 \mid \bar{E}_2].$$

Thus  $E_1$  and  $E_2$  are mutually independent. It follows that  $G^d[\{E_1, E_2\}]$  (and thus  $G^{VDL}[\{E_1, E_2\}]$  and  $G^{MT}[\{E_1, E_2\}]$  too) is the single undirected edge  $\{1, 2\}$ , whereas the graph with vertex set  $\{1, 2\}$  and no edges is a dependency graph (and thus a lopsidedependency graph too).  $\diamond$

### 3.3 The Local Lemma

Recall that, given a set of undesirable events  $\mathcal{E}$ , our aim is to find conditions that guarantee us the existence of a point in the probability space, such that none of the events in  $\mathcal{E}$  occur. To this end, we have the various versions of the *Lovász Local Lemma (LLL)* and Shearer's Lemma, which we state in this section. For a relevant survey, see Szegedy [206].

#### 3.3.1 Main forms of the local lemma

The Lovász Local Lemma (LLL) was originally stated and proved in 1975 by Erdős and Lovász [85]. Its original form is now known as the *symmetric Lovász Local Lemma*.

**Theorem 3.3.1** (Symmetric LLL). *Let  $\mathcal{E} = \{E_1, \dots, E_m\}$  be a set of undesirable events and  $G[\mathcal{E}]$  a dependency graph for  $\mathcal{E}$ . Let also  $|N_j| \leq d$  and  $\Pr[E_j] \leq p$ , for  $j = 1, \dots, m$ ,  $d \in \mathbb{N}$  and  $p \in (0, 1)$ . If*

$$4dp \leq 1$$

then

$$\Pr \left[ \bigwedge_{j=1}^m \bar{E}_j \right] > 0.$$

Thus, given an upper bound to the probability of each event to occur and another for the sizes of their neighborhoods in a dependency graph, LLL gives us a sufficient condition to avoid the events. We will provide a proof for the symmetric version of the LLL as a corollary of the more general, *asymmetric LLL*. A direct proof of the symmetric LLL can be found in Spencer [200].

**Theorem 3.3.2** (Asymmetric LLL). *Let  $\mathcal{E} = \{E_1, \dots, E_m\}$  be a set of undesirable events and  $G[\mathcal{E}]$  a dependency graph for  $\mathcal{E}$ . Let also  $\chi_1, \dots, \chi_m \in (0, 1)$ . If*

$$\Pr[E_j] \leq \chi_j \prod_{i \in N_j} (1 - \chi_i),$$

*for  $j = 1, \dots, m$ , then*

$$\Pr \left[ \bigwedge_{j=1}^m \bar{E}_j \right] \geq \prod_{j=1}^m (1 - \chi_j) > 0.$$

*Proof.* We follow the proof of Alon and Spencer [8]. First, observe that:

$$\begin{aligned} \Pr \left[ \bigwedge_{j=1}^m \bar{E}_j \right] &= \Pr[\bar{E}_1] \cdot \Pr[\bar{E}_2 \mid \bar{E}_1] \cdots \Pr \left[ \bar{E}_m \mid \bigwedge_{j=1}^{m-1} \bar{E}_j \right] = \\ &= (1 - \Pr[E_1]) \cdot (1 - \Pr[E_2 \mid \bar{E}_1]) \cdots \left( 1 - \Pr \left[ E_m \mid \bigwedge_{j=1}^{m-1} \bar{E}_j \right] \right). \end{aligned}$$

Thus, it suffices to show that:

$$\Pr \left[ E_j \mid \bigwedge_{i=1}^{j-1} \bar{E}_i \right] \leq \chi_j,$$

for  $j = 1, \dots, m$ . We show something stronger: for every  $S \subseteq \{1, \dots, m\}$ , such that  $|S| < n$  and any  $j \in \{1, \dots, m\} \setminus S$ :

$$\Pr \left[ E_j \mid \bigwedge_{i \in S} \bar{E}_i \right] \leq \chi_j.$$

We do that by induction on the size of  $S$ . First, assume that  $|S| = 0$ . By hypothesis:

$$\Pr[E_j] \leq \chi_j \prod_{i \in N_j} (1 - \chi_i) \leq \chi_j.$$

Assume that the required holds for all  $S$  such that  $|S| < s$ . For a subset  $S$

with  $s$  elements, let  $S_1 = \{i \in S \mid i \in N_j\}$  and  $S_2 = S \setminus S_1$ . It holds that:

$$\begin{aligned} \Pr \left[ E_j \mid \bigwedge_{i \in S} \bar{E}_i \right] &= \frac{\Pr \left[ E_j \wedge \bigwedge_{i \in S} \bar{E}_i \right]}{\Pr \left[ \bigwedge_{i \in S} \bar{E}_i \right]} \\ &= \frac{\Pr \left[ E_j \wedge \bigwedge_{i \in S_1} \bar{E}_i \mid \bigwedge_{l \in S_2} \bar{E}_l \right]}{\Pr \left[ \bigwedge_{i \in S_1} \bar{E}_i \mid \bigwedge_{l \in S_2} \bar{E}_l \right]}. \end{aligned} \quad (3.11)$$

For the numerator of (3.11), we have:

$$\begin{aligned} \Pr \left[ E_j \wedge \bigwedge_{i \in S_1} \bar{E}_i \mid \bigwedge_{l \in S_2} \bar{E}_l \right] &\leq \Pr \left[ E_j \mid \bigwedge_{l \in S_2} \bar{E}_l \right] = \\ &\Pr[E_j] \leq x_j \prod_{i \in N_j} (1 - x_i), \end{aligned} \quad (3.12)$$

where the first inequality is a basic property in probability, the equality holds since  $E_j$  is mutually independent from the events in  $S_2$  and the last inequality holds by the hypothesis.

For the denominator, if  $S_1 = \emptyset$ , then by (3.11) and (3.12) we get that:

$$\Pr \left[ E_j \mid \bigwedge_{i \in S} \bar{E}_i \right] \leq x_j \prod_{i \in N_j} (1 - x_i) \leq x_j.$$

Otherwise, assume that  $S_1 = \{i_1, \dots, i_t\}$ ,  $t \geq 1$ . Then, it holds that:

$$\begin{aligned} \Pr \left[ \bigwedge_{i \in S_1} \bar{E}_i \mid \bigwedge_{l \in S_2} \bar{E}_l \right] &= \Pr \left[ \bar{E}_{i_1} \wedge \dots \wedge \bar{E}_{i_t} \mid \bigwedge_{l \in S_2} \bar{E}_l \right] \\ &= \Pr \left[ \bar{E}_{i_1} \mid \bigwedge_{l \in S_2} \bar{E}_l \right] \cdots \Pr \left[ \bar{E}_{i_t} \mid \bigwedge_{k=1}^{t-1} \bar{E}_{i_k} \wedge \bigwedge_{l \in S_2} \bar{E}_l \right] = \\ &\left( 1 - \Pr \left[ E_{i_1} \mid \bigwedge_{l \in S_2} \bar{E}_l \right] \right) \cdots \left( 1 - \Pr \left[ E_{i_t} \mid \bigwedge_{k=1}^{t-1} \bar{E}_{i_k} \wedge \bigwedge_{l \in S_2} \bar{E}_l \right] \right) \geq \\ &(1 - x_{i_1}) \cdots (1 - x_{i_t}) \geq \prod_{i \in N_j} (1 - x_i), \end{aligned} \quad (3.13)$$

where the first inequality holds by the induction hypothesis. By combining Eq. (3.11), (3.12) and (3.13), we get the required and thus the proof is complete.  $\square$

We can now obtain Th. 3.3.1 as a corollary of Th. 3.3.2. Note that the condition now is  $e(d+1)p \leq 1$ . Easily this is an improvement for  $d > 2$  and, as Shearer [196] has shown, it is the best possible.

**Corollary 3.3.1** (Symmetric LLL). *Let  $\mathcal{E} = \{E_1, \dots, E_m\}$  be a set of undesirable events and  $G[\mathcal{E}]$  a dependency graph for  $\mathcal{E}$ . Let also  $|N_j| \leq d$  and  $\Pr[E_j] \leq p$ , for  $j = 1, \dots, m$ ,  $d \in \mathbb{N}$  and  $p \in (0, 1)$ . If*

$$e(d+1)p \leq 1$$

then

$$\Pr \left[ \bigwedge_{j=1}^m \bar{E}_j \right] > 0.$$

*Proof.* Again, the proof can be found in [8]. For  $d = 0$ , the condition becomes  $p \leq 1/e$  and the events are mutually independent. The result follows trivially.

For  $d > 0$ , set  $x_j = \frac{1}{d+1}$  for all  $j \in \{1, \dots, m\}$ . By hypothesis, we have:

$$p \leq \frac{1}{d+1} \cdot \frac{1}{e}.$$

Since  $\Pr[E_j] \leq p$ ,  $|N_j| \leq d$ ,  $j = 1, \dots, m$  and:

$$\frac{1}{e} < \left(1 - \frac{1}{d+1}\right)^d,$$

for  $d \geq 1$ , we have:

$$\Pr[E_j] \leq p < \frac{1}{d+1} \cdot \left(1 - \frac{1}{d+1}\right)^d \leq x_j \prod_{i \in N_j} (1 - x_i),$$

for  $j = 1, \dots, m$ .  $\square$

Both Th. 3.3.1 and Th. 3.3.2 can be stated and proven using a lopsidedependency graph  $G$ . We refer the interested reader to the original statement of the lopsidedependency LLL by Erdős and Spencer [86]. We will discuss lopsidedependent versions of the LLL in the sequel.

A sufficient *and also necessary* condition to avoid all events was given by Shearer [196]. Let  $I(G)$  be the set of *independent sets* of a graph  $G$ .

**Theorem 3.3.3** (Shearer’s Lemma). *Let  $\mathcal{E} = \{E_1, \dots, E_m\}$  be a set of undesirable events and  $G := G[\mathcal{E}]$  a dependency graph for  $\mathcal{E}$ . Let also  $\Pr[E_j] = p_j \in (0, 1)$ ,  $j = 1, \dots, m$  and  $\mathbf{p} = (p_1, \dots, p_m)$ . For all  $I \in I(G)$ , if*

$$q_I(G, \mathbf{p}) := \sum_{J \in I(G): I \subseteq J} (-1)^{|J \setminus I|} \prod_{j \in J} p_j > 0,$$

then

$$\Pr \left[ \bigwedge_{j=1}^m \bar{E}_j \right] > 0.$$

$q_I(G, \mathbf{p})$  is called the *independent polynomial* of  $G$ . The reason that Shearer’s Lemma does not make the previous versions of the LLL obsolete, is that it is very difficult to apply it in applications. Computing all the independent sets of a dependency graph is a very hard problem.

Note that Theorem [3.3.3](#) refers to the abstract framework. This means that it is optimal, for the level of generality to which it applies. In the variable framework, a corresponding condition is given by He et al. [\[123\]](#).

### 3.3.2 The Local Lemma in the variable framework

Since  $G^s$  is a dependency graph and  $G^{MT}$ ,  $G^{VDL}$  and  $G^d$  are lopsidedependency graphs, they can all be used in Th. [3.3.1](#) and [3.3.2](#) to obtain the corresponding Theorems. We provide algorithmic proofs for these theorems in Ch [6](#), along with one for Shearer’s Lemma using a symmetric version of the  $d$ -dependency relation.

**Theorem 3.3.4** (Symmetric LLL in the variable framework). *Let  $\mathcal{E} = \{E_1, \dots, E_m\}$  be a set of undesirable events and assume that  $G := G[\mathcal{E}] \in \{G^s, G^{MT}, G^{VDL}, G^d\}$ . Let also  $|N_j| \leq d$  and  $\Pr[E_j] \leq p$ , for  $j = 1, \dots, m$ ,  $d \in \mathbb{N}$  and  $p \in (0, 1)$ . If*

$$e(d + 1)p \leq 1$$

then

$$\Pr \left[ \bigwedge_{j=1}^m \bar{E}_j \right] > 0,$$

*i.e. there exists an assignment of values to the variables  $X_i$  for which none of the events  $E_j$  hold.*

We now use the symmetric LLL in the setting of Ex. [3.2.3](#) to see how big the maximum probability  $p$  can be, in order for a solution to be guaranteed to exist.

**Example 3.3.1.** Consider Ex. [3.2.3](#). Each of the four Bernoulli trials succeeds with probability  $x \in [0, 1)$ , thus:

$$\begin{aligned} \Pr[E_1] &= ((1-x) + x^2) \cdot (1-x), \\ \Pr[E_2] &= \Pr[E_3] = x(1-x). \end{aligned}$$

Easily, the maximum probability is:

$$p = \Pr[E_1] = (-x^3 + 2x^2 - 2x + 1).$$

Let  $d^s$ ,  $d^{MT}$ ,  $d^{VDL}$  and  $d^d$  be the corresponding sizes of the maximum neighborhood in each of the four graphs we consider. We have already seen that  $d^s = 2 = d^{MT}$  and  $d^{VDL} = d^d = 1$ .

Thus, by applying Theorem [3.3.4](#) for each of the two cases, we have that in the former a solution is guaranteed for any  $x > 0.861$ , whereas in the latter, for any  $x > 0.778$ .  $\diamond$

**Theorem 3.3.5** (Asymmetric LLL in the variable framework). Let  $\mathcal{E} = \{E_1, \dots, E_m\}$  be a set of undesirable events and assume that  $G := G[\mathcal{E}] \in \{G^s, G^{MT}, G^{VDL}, G^d\}$ . Let also  $\chi_1, \dots, \chi_m \in (0, 1)$ . If

$$\Pr[E_j] \leq \chi_j \prod_{i \in N_j} (1 - \chi_i),$$

$j = 1, \dots, m$ , then

$$\Pr \left[ \bigwedge_{j=1}^m \bar{E}_j \right] \geq \prod_{j=1}^m (1 - \chi_j) > 0,$$

i.e. there exists an assignment of values to the variables  $X_i$  for which none of the events  $E_j$  hold.

Let us now examine when the asymmetric LLL guarantees the existence of a solution in the setting of Ex. [3.2.4](#).

**Example 3.3.2.** By applying the condition of Theorem 3.3.5 in the events of Ex. 3.2.4, for  $n = 3$ , we get that there exist  $\chi_1, \chi_2, \chi_3 \in (0, 1)$  such that:

$$\begin{aligned} \Pr[E_1] &\leq \chi_1(1 - \chi_2)(1 - \chi_3), \\ \Pr[E_2] &\leq \chi_2(1 - \chi_1)(1 - \chi_3), \\ \Pr[E_3] &\leq \chi_3(1 - \chi_1)(1 - \chi_2). \end{aligned}$$

Thus, for the asymmetric LLL to apply in case we use  $G^s$ ,  $G^{MT}$  or  $G^{VDL}$ , it must hold that:

$$2p - p^2 \leq \chi(1 - \chi)^2,$$

where  $\chi = \min\{\chi_1, \chi_2, \chi_3\}$ . This is maximized for

$$\chi = \frac{2^2}{3^3} = \frac{4}{27},$$

thus  $p$  must be at most 0.077. Using  $G^d$  on the other hand, we get the empty graph, and thus we only need  $p$  to be strictly less than 1.  $\diamond$

For Shearer's Lemma, we need to work with undirected graphs, since independent sets are not defined for directed graphs. We have already seen that the underlying undirected graph of  $G^{VDL}$  is  $G^{MT}$ . Let  $G^{sd}(\mathcal{E})$  be the graph whose:

- vertex set is  $\{1, \dots, m\}$  and
- $E = \{(i, j) \mid E_j \text{ is } d\text{-dependent on } E_i \text{ or } E_i \text{ is } d\text{-dependent on } E_j\}$ .

**Theorem 3.3.6** (Shearer's Lemma in the variable framework). Let  $\mathcal{E} = \{E_1, \dots, E_m\}$  be a set of undesirable events and assume that  $G := G[\mathcal{E}] \in \{G^s, G^{MT}, G^{sd}\}$ . Let also  $\Pr[E_j] = p_j \in (0, 1)$ ,  $j = 1, \dots, m$  and  $\mathbf{p} = (p_1, \dots, p_m)$ . For all  $I \in I(G)$ , if

$$q_I(G, \mathbf{p}) := \sum_{J \in I(G): I \subseteq J} (-1)^{|J \setminus I|} \prod_{j \in J} p_j > 0,$$

then

$$\Pr \left[ \bigwedge_{j=1}^m \bar{E}_j \right] > 0,$$

i.e. there exists an assignment of values to the variables  $X_i$  for which none of the events  $E_j$  hold.

We consider again Ex. 3.2.4.

**Example 3.3.3.** *The situation is as in Ex. 3.3.2. Using  $G^s$  or  $G^{MT}$ , we get as dependency graph the 3-cycle  $(1, 2, 3)$  and henceforth by simple calculations, Shearer's lemma requires that:*

$$1 - 3(2p - p^2) > 0 \Leftrightarrow p < 0.184,$$

*a stronger requirement than the one that suffices to show that the undesirable events can be avoided through our  $d$ -dependency notion. However,  $G^{sd}$  has again no edges, so our version of Shearer's lemma gives  $p < 1$ .  $\diamond$*

### 3.4 Acyclic Edge Coloring

Let  $G = (V, E)$  be a (simple) graph with  $l$  vertices and  $m$  edges (both  $l$  and  $m$  are considered constants) and assume, to avoid trivialities, that  $\Delta > 1$ . An edge-coloring of  $G$  is *proper* if no adjacent edges have the same color. A proper edge-coloring is  *$k$ -acyclic* if there are no *bichromatic*  $k$ -cycles,  $k \geq 3$  and *acyclic* if there are no bichromatic cycles of any length. Note that for a cycle to be bichromatic in a proper coloring, its length must be even. The *acyclic chromatic index* of  $G$ , denoted by  $\chi(G)$ , is the least number of colors needed to produce a proper, acyclic edge-coloring of  $G$ .

In the algorithms of Sec. 6.4, not necessarily proper edge-colorings are constructed by independently selecting one color for each edge from a palette of  $K > 0$  colors, uniformly at random (u.a.r.). Thus, for any edge  $e \in E$  and any color  $i \in \{1, \dots, K\}$ ,

$$\Pr[e \text{ receives color } i] = \frac{1}{K}. \quad (3.14)$$

We assume that we have  $K = \lceil (2 + \epsilon)(\Delta - 1) \rceil$  colors at our disposal, where  $\epsilon > 0$  is an arbitrarily small constant. We show that this number of colors suffice to algorithmically construct, with positive probability, a proper, acyclic edge-coloring for  $G$ . Therefore, since for any  $\Delta$ , there exists a constant  $\epsilon > 0$  such that  $\lceil (2 + \epsilon)(\Delta - 1) \rceil \leq 2(\Delta - 1) + 1 = 2\Delta - 1$ , it follows that  $\chi(G) \leq 2\Delta - 1$ , for any graph  $G$ . In all that follows, we assume the existence of some arbitrary ordering of the edges and the cycles of  $G$ . Edges that in some traversal of an even-length cycle are one edge apart are said to be of the same parity. Among the two consecutive traversals of the edges of a cycle,



we arbitrarily select one and call it *positive*. Given an edge  $e$  and a  $2k$ -cycle  $C$  containing it, we define  $C(e) := \{e = e_1^C, \dots, e_{2k}^C\}$  to be the set of edges of  $C$ , in the positive traversal starting from  $e$ . The two disjoint and equal cardinality subsets of  $C(e)$  comprised of edges of the same parity that are at even (odd, respectively) distance from  $e$  are to be denoted by  $C^0(e)$  ( $C^1(e)$ , respectively).

We now give a cornerstone result proven by Esperet and Parreau [87]:

**Lemma 3.4.1** (Esperet and Parreau [87]). *At any step of any successive coloring of the edges of a graph, there are at most  $2(\Delta - 1)$  colors that should be avoided in order to produce a proper 4-acyclic coloring.*

*Proof Sketch.* Notice that for each edge  $e$ , one has to avoid the colors of all edges adjacent to  $e$ , and moreover for each pair of homochromatic (of the same color) edges  $e_1, e_2$  adjacent to  $e$  at different endpoints (which contribute one to the count of colors to be avoided), one has also to avoid the color of the at most one edge  $e_3$  that together with  $e, e_1, e_2$  define a cycle of length 4. Thus, easily, the total count of colors to be avoided does not exceed the number of adjacent edges of  $e$ , which is at most  $2(\Delta - 1)$ .  $\square$

## 3.5 Coding Theory

In this section we provide some preliminary facts about separating codes. Let  $\mathcal{D}$  be a finite domain of size  $q$ , that, in the language of Coding Theory, is called an *alphabet*. We call the  $N$ -ary vectors of  $\mathcal{D}^N$  *words*.

An  $(N, M)_q$  *code*  $\mathcal{C}$  is a subset of  $\mathcal{D}^N$  of size  $M$ . A word in  $\mathcal{C}$  is called a *code word*. The *Hamming distance* (or simply, distance) between two code words is the number of positions where they differ. The *minimum distance* of  $\mathcal{C}$ , denoted by  $d$ , is defined as the smallest distance between two different code words.

In algebraic coding theory,  $\mathcal{D}$  is usually  $\mathbb{F}_q$ , the finite field with  $q$  elements. In this case, a code  $\mathcal{C}$  is *linear* if it forms a subspace of  $\mathbb{F}_q^n$ . An  $[N, k, d]$ -code is a (linear) code with length  $n$ , dimension  $k$  and minimum distance  $d$ .

Let  $U = \{\mathbf{u}^1, \dots, \mathbf{u}^c\} \subset \mathcal{C}$  be a subset of size  $|U| = c$ , where  $\mathbf{u}^i = (u_1^i, \dots, u_N^i)$ ,  $i = 1, \dots, c$ . We denote by  $U_j = \{u_j^1, \dots, u_j^c\}$  the set of the alphabet elements in the  $j$ -th coordinate of the words in  $U$ . Consider now the following definitions.

**Definition 3.5.1** (Sagalovich [190]). A code  $\mathcal{C}$  is a  $c$ -separating code if for any two disjoint sets  $U$  and  $V$  of code words such that  $|U| \leq c$ ,  $|V| \leq c$  and  $U \cap V = \emptyset$ , there exists at least one coordinate  $j$  such that  $U_j$  and  $V_j$  are disjoint, i.e.  $U_j \cap V_j = \emptyset$ . We say that the coordinate  $j$  separates  $U$  and  $V$ .

**Definition 3.5.2.** Let  $\mathcal{C}$  be an  $(N, M)_q$  code over  $\mathcal{Q}$ . The rate  $R$  of  $\mathcal{C}$  is defined as

$$R = \frac{\log_q M}{N}. \quad (3.15)$$

Let  $R(N, c)_q$  be the optimal rate of a  $c$ -separating  $(N, M)_q$  code. We are interested in the asymptotic rate:

$$\underline{R}_q(c) = \liminf_{N \rightarrow \infty} R_q(N, c). \quad (3.16)$$

As stated in the introduction we focus exclusively on binary  $c$ -separating codes. We will discuss the general case  $c \geq 2$  and we will also particularize our results on the widely discussed case of  $c = 2$ .

The existence of binary  $c$ -separating codes of positive asymptotic rate is already known. For completeness we provide the proof below.

**Proposition 3.5.1** (Barg et al. [13]). There exist binary  $c$ -separating codes of length  $n$  and size  $\frac{1}{2}(1 - 2^{-(2c-1)})^{-n/(2c-1)}$  and thus:

$$\underline{R}(N, c)_2 \geq -\frac{\log_2(1 - 2^{-(2c-1)})}{2c-1} - \frac{1}{N}. \quad (3.17)$$

*Proof.* Let  $\mathcal{C}$  be a random binary  $(N, M)$  code. We consider pairs of sets of at most  $c$  code words of  $\mathcal{C}$ . The probability  $\Pr[E_{U,V}]$  that two such sets  $U$  and  $V$  are not separated is:

$$(1 - 2^{-(2c-1)})^N.$$

Then the expected number  $E(N_s)$  of pairs of sets in  $\mathcal{C}$  that are not separated is:

$$E(N_s) \leq \binom{M}{c} \binom{M-c}{c} (1 - 2^{-(2c-1)})^N.$$

Using well known approximations, we have:

$$E(N_s) \leq \frac{M^c}{c!} \frac{(M-c)^c}{c!} (1 - 2^{-(2c-1)})^N \leq \frac{M^{2c}}{c!c!} (1 - 2^{-(2c-1)})^N.$$

Take an  $M$  such that  $E(N_s) < M/2$ . Then by removing a code word from each pair of sets that is not separated, we are left with at least  $M - (M/2) = M/2$  of our original code words and with all pairs of sets separated. It holds that:

$$\frac{M^{2c}}{c!c!} (1 - 2^{-(2c-1)})^n < \frac{M}{2} \Rightarrow M^{2c-1} < \frac{c!c!}{2} (1 - 2^{-(2c-1)})^{-N},$$

thus, by taking

$$M < \left( \frac{c!c!}{2} \right)^{\frac{1}{2c-1}} (1 - 2^{-(2c-1)})^{-\frac{N}{2c-1}},$$

there exists a  $c$ -separating,  $(N, M/2)$ -code.

This means that there exists a code with rate:

$$\underline{R}(N, c)_2 \geq \frac{\log_2(M/2)}{N} = \frac{\log_2 \left( \frac{1}{2} \left( \frac{c!c!}{2} \right)^{\frac{1}{2c-1}} (1 - 2^{-(2c-1)})^{-\frac{N}{2c-1}} \right)}{N}.$$

For the right side of the previous equation, we have that:

$$\begin{aligned} \frac{\log_2(1/2)}{N} + \frac{1}{N} \frac{1}{2c-1} \log_2 \left( \frac{c!c!}{2} \right) - \frac{\log_2(1 - 2^{-(2c-1)})}{2c-1} \\ \geq -\frac{\log_2(1 - 2^{-(2c-1)})}{2c-1} - \frac{1}{N}. \end{aligned}$$

□

Focusing on the rate of the binary 2-separating codes, we have the following corollary:

**Corollary 3.5.1** (Sagalovich [190]). *There exist binary 2-separating codes of rate:*

$$\underline{R}_2(2) \geq 1 - \log_2(7/8) = 0.0642.$$

We proceed now with two examples of well known and extensively studied codes.

**Definition 3.5.3** (MacWilliams & Sloane. [168]). *The binary simplex code  $\mathcal{S}_k$  is a  $(2^k - 1, k, 2^{k-1})$ -code which is the dual of the  $(2^k - 1, 2^k - 1 - k, 3)$ -Hamming code.*

$\mathcal{S}_k$  consists of the all-zero vector  $\mathbf{0}$  and  $2^k - 1$  code words of weight  $2^{k-1}$ . It is called a simplex code, because every pair of code words is at the same distance apart.

For this family of codes, we have the following lemma, which we state here without proof.

**Lemma 3.5.1.** *The binary simplex code is 2-separating.*

The first families of algebraic geometric codes (AG) were constructed by V.D. Goppa [109], using the theory of algebraic curves. We denote by  $N$ , the number of points with coordinates in the finite field of elements over which the curve is defined. The rate of an  $(N, M)_q$  AG code with minimum distance  $d$  satisfies:

$$R \geq 1 - \frac{d}{N} - \frac{g}{N}, \quad (3.18)$$

where  $g$  is the genus of the curve over which the code is defined. The lower the ratio  $g/N$  the better the rate. Unfortunately, by the Drinfeld-Vlăduț bound  $g/N$  is lower bounded:

$$\liminf_{g \rightarrow \infty} \frac{g}{N} \geq \frac{1}{\sqrt{q} - 1}. \quad (3.19)$$

The Drinfeld-Vlăduț bound has been achieved by two explicitly described sequences of curves [96, 97]. Therefore, AG codes of asymptotic rate:

$$R \geq 1 - \frac{d}{N} - \frac{1}{\sqrt{q} - 1} \quad (3.20)$$

can be constructed.

# Chapter 4

## Universal Algebra and Computational Complexity

The algebraic approach consists of obtaining results regarding the computational complexity of CSP's through universal algebraic tools. We begin by studying, in Sec. [4.1](#), the well known sets of operators over finite domains called *clones*. Then, in Sec. [4.2](#), we provide a brief overview of results in the computational complexity of CSP's and MCSP's. We mainly focus in *dichotomy theorems*, that provide necessary and sufficient conditions for a CSP to be tractable.

### 4.1 Clone Theory

In this section, we discuss the properties of sets of operators and relations over a finite domain set and the central algebraic notion of operators that *preserve* relations. A big part of what follows works also in the case of infinite domains. We indicate the parts where the presented theory does not cover the infinite case and provide some suitable references.

Given a finite domain  $\mathcal{D}$  and a set  $F$  of finitary operations on  $\mathcal{D}$ , the pair  $(\mathcal{D}, F)$  is a *universal algebra*. We are interested in specific sets of operations  $F$ , that are called *clones*. The main reason that clones are (algebraically) interesting, is that two algebras whose sets of finitary operations give rise to the same clone, are in a way equivalent (they have the same sub-algebras, endomorphisms, congruences etc.). For general information on clones, we propose the work of Szendrei [\[208\]](#) and the short exposition of Kerkhoff et

al. [138]. As we will see in the sequel, the set of clones over any domain  $\mathcal{D}$  has the strong structural property of being a *lattice*. The book of Davey and Priestley [64] is a very good starting point on this subject. For an analytical and easy to follow exposition of the Boolean case, that is, the case where  $\mathcal{D}$  has only two elements, see Bohler et al. [30,31].

### 4.1.1 Operators on finite domains

Assume we have a finite domain  $\mathcal{D}$ . A  $k$ -ary operation over  $\mathcal{D}$  is a function  $f : \mathcal{D}^k \mapsto \mathcal{D}$ , for  $k \in \mathbb{N}$ . The *domain* of  $f$  is denoted by  $\text{Dom}(f) = \mathcal{D}^k$ , and its image by  $\text{Im}(f) \subseteq \mathcal{D}$ . Under this notation, a  $k$ -ary *partial* function  $f$  on  $\mathcal{D}$  is an operator where  $\text{Dom}(f) \subseteq \mathcal{D}^k$ . Trivially, every operator is partial. For any  $I = \{i_1, \dots, i_l\} \subseteq \{1, \dots, k\}$ , we denote the restriction of a vector  $\mathbf{a} = (a_1, \dots, a_k) \in \mathcal{D}^k$  to  $I$  by:

$$\mathbf{a}_I := (a_{i_1}, \dots, a_{i_l}).$$

For a subset  $R \subseteq \mathcal{D}^k$ , we again have its restriction:

$$R_I := \{\mathbf{a}_I \mid \mathbf{a} \in R\}$$

and, finally, the restriction of  $f$  to a subset  $A \subseteq \mathcal{D}^k$  is the operator  $f|_A : A^k \mapsto A$ , where, for every  $\mathbf{a} \in A$ ,

$$f|_A(\mathbf{a}) = f(\mathbf{a}).$$

We now consider various operators and classes of operators we will need in the sequel.

First assume that  $\mathcal{D}$  is Boolean. We can suppose that  $\mathcal{D} = \{0, 1\}$  without loss of generality. We use the following Boolean operators on  $\{0, 1\}$ .

1. Binary *symmetric* operators:

$$\wedge(x, y) := \begin{cases} 1, & \text{if } x = y = 1, \\ 0, & \text{else} \end{cases} \quad (4.1)$$

and

$$\vee(x, y) := \begin{cases} 0, & \text{if } x = y = 0, \\ 1, & \text{else} \end{cases} \quad (4.2)$$

2. Ternary *majority* operator:

$$\text{maj}(x, y, z) := \begin{cases} x, & \text{if } x = y \text{ or } x = z, \\ y, & \text{else.} \end{cases} \quad (4.3)$$

3. Ternary *affine* operator:

$$\oplus(x, y, z) := \begin{cases} x, & \text{if } y = z, \\ y, & \text{if } x = z, \\ z, & \text{else.} \end{cases} \quad (4.4)$$

4. Ternary symmetric operators:

$$\wedge^{(3)}(x, y, z) := \wedge(\wedge(x, y), z), \quad (4.5)$$

$$\vee^{(3)}(x, y, z) := \vee(\vee(x, y), z), \quad (4.6)$$

Furthermore, we say that an operation  $f : \{0, 1\}^k \mapsto \{0, 1\}$  is:

- *invariant under permutations of its input*, if for all  $(a_1, \dots, a_k) \in \{0, 1\}^k$  and for all permutations  $p : \{1, \dots, k\} \mapsto \{1, \dots, k\}$ :

$$f(a_1, \dots, a_k) = f(a_{p(1)}, \dots, a_{p(k)}), \quad (4.7)$$

- *monotone*, if it holds that for any index  $i \in \{1, \dots, k\}$  and for all vectors  $(a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_k) \in \{0, 1\}^{k-1}$ :

$$\begin{aligned} f(a_1, \dots, a_{i-1}, 0, a_{i+1}, \dots, a_k) = 1 &\Rightarrow \\ f(a_1, \dots, a_{i-1}, 1, a_{i+1}, \dots, a_k) = 1 &\quad (4.8) \end{aligned}$$

and

- *linear*, if there exist constants  $c_0, c_1, \dots, c_k \in \{0, 1\}$  such that for all  $(a_1, \dots, a_k) \in \{0, 1\}^k$ :

$$f(a_1, \dots, a_k) = c_0 \oplus c_1 a_1 \oplus \dots \oplus c_k a_k. \quad (4.9)$$

For an arbitrary domain  $\mathcal{D}$  of finite cardinality, let  $f : \mathcal{D}^k \mapsto \mathcal{D}$  be a  $k$ -ary operator.

We say that  $f$  is *idempotent* or *unanimous*, if  $f(a, \dots, a) = a$ , for all  $a \in \mathcal{D}$ . It is *supportive* or *conservative*, if  $f(a_1, \dots, a_k) \in \{a_1, \dots, a_k\}$ , for all  $a_1, \dots, a_k \in \mathcal{D}$ . Note that if  $\mathcal{D}$  is Boolean, these notions coincide. Furthermore, the only unary idempotent operation is the *identify* operator  $\text{id} : \mathcal{D} \mapsto \mathcal{D}$ , where  $\text{id}(a) := a$ , for all  $a \in \mathcal{D}$ . Also,  $f$  is *constant*, if there is some  $c \in \mathcal{D}$  such that  $f(a_1, \dots, a_k) = c$ , for all  $a_1, \dots, a_k \in \mathcal{D}$ .

We call  $f$  a *projection* (operator), if there exists a  $d \in \{1, \dots, k\}$  such that  $f(a_1, \dots, a_k) = a_d$ , for all  $(a_1, \dots, a_k) \in \mathcal{D}^k$ . In such a case, we denote  $f$  by  $\text{pr}_d^k$ . Note that a projection is both idempotent and conservative.  $f$  is a *semi-projection* if there exists a  $d \in \{1, \dots, k\}$  such that  $f(a_1, \dots, a_k) = a_d$ , for all  $(a_1, \dots, a_k) \in \mathcal{D}^k$  such that  $|\{a_1, \dots, a_k\}| < k$ . In that notation, a projection is a *trivial* semi-projection.

A binary operator  $f : \mathcal{D}^2 \mapsto \mathcal{D}$  is *symmetric*, if for all  $x, y \in \mathcal{D}$ ,  $f(x, y) = f(y, x)$ . A ternary operator  $f : \mathcal{D}^3 \mapsto \mathcal{D}$  is a

- a *majority* operator if, for all  $x, y \in \mathcal{D}$ :

$$f(x, x, y) = f(x, y, x) = f(y, x, x) = x, \quad (4.10)$$

- a *minority* operator if, for all  $x, y \in \mathcal{D}$ :

$$f(x, x, y) = f(x, y, x) = f(y, x, x) = y, \quad (4.11)$$

and,

- a *weak-near unanimity (WNU)* operator if, for all  $x, y \in \mathcal{D}$ :

$$f(x, x, y) = f(x, y, x) = f(y, x, x). \quad (4.12)$$

## 4.1.2 Clones and co-clones

Let  $\mathcal{O}_{\mathcal{D}}^{(k)}$  denote the set of all  $k$ -ary operations on  $\mathcal{D}$ ,  $\mathcal{J}_{\mathcal{D}}^{(k)}$  the set of all  $k$ -ary projections over  $\mathcal{D}$  and set:

$$\begin{aligned} \mathcal{O}_{\mathcal{D}} &:= \bigcup_{k \in \mathbb{N}} \mathcal{O}_{\mathcal{D}}^{(k)}, \\ \mathcal{J}_{\mathcal{D}} &:= \bigcup_{k \in \mathbb{N}^*} \mathcal{J}_{\mathcal{D}}^{(k)}, \end{aligned}$$



where  $\mathbb{N}^* := \mathbb{N} \setminus \{0\}$ . Also, let  $\mathbf{P}(A)$  denote the *powerset* of a set  $A$ , that is, the set containing all of its subsets. Set:

$$\mathcal{R}_{\mathcal{D}} := \left( \bigcup_{n \in \mathbb{N}^*} \mathbf{P}(\mathcal{D}^n) \right) \setminus \emptyset,$$

that is, the set of all finite relations over  $\mathcal{D}$ , excluding the empty one.

We consider now a way to combine operators in order to produce new ones, called the *superposition* of operators. We are interested in superposition of operator since it preserves the property of an operator to be a *polymorphism* of a relation (see Subsec. [4.1.4](#)).

**Definition 4.1.1.** *Let  $f \in \mathcal{O}_{\mathcal{D}}^{(k)}$  and  $g_1, \dots, g_k \in \mathcal{O}_{\mathcal{D}}^{(l)}$ . The superposition of  $f, g_1, \dots, g_k$  is the  $l$ -ary operator  $h := f(g_1, \dots, g_k)$ , where:*

$$h(x_1, \dots, x_l) := f(g_1(x_1, \dots, x_l), \dots, g_k(x_1, \dots, x_l)), \text{ for all } x_1, \dots, x_l \in \mathcal{D}.$$

Given a set of operations  $F$ , we consider the set containing all the operations that can be produced by the superposition of operators in  $F$  and projections.

**Definition 4.1.2.** *Let  $F \subseteq \mathcal{O}_{\mathcal{D}}$  be a set of functions on  $\mathcal{D}$ .  $F$  is a clone if:*

1.  $\mathcal{J}_{\mathcal{D}} \subseteq F$ , that is,  $F$  contains all the projections and
2.  $F$  contains all functions  $h \in \mathcal{O}_{\mathcal{D}}^{(l)}$ ,  $l \in \mathbb{N}$ , for which there exist a  $k$ -ary  $f \in F$  and  $k$   $l$ -ary  $g_1, \dots, g_k \in F$ , such that  $h$  is their superposition, that is  $h = f(g_1, \dots, g_k)$ .

For any set of functions  $F \subseteq \mathcal{O}_{\mathcal{D}}$ , let:

$$[F] := \bigcap \{G \subseteq \mathcal{O}_{\mathcal{D}} \mid G \supseteq F \text{ and } G \text{ is a clone}\} \quad (4.13)$$

be the *least clone containing  $F$* , or the clone *generated by  $F$* .

**Lemma 4.1.1.** *For any set of functions  $F \subseteq \mathcal{O}_{\mathcal{D}}$ ,  $[F]$  is a clone.*

*Proof.* By definition, for any clone  $G$ ,  $\mathcal{J}_{\mathcal{D}} \subseteq G$ . Thus,  $\mathcal{J}_{\mathcal{D}} \subseteq [F]$ . Now, let  $h \in [F] \cap \mathcal{O}_{\mathcal{D}}^{(k)}$  and  $g_1, \dots, g_k \in [F] \cap \mathcal{O}_{\mathcal{D}}^{(l)}$ , where  $k, l \in \mathbb{N}$ . Then,  $f, g_1, \dots, g_k$  are in all clones  $G$  such that  $G \supseteq F$ . Again by definition,  $h = f(g_1, \dots, g_k)$  in all such  $G$  and thus in  $[F]$  too.  $\square$

Thus, for each  $F \subseteq \mathcal{O}_{\mathcal{D}}$ , the clone generated by  $F$  is a well-defined object. Also, it is straightforward to observe that  $F \subseteq \mathcal{O}_{\mathcal{D}}$  is a clone if and only if  $F = [F]$ . We proceed with some examples.

**Example 4.1.1.** *We consider various sets of functions on  $\mathcal{D}$  and the clones they generate.*

- i. *Obviously,  $\mathcal{O}_{\mathcal{D}}$  is a clone, since  $[\mathcal{O}_{\mathcal{D}}] = \mathcal{O}_{\mathcal{D}}$ .*
- ii. *Consider the projection functions  $\text{pr}_i^k, \text{pr}_{j_1}^l, \dots, \text{pr}_{j_k}^l$ , where  $k, l \in \mathbb{N}^*$ ,  $i \in \{1, \dots, k\}$  and  $j_1, \dots, j_k \in \{1, \dots, l\}$ . Then, for all  $x_1, \dots, x_l \in \mathcal{D}$ , it holds that:*

$$\begin{aligned} & (\text{pr}_i^k(\text{pr}_{j_1}^l, \dots, \text{pr}_{j_k}^l))(x_1, \dots, x_l) = \\ & \text{pr}_i^k(\text{pr}_{j_1}^l(x_1, \dots, x_l), \dots, \text{pr}_{j_k}^l(x_1, \dots, x_l)) = \\ & \text{pr}_i^k(x_{j_1}, \dots, x_{j_k}) = x_{j_i}. \end{aligned}$$

*Thus,  $\text{pr}_i^k(\text{pr}_{j_1}^l, \dots, \text{pr}_{j_k}^l) = \text{pr}_{j_i}^k$ . Since  $k, l, i, j_1, \dots, j_k$  where all chosen arbitrarily, it follows that the superposition of projections is always a projection function. This means that  $[\mathcal{J}_{\mathcal{D}}] = \mathcal{J}_{\mathcal{D}}$  and, consequently, that  $\mathcal{J}_{\mathcal{D}}$  is a clone.*

- iii. *Let  $\text{Id} := \{f \in \mathcal{O}_{\mathcal{D}} \mid f \text{ is idempotent}\}$ .  $\text{Id}$  is a clone. Indeed all projections are obviously idempotent and superposition again easily preserves idempotency. Thus  $[\text{Id}] = \text{Id}$ .*
- iv.  *$\mathcal{J}_{\mathcal{D}} \cup \{\wedge\}$  is not a clone, since  $\wedge^{(3)} \notin \mathcal{J}_{\mathcal{D}} \cup \{\wedge\}$ , although it can be constructed by superposition from operators in  $\mathcal{J}_{\mathcal{D}} \cup \{\wedge\}$ . Indeed, first consider the ternary operator  $\wedge^+ := \wedge(\text{pr}_1^3(x, y, z), \text{pr}_2^3(x, y, z))$ , that ignores the third element of its input and is again equal to 1 if and only if  $x = y = 1$ . Obviously,  $\wedge^+ \in [\{\wedge\}]$ . Furthermore,  $\wedge^{(3)}(x, y, z) := \wedge(\wedge^+(x, y, z), \text{pr}_3^3(x, y, z))$  and thus  $\wedge^{(3)} \in [\{\wedge\}]$ .*

A subset  $F \subseteq \mathcal{O}_{\mathcal{D}}$  is called *complete* if  $[F] = \mathcal{O}_{\mathcal{D}}$ . A clone  $F$  is *maximal*, if for any  $f \notin F$ ,  $F \cup \{f\}$  is complete. Finally, a clone  $F$  is *minimal* if the only proper subset of  $F$  that is a clone is  $\mathcal{J}_{\mathcal{D}}$ . All these notions have been extensively studied in trying to identify the structure of the set of clones over a domain  $\mathcal{D}$ .

We turn now our attention to sets  $\mathcal{R}$  of relations over  $\mathcal{D}$ . Recall that an  $n$ -ary relation of  $\mathcal{D}$  is a subset of  $\mathcal{D}^n$  and that we denote the set of all

relation over  $\mathcal{D}$ , apart the empty one, by  $\mathcal{R}_{\mathcal{D}}$ . Let  $R \in \mathcal{R}_{\mathcal{D}}$  be an  $n$ -ary relation,  $n \in \mathbb{N}^*$ . An element  $\mathbf{a} \in R$  is an  $n$ -ary vector  $(a_1, \dots, a_n)$ . Given  $k$  such vectors  $\mathbf{a}^1, \dots, \mathbf{a}^k$ , it is convenient to view them as a  $k \times n$  matrix  $A$  over  $\mathcal{D}$ , where  $\mathbf{a}^i$  is the  $i$ -th row of  $A$ ,  $i = 1, \dots, k$ . Under this notation, the element at the  $i$ -th row and  $j$ -th column of  $A$  is  $a_j^i$ ,  $i = 1, \dots, k$  and  $j = 1, \dots, n$ . Furthermore, we use  $\mathbf{a}_j$  to denote the  $j$ -th column of  $A$ , i.e.:

$$\begin{aligned}\mathbf{a}^i &= (a_1^i, \dots, a_n^i), \\ \mathbf{a}_j &= (a_j^1, \dots, a_j^k)^T,\end{aligned}$$

for  $i = 1, \dots, k$  and  $j = 1, \dots, n$ .

For arbitrary and possibly infinite domains  $\mathcal{D}$ , Pöschel [184, 185] defines sets of relations that are *closed under general superposition*. This is a rather technical definition which is not very easy to use in practice. Fortunately, since we are interested primarily in finite domains, we can use an alternative definition. First, we need some notation.

For all  $I \subseteq \{1, \dots, n\}^2$ , let the  $n$ -ary *equivalence relation on  $I$*  be:

$$E_I^n := \{(a_1, \dots, a_n) \in \mathcal{D}^n \mid a_i = a_j, \forall i, j \in I\}. \quad (4.14)$$

Let  $\pi[n]$  be the set of permutations of the set  $\{1, \dots, n\}$ . Let also  $R \subseteq \mathcal{D}^n$  be an  $n$ -ary relation. We say that  $Q$  is a *permutation* of  $R$  and write  $R \approx Q$ , if:

$$Q := \{(a_{i_1}, \dots, a_{i_n}) \mid (a_1, \dots, a_n) \in R \text{ and } (i_1, \dots, i_n) \in \pi[n]\}. \quad (4.15)$$

Furthermore, the *projection* of  $R$  to  $I \subseteq \{1, \dots, n\}$ , is the relation:

$$R_I := \{(a_{i_1}, \dots, a_{i_k}) \mid (a_1, \dots, a_n) \in R \text{ and } I = \{i_1, \dots, i_k\}\}. \quad (4.16)$$

**Definition 4.1.3.** *A set of relations  $\mathcal{R}$  over a finite domain  $\mathcal{D}$  is a co-clone if and only if the following hold:*

- (i)  $\mathcal{R}$  contains all equivalence relations, for all  $n \in \mathbb{N}^*$  and  $I \subseteq \{1, \dots, n\}^2$ .
- (ii)  $\mathcal{R}$  is closed under direct Cartesian products, that is, for all  $R \subseteq \mathcal{D}^n$  and  $Q \subseteq \mathcal{D}^m$  such that  $R, Q \in \mathcal{R}$ , the Cartesian product:

$$R \times Q := \{(a_1, \dots, a_n, b_1, \dots, b_m) \mid (a_1, \dots, a_n) \in R, (b_1, \dots, b_m) \in Q\},$$

is also in  $\mathcal{R}$ .

(iii)  $\mathcal{R}$  is closed under intersections, i.e. for any  $\mathcal{T} \subseteq \mathcal{R}$  such that all  $R \in \mathcal{T}$  have the same arity  $n$ :

$$\bigcap_{R \in \mathcal{T}} R \in \mathcal{R}.$$

(iv)  $\mathcal{R}$  is closed under permutations, i.e. if  $R \in \mathcal{R}$  and  $Q \approx R$ , then  $Q \in \mathcal{R}$ .

(v)  $\mathcal{R}$  is closed under projections, i.e. if  $R \in \mathcal{R}$  is an  $n$ -ary relation and  $I \subseteq \{1, \dots, n\}$ ,  $R_I \in \mathcal{R}$ .

To get a sense as to how we can combine relations of a co-clone in order to produce other ones inside the co-clone, we prove the following technical result, that we will need in the sequel.

**Corollary 4.1.1.** *Let  $\mathcal{R}$  be a co-clone on  $\mathcal{D}$ . Then  $\mathcal{D}^n \in \mathcal{R}$ , for all  $n \in \mathbb{N}$ .*

*Proof.* Take any equivalence relation  $E_I^n$ , and project it to some  $j \notin I$ , to obtain  $\mathcal{D}$ . Then, take the Cartesian product of  $\mathcal{D}$  with itself  $n$  times.  $\square$

As we did with clones, we define the *least co-clone* that contains a given set of relations:

$$\langle \mathcal{R} \rangle := \bigcap \{ \mathcal{S} \text{ is a set of relations} \mid \mathcal{S} \supseteq \mathcal{R} \text{ and } \mathcal{S} \text{ is a co-clone} \}. \quad (4.17)$$

**Lemma 4.1.2.** *For any set of relations  $\mathcal{R}$ ,  $\langle \mathcal{R} \rangle$  is a co-clone.*

*Proof.* Obvious, by Eq. [4.17](#) and Definition [4.1.3](#).  $\square$

Thus, for each set of relations  $\mathcal{R}$ , the least co-clone containing  $\mathcal{R}$ , or the co-clone generated by  $\mathcal{R}$ , is a well-defined object. Also, it is immediate to observe that  $\mathcal{R}$  is a co-clone if and only if  $\mathcal{R} = \langle \mathcal{R} \rangle$ . We proceed with some examples.

**Example 4.1.2.** *We consider various subsets  $\mathcal{R} \subseteq \mathcal{R}_{\mathcal{D}}$  and the co-clones they generate.*

i. Obviously,  $\bigcup_{n \in \mathbb{N}} \mathcal{D}^n$  is a co-clone, since  $\langle \bigcup_{n \in \mathbb{N}} \mathcal{D}^n \rangle = \bigcup_{n \in \mathbb{N}} \mathcal{D}^n$ .

ii. It is easy to observe that by intersecting, permuting, projecting or taking the Cartesian product of equivalence relations, we end up again with equivalence relations. Thus:

$$\{E_I^n \mid n \in \mathbb{N}^* \text{ and } I \subseteq \{1, \dots, n\}^2\}$$

is a co-clone.

iii. Let  $\mathcal{R}, \mathcal{S}$  be two arbitrary sets of relations. Then

$$\langle \mathcal{R} \rangle \times \langle \mathcal{S} \rangle \subseteq \langle \mathcal{R} \times \mathcal{S} \rangle.$$

Indeed, let  $R \in \langle \mathcal{R} \rangle \times \langle \mathcal{S} \rangle$ . Then,  $R = P \times Q$ , for some  $P \in \langle \mathcal{R} \rangle$  and  $Q \in \langle \mathcal{S} \rangle$ . Then, it must be the case that there exist relations  $P_1, \dots, P_s \in \mathcal{R}$  with arities  $p_1, \dots, p_s$  respectively and  $Q_1, \dots, Q_t \in \mathcal{S}$  with arities  $q_1, \dots, q_t$  respectively, such that  $P$  and  $Q$  are produced by Cartesian products, intersections and/or projections of  $P_1, \dots, P_s$  and  $Q_1, \dots, Q_t$  respectively.

Easily,  $P_i, Q_j \in \langle \mathcal{R} \times \mathcal{S} \rangle$ , for  $i = 1, \dots, s$  and  $j = 1, \dots, t$ . Indeed, obviously  $P_i \times Q_j \in \mathcal{R} \times \mathcal{S}$  for any  $i \in I := \{1, \dots, s\}$  and  $j \in J := \{1, \dots, t\}$  and

$$P_i = (P_i \times Q_j)_I, \quad Q_j = (P_i \times Q_j)_J.$$

It follows that  $P, Q \in \langle \mathcal{R} \times \mathcal{S} \rangle$  and thus, so does their Cartesian product  $R = P \times Q$ .

The opposite inclusion does not always hold. For  $P$  or  $Q$  above to be in  $\langle \mathcal{R} \rangle \times \langle \mathcal{S} \rangle$ , it must be the case that they are themselves Cartesian products, which is not true in most cases. But we have shown that  $P, Q \in \langle \mathcal{R} \times \mathcal{S} \rangle$ .  $\square$

Let  $\mathcal{L}_{\mathcal{D}}^o$  and  $\mathcal{L}_{\mathcal{D}}^r$  be the sets of all clones and co-clones over  $\mathcal{D}$  respectively. As we see in the next subsection, both these sets, under the relation of set-inclusion have an interesting structure, that of a complete lattice.

### 4.1.3 Lattices

For a finite set  $A$ , we say that  $\sqsubseteq$  is a *partial order* on  $A$  if it satisfies:

- (i) *reflexivity*:  $a \sqsubseteq a$ , for all  $a \in A$ ,
- (ii) *antisymmetry*:  $a \sqsubseteq b$  and  $b \sqsubseteq a$  imply  $a = b$ , for all  $a, b \in A$  and
- (iii) *transitivity*:  $a \sqsubseteq b$  and  $b \sqsubseteq c$  imply  $a \sqsubseteq c$ , for all  $a, b, c \in A$ .

Furthermore  $\sqsubseteq$  is a *total ordering* on  $A$ , if for all  $a, b \in A$ , either  $a \sqsubseteq b$  or  $b \sqsubseteq a$ . For example,  $\leq$  is a total ordering of  $\mathbb{N}$  and  $\subseteq$  a partial order of its powerset  $\mathbf{P}(\mathbb{N})$ .

We say that  $b$  covers  $a$  in  $(A, \sqsubseteq)$  if (i)  $a \neq b$  (ii)  $a \sqsubseteq b$  and (iii) for all  $c \neq b$  such that  $a \sqsubseteq c \sqsubseteq b$ , it holds that  $c = a$ .  $A$  has a *bottom* element, denoted by  $\perp$ , if there exists some  $a \in A$  such that  $a \sqsubseteq b$ , for all  $b \in A$ . It has a *top* element, denoted by  $\top$ , if there exists some  $a \in A$  such that  $a \sqsupseteq b$ , for all  $b \in A$ . Easily, the top and bottom elements are unique if they exist.

**Definition 4.1.4.** Let  $(A, \sqsubseteq)$  be a (partially) ordered set with top and bottom elements. An element  $a \in A$  is an *atom* if it covers  $\perp$  and a *coatom* if it is covered by  $\top$ .

A subset  $B$  of  $A$  is a *chain*, if for all  $a, b \in B$ , either  $a \sqsubseteq b$  or  $b \sqsubseteq a$ . Thus a totally ordered set  $A$  is a chain.  $B$  is an *anti-chain* if for all  $a, b \in B$ ,  $a \sqsubseteq b$  implies  $a = b$ .

Assume that  $(A, \sqsubseteq_A)$ ,  $(B, \sqsubseteq_B)$  are two (partially) ordered sets. An *order-isomorphism* is a function  $h : A \mapsto B$  such that:

$$a \sqsubseteq_A b \text{ if and only if } h(a) \sqsubseteq_B h(b), \text{ for all } a, b \in A. \quad (4.18)$$

It is easy to see that an order-isomorphism  $h$  is necessarily a bijection and that its inverse  $h^{-1} : B \mapsto A$  is also an order-isomorphism.

Let  $B \subseteq A$ . An *upper bound* for  $B$  is an element  $a \in A$  such that  $b \sqsubseteq a$ , for all  $b \in B$ . A *lower bound* for  $B$  is an element  $a \in A$  such that  $a \sqsubseteq b$ , for all  $b \in B$ . Let:

$$B^u := \{a \in A \mid a \text{ is an upper bound for } B\}, \quad (4.19)$$

$$B^l := \{a \in A \mid a \text{ is a lower bound for } B\}. \quad (4.20)$$

The *least upper bound or supremum* of  $B$ , if it exists, is the least element of  $B^u$ . Analogously, the *greatest lower bound or infimum* of  $B$ , if it exists, is the greatest element of  $B^l$ . We denote them by  $\sup B$  and  $\inf B$  respectively.

Using a notation more common to the field of universal algebra, we say that the *join* of  $B$ , when it exists, is:

$$\bigvee_{(A, \sqsubseteq_A)} B := \sup B \quad (4.21)$$

and the *meet* of  $B$ , when it exists, is:

$$\bigwedge_{(A, \sqsubseteq_A)} B := \inf B. \quad (4.22)$$

Also, when  $B$  has only two elements, say  $a$  and  $b$ , we write  $a \vee b$  and  $a \wedge b$  instead of  $\bigvee_{(A, \sqsubseteq_A)} \{a, b\}$  and  $\bigwedge_{(A, \sqsubseteq_A)} \{a, b\}$ .

**Definition 4.1.5.** Let  $(A, \sqsubseteq)$  be an ordered set.

- (i) If  $a \vee b$  and  $a \wedge b$  exist for all  $a, b \in A$ ,  $(A, \sqsubseteq)$  is a lattice.
- (ii) If  $\bigvee_{(A, \sqsubseteq)} B$  and  $\bigwedge_{(A, \sqsubseteq)} B$  exist for all  $B \subseteq A$ , then  $(A, \sqsubseteq)$  is a complete lattice.

**Example 4.1.3.** Let  $(\mathbb{N}, \leq)$  be the set of natural numbers, under the usual ordering.  $(\mathbb{N}, \leq)$  is a lattice, since it is a chain and thus  $n \vee m = \max\{n, m\}$  and  $n \wedge m = \min\{n, m\}$ , for all  $n, m \in \mathbb{N}$ . It is not a complete lattice though, since for any  $A \subseteq \mathbb{N}$  that is not upper bounded,  $\bigvee_{\mathbb{N}} A$  does not exist.

On the other hand, let  $(\mathbf{P}(A), \subseteq)$  is a complete lattice for any finite set  $A$ . Indeed, for any  $B \subseteq \mathbf{P}(A)$ , it holds that:

$$\bigvee_{\mathbf{P}(A)} B = \bigcup_{X \in B} X,$$

$$\bigwedge_{\mathbf{P}(A)} B = \bigcap_{X \in B} X.$$

Finally, the ordered set  $(\mathbf{P}(A) \setminus \{\emptyset\}, \subseteq)$ , where  $A$  is any finite set with at least two distinct elements, is not a lattice, since any two distinct singletons  $\{a\}$  and  $\{b\}$ ,  $a, b \in A$  have no meet.

As is well known, the structures we describe above reflect the way clones (and co-clones) over a finite set are related.

**Proposition 4.1.1.**  $(\mathcal{L}_{\mathcal{D}}^o, \subseteq)$  and  $(\mathcal{L}_{\mathcal{D}}^r, \subseteq)$  are complete lattices.

*Proof.* This proof uses only tools from set theory and, consequently, there is no essential difference between using clones or co-clones. Thus, we provide a proof for  $\mathcal{L}_{\mathcal{D}}^o$ . The one for  $\mathcal{L}_{\mathcal{D}}^r$  follows along the same lines.

Let  $\mathcal{B} \subseteq \mathcal{L}_{\mathcal{D}}^o$  be a set of clones on  $\mathcal{D}$ . We have that:

$$\bigcup_{F \in \mathcal{B}} F \text{ and } \bigcap_{F \in \mathcal{B}} F$$

are, by definition, the smallest set containing every element of  $\mathcal{B}$  and the largest set contained in every element of  $\mathcal{B}$  respectively. First observe that the join of  $\mathcal{B}$  is an upper bound for  $\mathcal{B}$ , thus, by the above:

$$\bigcup_{F \in \mathcal{B}} F \subseteq \bigvee_{(\mathcal{L}_{\mathcal{D}}^o, \subseteq)} \mathcal{B}.$$

By the definition of the clone generated by a set and since the join of  $\mathcal{B}$  is a clone, we have that:

$$\left[ \bigcup_{F \in \mathcal{B}} F \right] \subseteq \bigvee_{(\mathcal{L}_{\mathcal{D}}^{\circ}, \subseteq)} \mathcal{B}.$$

The opposite inclusion is obtained by recalling that the join of  $\mathcal{B}$  is the least upper bound for  $\mathcal{B}$ . Thus:

$$\left[ \bigcup_{F \in \mathcal{B}} F \right] = \bigvee_{(\mathcal{L}_{\mathcal{D}}^{\circ}, \subseteq)} \mathcal{B}.$$

By definition, the meet of  $\mathcal{B}$  in  $\mathcal{L}_{\mathcal{D}}^{\circ}$  is the largest clone that is contained in every  $F \in \mathcal{B}$ . It follows that:

$$\bigwedge_{(\mathcal{L}_{\mathcal{D}}^{\circ}, \subseteq)} \mathcal{B} \subseteq \bigcap_{F \in \mathcal{B}} F. \quad (4.23)$$

We now show that the right-hand side of Eq. [4.23](#) is a clone, which implies the opposite inclusion and thus, that:

$$\bigwedge_{(\mathcal{L}_{\mathcal{D}}^{\circ}, \subseteq)} \mathcal{B} = \bigcap_{F \in \mathcal{B}} F. \quad (4.24)$$

This is fairly straightforward. It holds that:

$$\bigcap_{F \in \mathcal{B}} F \subseteq F, \text{ for all } F \in \mathcal{B}.$$

Thus,

$$\left[ \bigcap_{F \in \mathcal{B}} F \right] \subseteq [F] = F, \text{ for all } F \in \mathcal{B},$$

since every  $F \in \mathcal{B}$  is a clone. Consequently:

$$\left[ \bigcap_{F \in \mathcal{B}} F \right] \subseteq \bigcap_{F \in \mathcal{B}} F$$

and, since the opposite inclusion always holds, we have that the two sets are equal and that the conjunction of the clones in  $\mathcal{B}$  is a clone. Thus, we proved Eq. [4.24](#).  $\square$



Note that although we in fact proved that the intersection of clones is always a clone, we did not show the equivalent result for the union of clones. In Subsec. [4.1.5](#), where we present  $\mathcal{L}_{\mathcal{D}}^o$  in case  $\mathcal{D}$  has only two distinct elements, we show that this is not always true.

Being a complete lattice is a strong structural property. Nevertheless, we are interested to go into as much detail as possible. Obtaining a complete description of  $\mathcal{L}_{\mathcal{D}}^o$ , for an arbitrary domain  $\mathcal{D}$  seems to be too ambitious. Even for a domain  $\mathcal{D}$  with cardinality 3,  $\mathcal{L}_{\mathcal{D}}^o$  turns out to be *uncountably infinite* and with an extremely complicated structure. In Subsec. [4.1.5](#), we shall see a complete description of  $\mathcal{L}_{\mathcal{D}}^o$  in the case where  $|\mathcal{D}| = 2$ , made by Post [\[186\]](#). Let us also mention that the proof of Prop. [4.1.1](#) holds also in case  $\mathcal{D}$  is infinite. This exceeds our scope though, so we refer the interested reader to Goldstern and Pinsker's work [\[108\]](#). Finally, in Subsec. [4.1.4](#), we shall see how to translate properties of  $\mathcal{L}_{\mathcal{D}}^o$  to corresponding ones of  $\mathcal{L}_{\mathcal{D}}^r$ .

Failing a complete description of  $\mathcal{L}_{\mathcal{D}}^o$  for a domain  $\mathcal{D}$  of arbitrary cardinality, we turn to other interesting aspects of its structure, namely its maximal and minimal clones. These classes of clones have been studied, since they are the largest and smallest non-trivial clones respectively. The results here are again partial.

Ivo Rosenberg [\[188\]](#) has characterized all maximal clones for any finite domain  $\mathcal{D}$ . The tools needed to present this result will be discussed in Subsec. [4.1.4](#). For minimal clones, we have complete descriptions by Post [\[186\]](#) and Csákány [\[62\]](#), for domains of cardinality 2 and 3 respectively. Finally, Rosenberg has provided a necessary condition for a clone  $F$  to be minimal, which we state here without proving.

**Theorem 4.1.1** (Rosenberg [\[189\]](#)). *Let  $\mathcal{D}$  be a finite domain and  $F$  be a minimal clone in  $\mathcal{L}_{\mathcal{D}}^o$ . Then,  $F = [f]$ , where  $f \in \mathcal{O}_{\mathcal{D}} \setminus \mathcal{J}_{\mathcal{D}}$  such that  $f$  is:*

- i. a unary operation and is either a retraction, i.e.  $f^2 = f$  or a permutation of prime order, or*
- ii. a binary idempotent operation, or*
- iii. a ternary majority operation, or*
- iv. a ternary minority operation, or*
- v. a  $k$ -ary non-trivial semi-projection.*

Some of the classes of operators in Theorem [4.1.1](#) appear many times in our results in the field of Aggregation Theory.

#### 4.1.4 Polymorphisms and the Galois Connection

The main tool that connects clones and co-clones, is the notion of preservation.

**Definition 4.1.6.** *Let  $R \in \mathcal{R}_{\mathcal{D}}$  be an  $n$ -ary relation,  $n \in \mathbb{N}^*$ , and  $f \in \mathcal{O}_{\mathcal{D}}$  a  $k$ -ary operation. We say that  $f$  preserves  $R$ , or that  $f$  is a polymorphism of  $R$  and write  $f \triangleright R$ , if for all  $\mathbf{a}^1, \dots, \mathbf{a}^k \in R$ :*

$$f(\mathbf{a}^1, \dots, \mathbf{a}^k) := (f(\mathbf{a}_1), \dots, f(\mathbf{a}_n)) \in R,$$

where  $f(\mathbf{a}_j) = f(a_j^1, \dots, a_j^k)$ ,  $j = 1, \dots, n$ . Furthermore, we say that  $f$  preserves a set of relations  $\mathcal{R}$ , if  $f \triangleright R$  for all  $R \in \mathcal{R}$  and that  $R$  is preserved by a set of operations  $B$ , if  $f \triangleright R$ , for all  $f \in B$ .

The main objects we study in this subsection, are sets of operations and relations such that the former preserve the latter.

**Definition 4.1.7.** *Let  $B$  be a set of operators and  $\mathcal{R}$  a set of relations. We denote the set of polymorphisms of  $\mathcal{R}$  by  $\text{Pol}(\mathcal{R})$  and the set of relations preserved by  $B$  as  $\text{Inv}(B)$ :*

$$\text{Pol}(\mathcal{R}) := \{f \in \mathcal{O}_{\mathcal{D}} \mid f \triangleright R, \forall R \in \mathcal{R}\}, \quad (4.25)$$

$$\text{Inv}(B) := \{R \in \mathcal{R}_{\mathcal{D}} \mid f \triangleright R, \forall f \in B\}. \quad (4.26)$$

If  $B$  or  $\mathcal{R}$  are singletons, we omit the brackets and write  $\text{Pol}(R)$  or  $\text{Inv}(f)$ , instead of  $\text{Pol}(\{R\})$  or  $\text{Inv}(\{f\})$  respectively.

The next two lemmas show that  $\text{Pol}(\mathcal{R})$  and  $\text{Inv}(B)$  have familiar structures.

**Lemma 4.1.3.** *Given any set of relations  $\mathcal{R}$ ,  $\text{Pol}(\mathcal{R})$  is a clone.*

*Proof.* Let  $R \in \mathcal{R}$  be an arbitrary  $n$ -ary relation and  $\mathbf{a}^1, \dots, \mathbf{a}^k \in R$ . Let also  $\text{pr}_d^k$  be a  $k$ -ary projection, where  $d \in \{1, \dots, k\}$ . Easily,  $\text{pr}_d^k(\mathbf{a}^1, \dots, \mathbf{a}^k) = \mathbf{a}^d \in R$  and thus, since  $k$ ,  $d$  and  $R$  are arbitrary,  $\mathcal{J}_{\mathcal{D}} \subseteq \text{Pol}(\mathcal{R})$ .

Now, let  $f, g_1, \dots, g_m \in \text{Pol}(\mathcal{R})$ , where  $f$  is  $m$ -ary and  $g_1, \dots, g_m$  are  $k$ -ary operators on  $\mathcal{D}$  and consider their superposition  $h := f(g_1, \dots, g_m)$ . It holds that:

$$h(\mathbf{a}^1, \dots, \mathbf{a}^k) = (f(g_1(\mathbf{a}_1), \dots, g_m(\mathbf{a}_1)), \dots, f(g_1(\mathbf{a}_n), \dots, g_m(\mathbf{a}_n))).$$

We show that  $h(\mathbf{a}^1, \dots, \mathbf{a}^k) \in R$ . Indeed, let:

$$(g_l(\mathbf{a}_1), \dots, g_l(\mathbf{a}_n)) := \mathbf{b}^l, \quad l = 1, \dots, m.$$

It follows that  $(\mathbf{b}^1, \dots, \mathbf{b}^m)$  is again an  $m \times k$  matrix with elements from  $R$  since  $g_1, \dots, g_m$  preserve  $R$ . Also:

$$h(\mathbf{a}^1, \dots, \mathbf{a}^k) = f(\mathbf{b}^1, \dots, \mathbf{b}^m) \in R,$$

again since  $f \triangleright R$ . Thus,  $\text{Pol}(\mathcal{R})$  is a clone.  $\square$

**Lemma 4.1.4.** *Given any set of functions  $B$ ,  $\text{Inv}(B)$  is a co-clone.*

*Proof.* In all that follows, let  $f \in B$  be an arbitrary  $k$ -ary operator. We prove each property of Definition [4.1.3](#) separately. To make notation easier to follow, assume that for  $\mathbf{a}^1, \dots, \mathbf{a}^k \in \mathcal{D}^k$ :

$$f(a^1, \dots, a^k) = (f(\mathbf{a}_1), \dots, f(\mathbf{a}_n)) = (b_1, \dots, b_n) = \mathbf{b}.$$

Let  $I \subseteq \{1, \dots, n\}^2$  and  $\mathbf{a}^1, \dots, \mathbf{a}^k \in E_I^n$ . Then,

$$a_s^i = a_t^i, \quad \forall s, t \in I, i = 1, \dots, k$$

and thus, obviously,  $b_s = b_t$ , for all  $s, t \in I$ . It follows that  $\mathbf{b} \in E_I^n$  and that  $f \triangleright E_I^n$ .

Assume now that  $R = P \times Q$ , where  $P$  is  $p$ -ary,  $Q$  is  $q$ -ary (and thus  $n = p + q$ ) and  $P, Q \in \text{Inv}(B)$ . Since  $f \triangleright P$  and  $f \triangleright Q$ :

$$\begin{aligned} (b_1, \dots, b_s) &\in P, \\ (b_{s+1}, \dots, b_{s+t}) &\in Q \end{aligned}$$

and thus  $\mathbf{b} \in P \times Q = R$ . Thus  $f \triangleright R$ .

Let  $Q_i \in \text{Inv}(B)$ ,  $i \in I$ , be  $n$ -ary relations, such that:

$$R = \bigcap_{i \in I} Q_i.$$

It follows that  $\mathbf{a}^1, \dots, \mathbf{a}^k \in Q_i$  and thus  $\mathbf{b} \in Q_i$  too, for all  $i \in I$ . Thus:

$$\mathbf{b} \in \bigcap_{i \in I} Q_i = R$$

and  $f \triangleright R$ .

Let  $P \in \text{Inv}(B)$  be an  $n$ -ary relation such that  $R \approx P$ . Then, there are vectors  $\mathbf{c}^1, \dots, \mathbf{c}^k \in P$  such that  $\mathbf{a}^i \approx \mathbf{c}^i$ ,  $i = 1, \dots, k$ . Then:

$$f(\mathbf{c}^1, \dots, \mathbf{c}^k) = \mathbf{d} \in P,$$

where  $\mathbf{d} \approx \mathbf{b}$ . Thus,  $\mathbf{b} \in R$  and  $f \triangleright R$ .

Let  $P \in \text{Inv}(B)$  be an  $m$ -ary relation,  $m < n$  such that there exists an  $I \subsetneq \{1, \dots, n\}$ :  $R = P_I$ . Then, there are vectors  $\mathbf{c}^1, \dots, \mathbf{c}^k \in P$  such that  $\mathbf{a}^i = \mathbf{c}^i$ ,  $i = 1, \dots, k$ . Then:

$$f(\mathbf{c}^1, \dots, \mathbf{c}^k) = \mathbf{d} \in P,$$

where  $\mathbf{b} = \mathbf{d}_I$ . Thus,  $\mathbf{b} \in R$  and  $f \triangleright R$ . □

Lemmas [4.1.3](#) and [4.1.4](#) indirectly imply that in terms of preservation, using a set of functions or the clone it generates (and accordingly with clones) has no difference. Although the result is of technical nature, we feel that it is somewhat in accordance with intuition.

**Corollary 4.1.2.** *For any set  $F \subseteq \mathcal{O}_{\mathcal{D}}$  of operations and any set  $\mathcal{R} \subseteq \mathcal{R}_{\mathcal{D}}$  of relations over a finite domain  $\mathcal{D}$ , it holds that:*

1.  $\text{Pol}(\langle \mathcal{R} \rangle) = \text{Pol}(\mathcal{R})$  and
2.  $\text{Inv}([B]) = \text{Inv}(B)$ .

*Proof.* First note that, since for any  $F \subseteq \mathcal{O}_{\mathcal{D}}$  and any  $\mathcal{R} \subseteq \mathcal{R}_{\mathcal{D}}$ ,  $F \subseteq [F]$  and  $\mathcal{R} \subseteq \langle \mathcal{R} \rangle$ , we immediately have that:

$$\text{Pol}(\langle \mathcal{R} \rangle) \subseteq \text{Pol}(\mathcal{R}), \tag{4.27}$$

$$\text{Inv}([F]) \subseteq \text{Inv}(F). \tag{4.28}$$

Now, for the opposite inclusion, we first need to show that any  $f \in \text{Pol}(\mathcal{R})$  is also in  $\text{Pol}(\langle \mathcal{R} \rangle)$ . Thus, we need to show that any function preserving  $\mathcal{R}$ , also preserves  $\langle \mathcal{R} \rangle$ . But this is exactly what we did in the proof of Lemma [4.1.4](#). Analogously, to show that  $\text{Inv}([B]) \subseteq \text{Inv}(B)$ , we need to show that any relation preserved by  $B$ , is also preserved by  $[B]$ , which is what we did in Lemma [4.1.3](#). □

The connection between the complete lattices  $\mathcal{L}_{\mathcal{D}}^o$  and  $\mathcal{L}_{\mathcal{D}}^r$  can now be established using a well known and powerful algebraic tool.

**Definition 4.1.8.** *Let  $(P, \sqsubseteq_P)$ ,  $(Q, \sqsubseteq_Q)$  be two ordered sets and  $f : P \mapsto Q$ ,  $g : Q \mapsto P$  two functions between these two sets.  $f$  and  $g$  form a Galois correspondence or Galois connection if the following hold:*

- (i) *If  $a \sqsubseteq_P b$  then  $f(b) \sqsubseteq_Q f(a)$ , for all  $a, b \in P$ ,*
- (ii) *If  $c \sqsubseteq_Q d$  then  $g(d) \sqsubseteq_P g(c)$ , for all  $c, d \in Q$  and*
- (iii)  *$a \sqsubseteq_P g(f(a))$  and  $b \sqsubseteq_Q f(g(b))$ , for all  $a \in P$  and  $b \in Q$ .*

A closure operator of an ordered set  $(P, \sqsubseteq)$  is a function  $f : P \mapsto P$  such that:

- (i)  $a \sqsubseteq f(a)$ , for all  $a \in P$ ,
  - (ii) if  $a \sqsubseteq b$  then  $f(a) \sqsubseteq f(b)$ , for all  $a, b \in P$ ,
  - (iii)  $f(a) = f(f(a))$ , for all  $a \in P$ .
- (4.29)

A bijection  $h : L \mapsto L'$  between to lattices  $(L, \sqsubseteq_L)$  and  $(L', \sqsubseteq_{L'})$  such that:

$$a \sqsubseteq_L b \Rightarrow h(b) \sqsubseteq_{L'} h(a), \quad \forall a, b \in L, \quad (4.30)$$

is called a *lattice anti-isomorphism*.

In our case, we see Pol and Inv as functions between  $\mathcal{L}_{\mathcal{D}}^r$  and  $\mathcal{L}_{\mathcal{D}}^o$ .

**Proposition 4.1.2.** *Let  $(\mathcal{L}_{\mathcal{D}}^o, \subseteq)$  and  $(\mathcal{L}_{\mathcal{D}}^r, \subseteq)$  be ordered sets under inclusion. The functions  $\text{Pol} : \mathcal{L}_{\mathcal{D}}^r \mapsto \mathcal{L}_{\mathcal{D}}^o$  and  $\text{Inv} : \mathcal{L}_{\mathcal{D}}^o \mapsto \mathcal{L}_{\mathcal{D}}^r$  form a Galois correspondence.*

*Proof.* By Def. 4.1.8, we need to prove three conditions. For (i), let  $\mathcal{R}, \mathcal{T} \in \mathcal{L}_{\mathcal{D}}^r$  such that  $\mathcal{R} \subseteq \mathcal{T}$  and assume  $f \in \text{Pol}(\mathcal{T})$ . Since  $f$  preserves every relation in  $\mathcal{T}$ , it also holds that  $f \triangleright R$ , for all  $R \in \mathcal{R}$ . Thus,  $f \in \text{Pol}(\mathcal{R})$  and  $\text{Pol}(\mathcal{T}) \subseteq \text{Pol}(\mathcal{R})$ . Item (ii) can be proven in the same way.

For (iii), let  $\mathcal{R} \in \mathcal{L}_{\mathcal{D}}^r$  and assume  $R \in \mathcal{R}$ . Since  $R$  is preserved by every operation in  $\text{Pol}(\mathcal{R})$ , it holds by definition that  $R \in \text{Inv}(\text{Pol}(\mathcal{R}))$  and thus  $\mathcal{R} \subseteq \text{Inv}(\text{Pol}(\mathcal{R}))$ . Analogously, we can show that  $F \subseteq \text{Pol}(\text{Inv}(F))$ , for all  $F \in \mathcal{L}_{\mathcal{D}}^o$ . □

We consider now the functions  $\text{Pol}(\text{Inv}) : \mathcal{L}_{\mathcal{D}}^o \mapsto \mathcal{L}_{\mathcal{D}}^o$  and  $\text{Inv}(\text{Pol}) : \mathcal{L}_{\mathcal{D}}^r \mapsto \mathcal{L}_{\mathcal{D}}^r$  and prove what is known as *Geiger's Theorem*. To that end, the lectures of Jin-Yi Cai et al. [44, 45] will be useful. Before proceeding we need to do some work. A partial  $k$ -ary operator  $f$  preserves an  $n$ -ary relation  $R$  if, for all  $\mathbf{a}^1, \dots, \mathbf{a}^k \in R$ :

$$\text{If } \mathbf{a}_1, \dots, \mathbf{a}_n \in \text{Dom}(f), \text{ then } (f(\mathbf{a}_1), \dots, f(\mathbf{a}_n)) \in R. \quad (4.31)$$

Thus, a partial  $k$ -ary function  $f$  preserves an  $n$ -ary relation  $R$ , if it preserves all  $k \times n$  matrices of vectors from  $R$  such that  $f$  is defined on each of their columns.

Now, let  $f$  be a  $k$ -ary operation and  $R$  an  $n$ -ary relation. Let also  $A \subseteq R$  be a set of  $k$  vectors from  $R$ , that we view as a  $k \times n$  matrix. Taking the transpose  $n \times k$  matrix  $A^T$  of  $A$ , we write  $f(A^T)$  to denote the column vector obtained by applying  $f$  to each row of  $A$ . Thus, under that notation,  $f$  is a polymorphism of  $R$  if and only if  $f(A^T)^T \in R$ , for all  $A \subseteq R$  such that  $|A| = k$ .

**Lemma 4.1.5.** *Let  $\mathcal{R}$  be a co-clone. Then, any partial polymorphism  $f$  of  $\mathcal{R}$  can be extended to a polymorphism of  $\mathcal{R}$ .*

*Proof.* Let  $f$  be a  $k$ -ary partial polymorphism of  $\mathcal{R}$ . Recall that by Lemma 4.1.3,  $\mathcal{J}_{\mathcal{D}} \subseteq \text{Pol}(\mathcal{R})$  and thus  $\text{Pol}(\mathcal{R}) \neq \mathcal{R}$ . We can assume that  $\emptyset \subsetneq \text{Dom}(f) \subsetneq \mathcal{D}^k$ , lest we have nothing to prove, since if  $f$  has an empty domain, it is extended by every  $g \in \text{Pol}(\mathcal{R})$  and, if it is defined on every vector of  $\mathcal{D}^k$ , it is a polymorphism of  $\mathcal{R}$ .

Now, assume that  $\text{Dom}(f) := \{\mathbf{a}^1, \dots, \mathbf{a}^m\}$  and  $\mathbf{a} \notin \text{Dom}(f)$ . Let also  $\mathcal{D} := \{d_1, \dots, d_s\}$  and

$$f_t(\mathbf{b}) = \begin{cases} d_t & \text{if } \mathbf{b} = \mathbf{a}, \\ f(\mathbf{b}) & \text{if } \mathbf{b} \in \text{Dom}(f), \end{cases}$$

for  $t = 1, \dots, s$ . We show that at least one of  $f_1, \dots, f_s$  is a partial polymorphism of  $\mathcal{R}$ . The result then follows by induction.

To obtain a contradiction, assume that  $f_t$  is not a polymorphism of  $\mathcal{R}$ , for  $t = 1, \dots, s$ . Thus, there exist (not necessarily distinct)  $n_t$ -ary relations  $R_t \in \mathcal{R}$ , such that  $f_t$  does not preserve  $R_t$ ,  $t = 1, \dots, s$ . Furthermore, there exist  $k \times n_t$  matrices  $M_t$ , comprised of vectors from  $R_t$ , such that  $f_t$  is defined on every column of  $M_t$  and  $f_t(M_t^T)^T \notin R_t$ ,  $t = 1, \dots, s$ . Also,  $M_t^T \subseteq \text{Dom}(f_t) = \{\mathbf{a}^1, \dots, \mathbf{a}^m, \mathbf{a}\}$ ,  $t = 1, \dots, s$ .

We need to take care of a technical detail here. For each  $t \in \{1, \dots, s\}$ , we can assume that there is no repeated column in  $M_t$ . Indeed assume there are two distinct  $p, q \in \{1, \dots, n_t\}$ , such that  $\mathbf{c}_p = \mathbf{c}_q$ , where  $\mathbf{c}_1, \dots, \mathbf{c}_{n_t}$  are the columns of  $M_t$ . Let  $R'_t$  contain the vectors of  $R_t$  that identify at indices  $p$  and  $q$ , that is:

$$R'_t := R_t \cap E_{p,q}^{n_t}$$

and  $R''_t$  be the projection of  $R'_t$  to  $\{1, \dots, q-1, q+1, \dots, n_t\}$ :

$$R''_t = \{(c_1, \dots, c_{q-1}, c_{q+1}, \dots, c_{n_t}) \mid \mathbf{c} \in R'_t\}.$$

Since  $\mathcal{R}$  is a co-clone, both  $R'_t$  and  $R''_t$  are in  $\mathcal{R}$ . Easily now, if  $N_t$  is  $M_t$  without  $\mathbf{c}_q$ ,  $N_t \subseteq R''_t$  and, furthermore, since  $f_t(M_t^T)^T \notin R_t$ , then  $f_t(N_t^T)^T \notin R''_t$ . Thus, we can use  $R''_t$  instead of  $R_t$ .

Let now:

$$R := \prod_{t=1}^s R_t.$$

Since  $\mathcal{R}$  is a co-clone and thus closed under Cartesian products, it holds that  $R \in \mathcal{R}$ . Furthermore, if  $M$  is the  $k \times n$  matrix, where  $n = n_1 + \dots + n_s$ , obtained by putting the columns of each  $M_t$  one after the other, it holds that  $M \subseteq R$ .

**Claim 4.1.1.**  *$M$  contains  $\mathbf{a}$  as a column exactly  $s$  times and at least one other column  $\mathbf{c} \in \{\mathbf{a}^1, \dots, \mathbf{a}^m\}$ .*

*Proof of Claim [4.1.1](#)* Assume that there is some  $t \in \{1, \dots, s\}$  such that  $\mathbf{a}$  is not a column of  $M_t$ . Then  $f(M_t^T)^T = f_t(M_t^T)^T \notin R_t$  and thus  $f$  is not a partial polymorphism of  $\mathcal{R}$ . Contradiction. That each  $M_t$  contains  $\mathbf{a}$  as a column exactly once is implied by the fact that columns are not repeated.

Now assume that there is no other column in  $M$ . Then,  $M_t = \mathbf{a}^T$ ,  $t = 1, \dots, s$ . Now, let  $r \in \{1, \dots, s\}$  such that the first element of  $\mathbf{a}$ ,  $a_1 = d_r$ . Then,  $f_r(M_r) = f_r(\mathbf{a}) = d_r \in R_r$ , since  $M_r \subseteq R_r$ . Contradiction.  $\square$

Again, let  $R' := R \cap E_I^n$ , where  $I = \{i \in n \mid \mathbf{c}_i = \mathbf{a}^T\}$  and let  $R'' = R'_I$ . Arguing as above,  $R'' \in \mathcal{R}$ . Let also  $N$  be  $M$  without the columns  $\mathbf{c}_i$ ,  $i \in I$ . Obviously,  $\emptyset \subsetneq N \subseteq R''$ . Also,  $N \subseteq \text{Dom}(f)$ , thus  $f(N^T)^T \in R''$ . It follows that for some  $t \in \{1, \dots, s\}$ ,  $f_t(N^T)^T \in R'$  and thus  $f_t(M_t^T)^T \in R_t$ . Contradiction.  $\square$

We are now ready to prove the following result.

**Theorem 4.1.2** (Geiger [29, 102, 136]). *For any set  $F \subseteq \mathcal{O}_{\mathcal{D}}$  of operations and any set  $\mathcal{R} \subseteq \mathcal{R}_{\mathcal{D}}$  of relations over a finite domain  $\mathcal{D}$ , it holds that:*

1.  $\text{Pol}(\text{Inv}(F)) = [F]$  and
2.  $\text{Inv}(\text{Pol}(\mathcal{R})) = \langle \mathcal{R} \rangle$ .

*Proof.* First, by Corollary 4.1.2, we can instead show that:

$$\begin{aligned} \text{Pol}(\text{Inv}([F])) &= [F], \\ \text{Inv}(\text{Pol}(\langle \mathcal{R} \rangle)) &= \langle \mathcal{R} \rangle. \end{aligned}$$

Equivalently, it suffices to show that for any clone  $F$  and any co-clone  $\mathcal{R}$  it holds that:

$$\text{Pol}(\text{Inv}(F)) = F, \tag{4.32}$$

$$\text{Inv}(\text{Pol}(\mathcal{R})) = \mathcal{R}. \tag{4.33}$$

We prove Eq. 4.32 and 4.33 separately.

Let  $f \in F$ . By the definitions of  $\text{Inv}$  and  $\text{Pol}$ ,  $f$  preserves every relation in  $\text{Inv}(F)$  and is thus also in  $\text{Pol}(\text{Inv}(F))$ . Thus,  $F \subseteq \text{Pol}(\text{Inv}(F))$ . For the opposite inclusion, we use contraposition.

Let  $f \notin F$  be a  $k$ -ary operation. We show there exists a relation  $R \in \text{Inv}(F)$  such that  $f$  does not preserve it. First, we order arbitrarily all the vectors of  $\mathcal{D}^k$ . Let  $A$  be the resulting  $|\mathcal{D}|^k \times k$  matrix. For each  $k$ -ary  $g \in F$ , let  $g(A)$  be the column obtained by applying  $g$  to each line of  $A$ . Let  $n := |\mathcal{D}|^k$  and  $m := |\{g \in F \mid g \text{ is } k\text{-ary}\}|$  and let  $B = \{b_j^i\}_m^n$  be the  $n \times m$  matrix obtained by taking all these column vectors together. Note that since  $F$  is a clone, all the projections are included in  $F$  and thus  $A$  is a sub-matrix of  $B$ .

Let  $R = \{\mathbf{b}_j^T \mid j = 1, \dots, m\}$ , that is  $R$  is the  $n$ -ary relation whose vectors are the columns of  $B$ . We first show that  $R \in \text{Inv}(F)$ . Let  $h$  be an  $l$ -ary operation in  $F$  and  $\mathbf{c}^1, \dots, \mathbf{c}^l \in R$ . Then, there exist  $k$ -ary functions  $g_1, \dots, g_l \in F$  such that  $\mathbf{c}^i = g_i(A)$ ,  $i = 1, \dots, l$ . Thus:

$$\begin{aligned} h(\mathbf{c}^1, \dots, \mathbf{c}^l) &= (h(\mathbf{c}_1), \dots, h(\mathbf{c}_n)) \\ &= (h(g_1(\mathbf{a}^1), \dots, g_l(\mathbf{a}^1)), \dots, h(g_1(\mathbf{a}^n), \dots, g_l(\mathbf{a}^n))). \end{aligned}$$

Thus,  $h(\mathbf{c}^1, \dots, \mathbf{c}^l)$  is the superposition of  $h$  and  $g_1, \dots, g_l$ . Since  $F$  is a clone,  $h(g_1, \dots, g_l) \in F$ . But then, since  $h(g_1, \dots, g_l)$  is  $k$ -ary, there is a column



of  $B$ , and a vector in  $R$ , that is equal to  $h(g_1, \dots, g_l)(\mathbf{a}^1, \dots, \mathbf{a}^k)$ . Thus  $R \in \text{Inv}(F)$ .

Finally, it remains to show that  $f$  does not preserve  $R$ . Assume it does. Then  $f(A) \in R$ . Thus, there is a  $k$ -ary  $g \in F$  such that  $f(A) = g(A)$ . But, since  $A$  contains all the  $k$ -ary vectors on  $\mathcal{D}$ , it follows that  $f = g$ . Contradiction, since  $f \notin F$ . This, concludes the proof of Eq. [4.32](#).

Let  $R \in \mathcal{R}$ . By the definitions of  $\text{Pol}$  and  $\text{Inv}$ ,  $R$  is preserved by every operation in  $\text{Pol}(\mathcal{R})$  and is thus also in  $\text{Inv}(\text{Pol}(\mathcal{R}))$ . Thus,  $\mathcal{R} \subseteq \text{Inv}(\text{Pol}(\mathcal{R}))$ . For the opposite inclusion, we use contraposition.

Let  $R$  be an  $n$ -ary relation not in  $\mathcal{R}$ . We show that there is an operation  $f \in \text{Pol}(\mathcal{R})$  such that  $R$  is not preserved by  $f$ . We begin by defining the following  $n$ -ary relation:

$$Q := \bigcap_{P \in \mathcal{R}: P \supseteq R} P.$$

As we proved in Cor. [4.1.1](#),  $\mathcal{D}^n \in \mathcal{R}$ . Also, by definition,  $\mathcal{D}^n \supseteq R$ . Thus,  $Q$  is not the empty relation. Furthermore, since we take the intersection of relations in  $\mathcal{R}$  and  $\mathcal{R}$  is a co-clone,  $Q \in \mathcal{R}$ . Thus, there exists some  $t \in Q \setminus R$ .

Assume that  $|R| = k$ . By considering  $R$  as a  $k \times n$  matrix, we show that there exists a  $k$ -ary  $f$  that is a partial polymorphism of  $\mathcal{R}$  such that  $f(R^T) = t^T$ . Thus  $f$  does not preserve  $R$  and, by Lemma [4.1.5](#),  $f$  extends to a polymorphism of  $\mathcal{R}$  that does not preserve  $R$ . Interestingly, we show that a partial  $f$ , defined only on the columns of  $R$ , such that  $f(R^T) = t^T$ , will do.

First of all, assume that the columns  $\mathbf{a}_1, \dots, \mathbf{a}_n$  of  $R$  are pairwise distinct. Then, set  $f(a_1^i, \dots, a_k^i) := t_i$ ,  $i = 1, \dots, n$ . Then,  $f$  is a partial function, defined on the columns of  $R$ , such that  $f(R^T) = t^T$ . On the other hand, assume that there are  $p, q \in \{1, \dots, n\}$  such that  $\mathbf{a}^p = \mathbf{a}^q$ . Then,  $R \subseteq E_{p,q}^n$ . By the definition of  $Q$ ,  $Q \subseteq E_{p,q}^n$  too. Thus  $t_p = t_q$  and we can define  $f$  as above. It remains to show that  $f$  is a partial polymorphism of  $\mathcal{R}$ .

To obtain a contradiction, assume that it is not. Thus, there is an  $m$ -ary  $R' \in \mathcal{R}$  and a  $k \times m$  matrix  $M \subseteq R'$ , such that  $f(M^T)^T \notin R'$ . By assuming that  $R'$  is of minimal arity with respect to this property, we can also assume that the columns of  $M$  are pairwise distinct. Indeed, if they are not, we can take  $R' \wedge E_{p,q}^m$ , where  $p, q \in \{1, \dots, m\}$  are such that the columns  $\mathbf{b}_p, \mathbf{b}_q$  of  $R'$  are equal and project to  $\{1, \dots, m\} \setminus \{q\}$ . Thus, we have a relation  $R'' \in \mathcal{R}$  (since  $\mathcal{R}$  is a co-clone, whose arity is strictly smaller than that of  $R'$ , and such that it is not preserved by  $f$ . Contradiction.

Now, since  $f$  is defined only on the columns of  $R$ , it holds that the columns

or  $M$  are amongst those of  $R$ . Without loss of generality, assume they are the  $m$  first such columns. Now, observe that  $R \subseteq R' \times \mathcal{D}^{n-m}$ , which again implies that  $Q \subseteq R' \times \mathcal{D}^{n-m}$  too. But this shows that  $t \in R' \times \mathcal{D}^{n-m}$ , which in turn implies that  $f(M^T)^T = (t_1, \dots, t_m) \in R'$ . Contradiction, since  $f(M^T)^T \notin R'$ .  $\square$

As stated, Theorem 4.1.2 works in case  $\mathcal{D}$  is finite. For infinite domains, it takes the form presented e.g. in [138]. We can now easily obtain the following two results. Note that these can be obtained without using Theorem 4.1.2 and are also true in case  $\mathcal{D}$  is infinite.

**Corollary 4.1.3.** *Pol and Inv are lattice anti-isomorphisms for  $\mathcal{L}_{\mathcal{D}}^r$  and  $\mathcal{L}_{\mathcal{D}}^o$ .*

*Proof.* In proposition 4.1.2, we have shown that both Pol and Inv reverse the order of  $\mathcal{L}_{\mathcal{D}}^o$  and  $\mathcal{L}_{\mathcal{D}}^r$ . Thus, it suffices to show that they are bijections.

Let  $\mathcal{R}, \mathcal{T} \in \mathcal{L}_{\mathcal{D}}^r$  such that  $\text{Pol}(\mathcal{R}) = \text{Pol}(\mathcal{T})$ . Then, we necessarily have that  $\text{Inv}(\text{Pol}(\mathcal{R})) = \text{Inv}(\text{Pol}(\mathcal{T}))$ , which, by Theorem 4.1.2, gives us that  $\mathcal{R} = \langle \mathcal{R} \rangle = \langle \mathcal{T} \rangle = \mathcal{T}$ . Thus Pol is 1 – 1.

Let  $F \in \mathcal{L}_{\mathcal{D}}^o$ . Then,  $\text{Inv}(F)$  is a co-clone and is thus in  $\mathcal{L}_{\mathcal{D}}^r$ . By Theorem 4.1.2:

$$\text{Pol}(\text{Inv}(F)) = [F] = F$$

and thus Pol is onto. The analogous arguments show that Inv is also a bijection.  $\square$

**Corollary 4.1.4.** *Pol(Inv) :  $\mathcal{L}_{\mathcal{D}}^o \mapsto \mathcal{L}_{\mathcal{D}}^o$  and Inv(Pol) :  $\mathcal{L}_{\mathcal{D}}^r \mapsto \mathcal{L}_{\mathcal{D}}^r$  are closure operations for  $\mathcal{L}_{\mathcal{D}}^o$  and  $\mathcal{L}_{\mathcal{D}}^r$  respectively.*

*Proof.* That  $F \subseteq \text{Pol}(\text{Inv}(F))$  and  $\mathcal{R} \subseteq \text{Inv}(\text{Pol}(\mathcal{R}))$ , for all  $F \in \mathcal{L}_{\mathcal{D}}^o$  and  $\mathcal{R} \in \mathcal{L}_{\mathcal{D}}^r$ , follows immediately by Theorem 4.1.2, since  $F = [F]$  and  $\mathcal{R} = \langle \mathcal{R} \rangle$ . Now, let  $F, G \in \mathcal{L}_{\mathcal{D}}^o$  such that  $F \subseteq G$ . Again, by Theorem 4.1.2,  $\text{Pol}(\text{Inv}(F)) = [F] = F \subseteq G = [G] = \text{Pol}(\text{Inv}(G))$ . Analogously for  $\text{Inv}(\text{Pol})$ . Finally, for any  $F \in \mathcal{L}_{\mathcal{D}}^o$  and  $\mathcal{R} \in \mathcal{L}_{\mathcal{D}}^r$ :

$$\begin{aligned} \text{Pol}(\text{Inv}(\text{Pol}(\text{Inv}(F)))) &= [\text{Pol}(\text{Inv}(F))] = [[F]] = F, \\ \text{Inv}(\text{Pol}(\text{Inv}(\text{Pol}(\mathcal{R})))) &= \langle \text{Inv}(\text{Pol}(\mathcal{R})) \rangle = \langle \langle \mathcal{R} \rangle \rangle = \mathcal{R}, \end{aligned}$$

again by Theorem 4.1.2.  $\square$

### 4.1.5 The Boolean Case: Post's Lattice

We present here the seminal result of Post, namely the complete description of the lattices of clones and co-clones over Boolean domains. Apart from being one of the most important results in the field of Universal Algebra, it plays a critical part in our work in the field of Judgement Aggregation. In the brief presentation we make here, we follow the approach of Böhler et al. [30, 31], which is also where Fig. 4.1, 4.2 and 4.3 come from. For an alternative approach, see Zverovich [128] and Pelletier [181] and for the original proof, see Post [186].

Fig. 4.1 contains short descriptions of the clones of Boolean operators. Below, we provide all the necessary notation to understand this table, that is not included in Sec. 4.1.1. Fig. 4.2 contains the lattice of Boolean clones. If  $F$  is below some  $F'$  and connected with it, then  $F \subseteq F'$ . Fig. 4.3 contains the lattice of Boolean co-clones. Notionally, the co-clone containing the relations preserved by the operations in clone  $F$ , i.e.  $\text{Inv}(F)$ , is denoted by  $IF$ . The *base* of a Boolean clone  $B$  is the minimal set of operations  $F$ , with respect to inclusion, such that  $[F] = B$ .

Let again  $f : \{0, 1\}^k \mapsto \{0, 1\}$ . For an  $a \in \{0, 1\}$ ,  $f$  is *a-reproducing*, if  $f(a, \dots, a) = a$ . Observe that  $f$  is idempotent if and only if it is *a-reproducing* for all  $a \in \{0, 1\}$ .

$f$  is *self-dual*, if for all  $(a_1, \dots, a_k) \in \{0, 1\}^k$ :

$$f(a_1, \dots, a_k) = \neg f(\bar{a}_1, \dots, \bar{a}_k),$$

where  $\bar{0} = 1$  and  $\bar{1} = 0$ .

For any  $a \in \{0, 1\}$ , a subset  $A \subseteq \{0, 1\}^k$  is *a-separating* there exists some  $j \in \{1, \dots, k\}$  such that for all  $(a_1, \dots, a_k) \in A$ ,  $a_j = a$ .  $f$  is *a-separating* if the inverse  $f^{-1}(a)$  is *a-separating*. It is *a-separating of level  $l$* , if every  $A \subseteq f^{-1}(a)$  such that  $|A| = l$  is *a-separating*. Furthermore, for each  $a \in \{0, 1\}$ ,  $f$  is constant and equal to  $a$ , denoted by  $c_a^k$ , if  $f(a_1, \dots, a_k) = a$  for all  $(a_1, \dots, a_k) \in \{0, 1\}^k$ .

For  $k = 0$ , we have two cases for  $f$ :  $c_0 := 0$  and  $c_1 := 1$ . For  $k = 1$ , again  $f$  can take two forms:  $\text{id}(a) := a$  and  $\text{not}(a) = \neg a$ , for all  $a \in \{0, 1\}$ . For  $k = 2$ , and  $= \wedge$ , or  $= \vee$ ,  $\text{xor} = \oplus$ ,  $\text{eq}(a, b) = 1$  if and only if  $a = b$ ,  $\text{imp}(a, b) = 0$  if and only if  $a = 1$  and  $b = 0$  and  $\text{nand}(a, b) = \neg(\wedge(a, b))$ .

Class	Definition	Base(s)
BF	all Boolean functions	$\{and, not\}$
R <sub>0</sub>	$\{f \in BF \mid f \text{ is 0-reproducing}\}$	$\{and, xor\}$
R <sub>1</sub>	$\{f \in BF \mid f \text{ is 1-reproducing}\}$	$\{or, x \oplus y \oplus 1\}$
R <sub>2</sub>	$R_1 \cap R_0$	$\{or, x \wedge (y \oplus z \oplus 1)\}$
M	$\{f \in BF \mid f \text{ is monotonic}\}$	$\{and, or, c_0, c_1\}$
M <sub>1</sub>	$M \cap R_1$	$\{and, or, c_1\}$
M <sub>0</sub>	$M \cap R_0$	$\{and, or, c_0\}$
M <sub>2</sub>	$M \cap R_2$	$\{and, or\}$
S <sub>0</sub> <sup>n</sup>	$\{f \in BF \mid f \text{ is 0-separating of degree } n\}$	$\{imp, dual(h_n)\}$
S <sub>0</sub>	$\{f \in BF \mid f \text{ is 0-separating}\}$	$\{imp\}$
S <sub>1</sub> <sup>n</sup>	$\{f \in BF \mid f \text{ is 1-separating of degree } n\}$	$\{x \wedge \bar{y}, h_n\}$
S <sub>1</sub>	$\{f \in BF \mid f \text{ is 1-separating}\}$	$\{x \wedge \bar{y}\}$
S <sub>02</sub> <sup>n</sup>	$S_0^n \cap R_2$	$\{x \vee (y \wedge \bar{z}), dual(h_n)\}$
S <sub>02</sub>	$S_0 \cap R_2$	$\{x \vee (y \wedge \bar{z})\}$
S <sub>01</sub> <sup>n</sup>	$S_0^n \cap M$	$\{dual(h_n), c_1\}$
S <sub>01</sub>	$S_0 \cap M$	$\{x \vee (y \wedge z), c_1\}$
S <sub>00</sub> <sup>n</sup>	$S_0^n \cap R_2 \cap M$	$\{x \vee (y \wedge z), dual(h_n)\}$
S <sub>00</sub>	$S_0 \cap R_2 \cap M$	$\{x \vee (y \wedge z)\}$
S <sub>12</sub> <sup>n</sup>	$S_1^n \cap R_2$	$\{x \wedge (y \vee \bar{z}), h_n\}$
S <sub>12</sub>	$S_1 \cap R_2$	$\{x \wedge (y \vee \bar{z})\}$
S <sub>11</sub> <sup>n</sup>	$S_1^n \cap M$	$\{h_n, c_0\}$
S <sub>11</sub>	$S_1 \cap M$	$\{x \wedge (y \vee z), c_0\}$
S <sub>10</sub> <sup>n</sup>	$S_1^n \cap R_2 \cap M$	$\{x \wedge (y \vee z), h_n\}$
S <sub>10</sub>	$S_1 \cap R_2 \cap M$	$\{x \wedge (y \vee z)\}$
D	$\{f \mid f \text{ is self-dual}\}$	$\{x\bar{y} \vee x\bar{z} \vee \bar{y}z\}$
D <sub>1</sub>	$D \cap R_2$	$\{xy \vee x\bar{z} \vee y\bar{z}\}$
D <sub>2</sub>	$D \cap M$	$\{xy \vee yz \vee xz\}$
L	$\{f \mid f \text{ is linear}\}$	$\{xor, c_1\}$
L <sub>0</sub>	$L \cap R_0$	$\{xor\}$
L <sub>1</sub>	$L \cap R_1$	$\{eq\}$
L <sub>2</sub>	$L \cap R_2$	$\{x \oplus y \oplus z\}$
L <sub>3</sub>	$L \cap D$	$\{x \oplus y \oplus z \oplus c_1\}$
V	$\{f \mid f \text{ is an } n\text{-ary } or\text{-function or a constant function}\}$	$\{or, c_0, c_1\}$
V <sub>0</sub>	$\{or\} \cup \{c_0\}$	$\{or, c_0\}$
V <sub>1</sub>	$\{or\} \cup \{c_1\}$	$\{or, c_1\}$
V <sub>2</sub>	$\{or\}$	$\{or\}$
E	$\{f \mid f \text{ is an } n\text{-ary } and\text{-function or a constant function}\}$	$\{and, c_0, c_1\}$
E <sub>0</sub>	$\{and\} \cup \{c_0\}$	$\{and, c_0\}$
E <sub>1</sub>	$\{and\} \cup \{c_1\}$	$\{and, c_1\}$
E <sub>2</sub>	$\{and\}$	$\{and\}$
N	$\{not\} \cup \{c_0\} \cup \{c_1\}$	$\{not, c_1\}, \{not, c_0\}$
N <sub>2</sub>	$\{not\}$	$\{not\}$
I	$\{id\} \cup \{c_1\} \cup \{c_0\}$	$\{id, c_0, c_1\}$
I <sub>0</sub>	$\{id\} \cup \{c_0\}$	$\{id, c_0\}$
I <sub>1</sub>	$\{id\} \cup \{c_1\}$	$\{id, c_1\}$
I <sub>2</sub>	$\{id\}$	$\{id\}$

Figure 4.1: Classes of Boolean operators

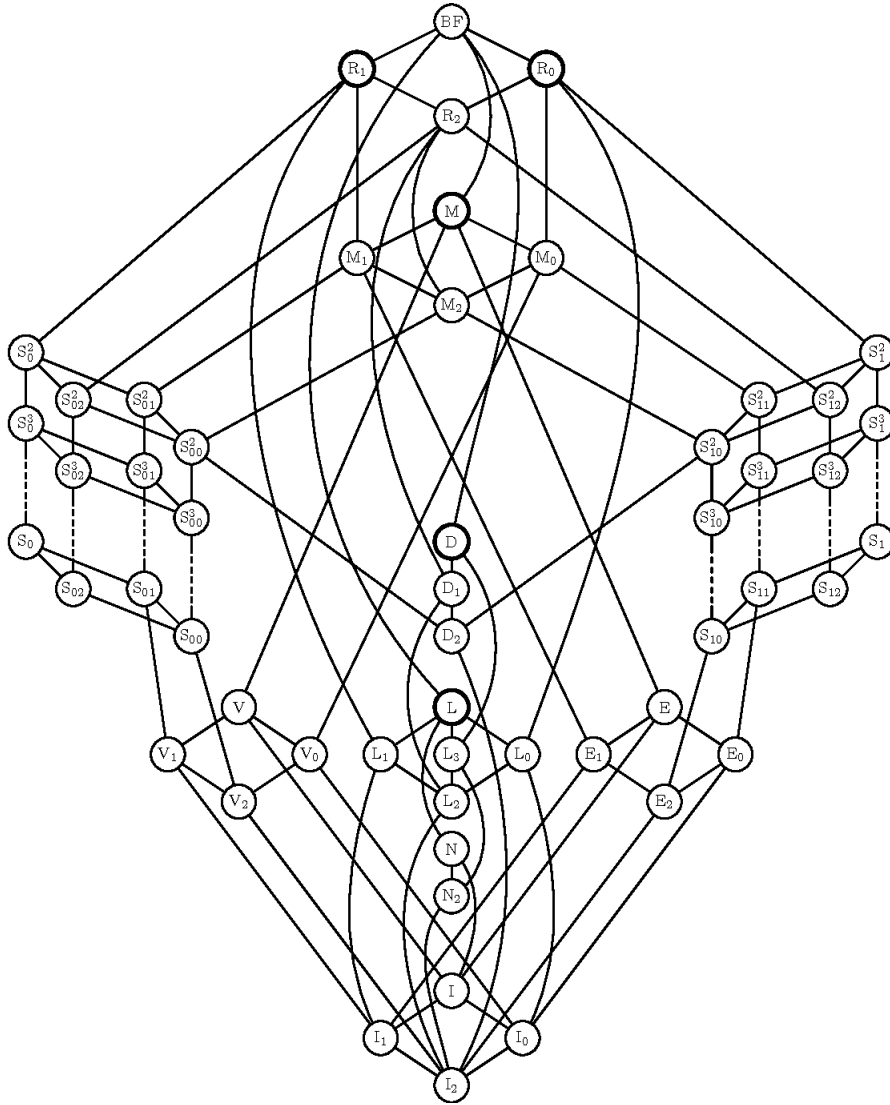


Figure 4.2: Lattice of Boolean clones

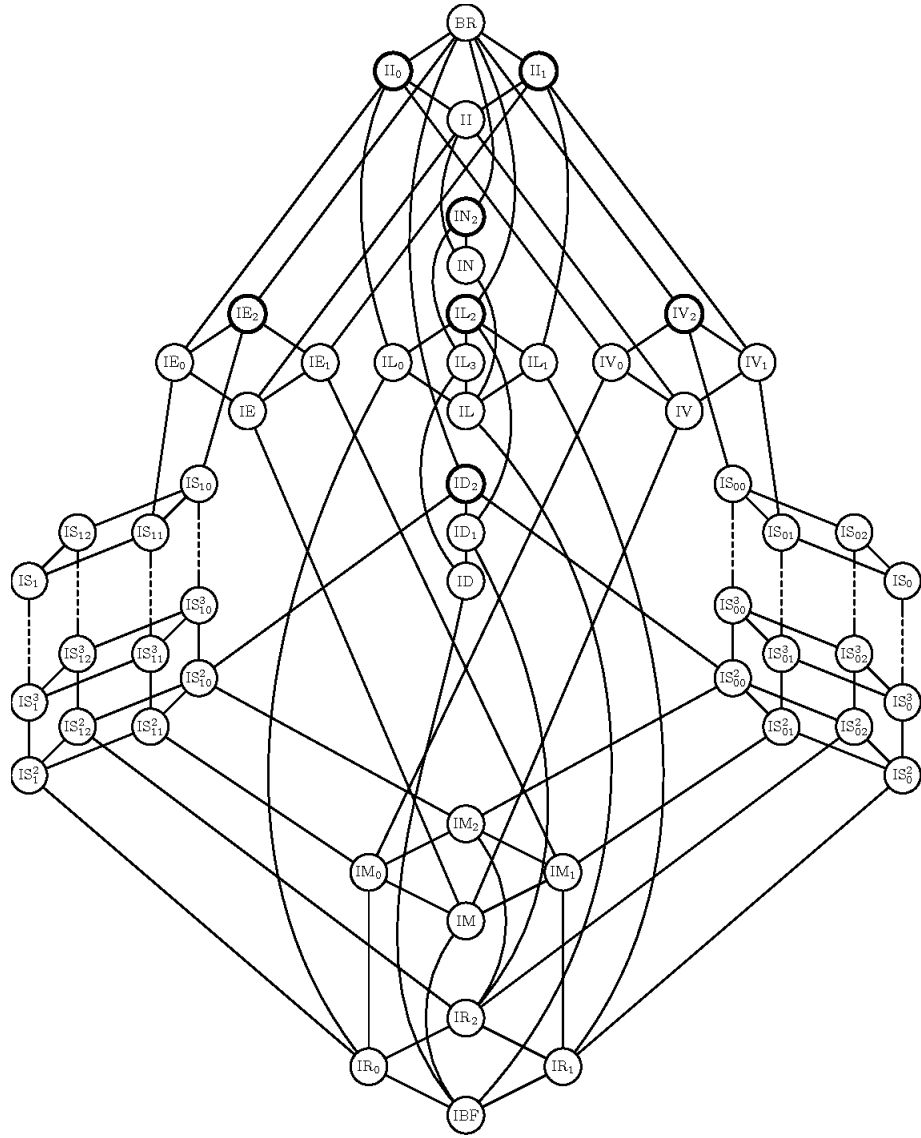


Figure 4.3: Lattice of Boolean co-clones

Observe that  $\{\text{id}\}$  is in fact the clone of all projections. Finally:

$$E := \{f \text{ } k\text{-ary}, k \in \mathbb{N} \mid f(a_1, \dots, a_k) = c_0 \wedge (c_1 \vee a_1) \wedge \dots \wedge (c_k \vee a_k)\},$$

$$V := \{f \text{ } k\text{-ary}, k \in \mathbb{N} \mid f(a_1, \dots, a_k) = c_0 \vee (c_1 \wedge a_1) \vee \dots \vee (c_k \wedge a_k)\},$$

where  $c_0, c_1, \dots, c_k$  are constants in  $\{0, 1\}$ .

## 4.2 Computational Complexity of CSP's

In this section we provide an overview of complexity theoretic results, both in the Boolean and non-Boolean framework. We begin, in Subsec. [4.2.1](#) with an alternative proof of Schaefer's dichotomy theorem, through Post's lattice. In Subsec. [4.2.2](#) we consider the Feder-Vardi conjecture and various results obtained for CSP's with non-Boolean domains, including Bulatov's Dichotomy Theorems for conservative and multi-sorted CSP's. Finally, in Subsec. [4.2.3](#) we consider the meta-questions in CSP's, including Carbonnel's Theorem that one can efficiently discern tractable from non-tractable CSP's.

### 4.2.1 Schaefer's Dichotomy Theorem through Post's Lattice

Schaefer's Dichotomy Theorem for the Boolean satisfiability problem has been originally proven in [\[192\]](#). Here we follow the approach presented in Bohler et al. [\[30, 31\]](#) using Post's lattice [\[186\]](#). We employ an analogous strategy in Ch. [7](#) in order to obtain many of our results there.

Our work in this subsection is situated in the Boolean framework, thus  $\mathcal{D}$  can be represented by  $\{0, 1\}$ . Assume  $\mathcal{R}, \mathcal{S} \in \mathcal{R}_{\mathcal{D}}$  are sets of Boolean relations, such that  $\mathcal{R} \subseteq \mathcal{S}$ . Easily, any instance  $I$  of  $\text{CSP}(\mathcal{R})$  is also an instance of  $\text{CSP}(\mathcal{S})$ , since all the constraints that can be formed by relations of  $\mathcal{R}$ , can also be formed by relations in  $\mathcal{S}$ . Thus, we immediately obtain the following result.

**Corollary 4.2.1.** *Let  $\mathcal{R}, \mathcal{S} \in \mathcal{R}_{\mathcal{D}}$  be sets of Boolean relations. If  $\mathcal{R} \subseteq \mathcal{S}$ , then  $\text{CSP}(\mathcal{R})$  is polynomial-time reducible to  $\text{CSP}(\mathcal{S})$ . Specifically,  $\text{CSP}(\mathcal{R})$  is polynomial-time reducible to  $\text{CSP}(\langle \mathcal{R} \rangle)$ , for all  $\mathcal{R} \subseteq \mathcal{R}_{\mathcal{D}}$ .*

It turns out that  $\text{CSP}(\mathcal{R})$  and  $\text{CSP}(\langle \mathcal{R} \rangle)$  are polynomial-time equivalent. For this, we need some more work.

**Lemma 4.2.1.** *For any set of relation  $\mathcal{R} \subseteq \mathcal{R}_{\mathcal{D}}$ ,  $\text{CSP}(\mathcal{R})$  and  $\text{CSP}(\langle \mathcal{R} \rangle)$  are polynomial-time equivalent.*

*Proof.* By Cor. 4.2.1, it suffices to show that:

$$\text{CSP}(\langle \mathcal{R} \rangle) \leq_P \text{CSP}(\mathcal{R}), \text{ for all } \mathcal{R} \subseteq \mathcal{R}_{\mathcal{D}}.$$

Recall the definition of CSP, in Fig. 2.1. Given an instance  $I$  of  $\text{CSP}(\langle \mathcal{R} \rangle)$ , we show how to translate it to an instance of  $\text{CSP}(\mathcal{R})$  in polynomial time.

To make the notation easier to follow, we assume that the variables used in this instance are  $x_1, \dots, x_m$  and we denote the variables in the scopes of the various constraints by  $z_1, \dots, z_n \in \{x_1, \dots, x_m\}$ , where the  $z_i$ 's need not be distinct. Now, if the instance has only constraints with relations from  $\mathcal{R}$ , we have nothing to prove. Thus below, we assume that all the constraints we discuss are not in  $\mathcal{R}$ .

Suppose there is some constraint  $C = (E_I^n, (z_1, \dots, z_n))$ . Let also  $I = \{i_1, \dots, i_k\}$  and let  $z_{i_1} = x_j$ ,  $j \in \{1, \dots, m\}$ . To any constraint  $C' = (R, \text{sc}(R))$ , replace any  $x_k \in \text{sc}(R)$  such that  $x_k = z_{i_l}$ , for some  $l \in \{1, \dots, k\}$ , with  $x_j$ . Then delete  $C$ . Easily this procedure is polynomial (in fact linear) and the produced instance is equivalent to  $I$ .

Assume now there is a constraint  $C = (R, (z_1, \dots, z_n))$ , and relations  $Q, P \in \mathcal{R}$  of arities  $q$  and  $p$  respectively, such that  $R = Q \times P$  (and thus  $q + p = n$ ). Replace the constraint  $C$  with  $C' = (Q, (z_1, \dots, z_q))$  and  $C'' = (P, (z_{q+1}, \dots, z_{q+p}))$ . Again, this procedure is linear and the produced instance is equivalent to  $I$ .

Let  $C = (R, (z_1, \dots, z_n))$  be a constraint of  $I$ , and  $\mathcal{S} \subseteq \mathcal{R}$  such that:

$$R = \bigcap_{Q \in \mathcal{S}} Q.$$

Every relation in  $\mathcal{S}$  must thus have arity  $n$ . Replace the constraint  $C$  with  $C_Q = (Q, (z_1, \dots, z_n))$ , for all  $Q \in \mathcal{S}$ . As before, this procedure is linear and the produced instance is equivalent to  $I$ .

If  $C = (R, (z_1, \dots, z_n))$  is a constraint, were  $R \approx Q$ , for some  $Q \in \mathcal{R}$ , replace it with  $C' = (Q, (z_{i_1}, \dots, z_{i_n}))$ , where  $\{i_1, \dots, i_n\}$  is the permutation of the columns of  $R$  needed to obtain  $Q$ . As above, this procedure is linear and the produced instance is equivalent to  $I$ .

Finally, let  $C = (R, (z_1, \dots, z_n))$  be a constraint, where  $R = Q_I$ , for some  $k$ -ary  $Q \in \mathcal{R}$ , where  $k \geq n$  and  $I \subseteq \{1, \dots, k\}$ . Without loss of generality, let



$I = \{1, \dots, n\}$  and set  $J := \{n + 1, \dots, k\}$ . Finally, let  $y_1, \dots, y_{k-n}$  be variables not in  $\{x_1, \dots, x_m\}$ . Replace  $C$  with  $C' = (Q, (z_1, \dots, z_n, y_1, \dots, y_{k-n}))$ . Again, this procedure is linear and the produced instance is equivalent to  $I$ .

The result now follows by repeatedly performing the above procedures until all the constraints contain relations from  $\mathcal{R}$ .  $\square$

We can now prove the main reduction we need for the proof of Schaefer's Theorem.

**Theorem 4.2.1** (Jeavons [130]). *Let  $\mathcal{R}, \mathcal{S} \in \mathcal{R}_{\mathcal{D}}$ . If  $\text{Pol}(\mathcal{S}) \subseteq \text{Pol}(\mathcal{R})$ , then  $\text{CSP}(\mathcal{R}) \leq_P \text{CSP}(\mathcal{S})$ .*

*Proof.* Since  $\text{Pol}(\mathcal{S}) \subseteq \text{Pol}(\mathcal{R})$ , by Prop. 4.1.2 we have that:

$$\text{Inv}(\text{Pol}(\mathcal{R})) \subseteq \text{Inv}(\text{Pol}(\mathcal{S}))$$

. Thus, by Th. 4.1.2, we have that  $\langle \mathcal{R} \rangle \subseteq \langle \mathcal{S} \rangle$ , which in turn, by Cor. 4.2.1, implies that  $\text{CSP}(\langle \mathcal{R} \rangle) \leq_P \text{CSP}(\langle \mathcal{S} \rangle)$ . The result now follows since, by Lemma 4.2.1,  $\text{CSP}(\langle \mathcal{R} \rangle)$  is polynomially equivalent to  $\text{CSP}(\mathcal{R})$  as are  $\text{CSP}(\langle \mathcal{S} \rangle)$  and  $\text{CSP}(\mathcal{S})$ .  $\square$

We consider now some special types of Boolean relations. Let  $R \subseteq \{0, 1\}^n$  be a Boolean relation. We say it is *0-valid* (resp. *1-valid*) if:

$$\underbrace{(0, 0, \dots, 0)}_{n\text{-times}} \in R \text{ (resp. } \underbrace{(1, 1, \dots, 1)}_{n\text{-times}} \in R).$$

$R$  is *Horn* (resp. *dual-Horn*) if there is a Horn (resp. dual-Horn) formula  $\phi$  such that  $\text{Mod}(\phi) = R$ . It is *bijunctive* if there is a 2-SAT formula  $\phi$  such that  $\text{Mod}(\phi) = R$  and *affine* if there is an affine  $\phi$  such that  $\text{Mod}(\phi) = R$ .

A set of relations  $\mathcal{R}$  is 0-valid (resp. 1-valid, Horn, dual-Horn, bijunctive, affine) if every  $R \in \mathcal{R}$  is 0-valid (resp. 1-valid, Horn, dual-Horn, bijunctive, affine).  $\mathcal{R}$  is *Schaefer*, if it is Horn, dual-Horn, bijunctive or affine.

There are efficient ways to recognize if a relation  $R$  is of any of the above types. For 0/1-validity, one simply has to check if the corresponding vectors are in  $R$ .  $R$  is Horn (resp. dual-Horn) if and only if  $\wedge \in \text{Pol}(R)$  ( $\vee \in \text{Pol}(R)$ ) (see [59, 68]). It is bijunctive if and only if  $\text{maj} \in \text{Pol}(R)$  (see [59, 192]) and affine if and only if  $\oplus \in \text{Pol}(R)$  (see [58, 59, 192]). If one is interested in obtaining the corresponding formula describing  $R$ , there is a standard method that can be found e.g. in Enderton [80]. There is also a very interesting approach of Zanuttini and Hébrard [219] that we will also use in Ch. 7.

We are now ready to prove Schaefer's Dichotomy Theorem for CSP's in the Boolean framework.

**Theorem 4.2.2** (Schaefer [192]). *Let  $\mathcal{R} \subseteq \mathcal{R}_{\mathcal{D}}$  be a set of Boolean relations. If  $\mathcal{R}$  is 0-valid or 1-valid or Schaefer,  $\text{CSP}(\mathcal{R}) \in \text{P}$ . Otherwise,  $\text{CSP}(\mathcal{R})$  is NP-complete.*

*Proof.* Consider  $\mathcal{L}_{\mathcal{D}}^r$  of Fig. 4.3. The *maximal* co-clones, that is the largest co-clones that are not equal to  $\text{BR} = \mathcal{R}_{\mathcal{D}}$ , denoted by bold circles, are  $II_0, II_1, IE_2, IV_2, ID_2, IL_2$  and  $IN_2$ . We show that

$$\text{if } \mathcal{R} \in \{II_0, II_1, IE_2, IV_2, ID_2, IL_2\} \text{ then, } \text{CSP}(\mathcal{R}) \in \text{P}$$

and that if  $\mathcal{R} \supseteq IN_2$ , then  $\text{CSP}(\mathcal{R})$  is NP-complete. In light of Theorem 4.2.1, the proof will be completed, since for every  $\mathcal{R} \in \mathcal{R}_{\mathcal{D}}$ , it holds either that  $\mathcal{R}$  is a subset of the first six co-clones, or a superset of  $IN_2$ .

If  $\mathcal{R} = II_0$  (resp.  $\mathcal{R} = II_1$ ), then  $\text{Pol}(\mathcal{R}) = I_0$  (resp.  $\text{Pol}(\mathcal{R}) = I_1$ ). Thus, every constraint in  $\text{CSP}(\mathcal{R})$  is satisfied by setting every variable equal to 0 (resp. 1).

If  $\mathcal{R} = IE_2$  (resp.  $IV_2, ID_2, IL_2$ ) then  $\text{Pol}(\mathcal{R}) = E_2$  (resp.  $V_2, D_2, L_2$ ). Thus,  $S$  is Horn (resp. dual-Horn, bijunctive, affine). There are known algorithms that solve  $\text{CSP}(\mathcal{R})$  in each of those cases, that can be found e.g. in Papadimitriou [177].

Assume now that  $\mathcal{R} = IN_2$ . Then  $\text{Pol}(\mathcal{R}) = N_2 = [\{\neg\}]$ . It follows that  $R_{NAE} \in \mathcal{R}$ , where:

$$R_{NAE} := \{0, 1\}^3 \setminus \{(0, 0, 0), (1, 1, 1)\}.$$

$\text{CSP}(R_{NAE})$  is in fact the NOT-ALL-EQUAL-SAT problem, that is the problem where, given a 3-SAT formula  $\phi$ , we want to find if there is a satisfying assignment of  $\phi$  such that the variables of each clause do not receive the same value. It is known that this is an NP-complete problem (again, see e.g. [177]) and thus the proof is finished.  $\square$

Another form that Th. 4.2.2 can take is the following. We say that a  $k$ -ary  $f$  on  $\{0, 1\}$  is *essentially unary*, if there is an  $i \in \{1, \dots, k\}$  such that

$$f(a_1, \dots, a_k) = \text{pr}_i^k(a_1, \dots, a_k) \text{ or } f(a_1, \dots, a_k) = \neg \text{pr}_i^k(a_1, \dots, a_k).$$

One can express that also as  $f$  being equal to the unary identity  $\text{id}$  or negation  $\neg$  operation on its  $i$ -th variable. It can be shown that if  $\mathcal{R} \supseteq IN_2$ , then  $\text{Pol}(\mathcal{R})$  contains only essentially unary functions. Thus, we have the following result.

**Corollary 4.2.2** (Jeavons et al. [131,134]). *Let  $\mathcal{R} \subseteq \mathcal{R}_{\mathcal{D}}$  be a set of Boolean relations. If  $\text{Pol}(\mathcal{R})$  contains only essentially unary operations, then  $\text{CSP}(\mathcal{R})$  is NP-complete. Otherwise,  $\text{CSP}(\mathcal{R}) \in \text{P}$ .*

## 4.2.2 Dichotomy Theorems

Given Schaefer's Dichotomy result, it is natural to inquire what happens if the CSP is defined over a non-Boolean domain. This was explicitly stated in what is now known as the "Feder-Vardi conjecture" [90]. The authors there give several indications as to why such a dichotomy might hold in the non-Boolean case too.

The main class Feder and Vardi consider is **SNP**, as defined in Subsection 2.4. Interestingly, for every problem  $A \in \text{NP}$ , there exists a problem  $B \in \text{SNP}$  such that  $A \leq_P B$  and  $B \leq_P A$ . This means that the existence of NP-intermediate problems would directly translate to NP-intermediate problems in the class **SNP**.

Feder and Vardi then consider three syntactical restrictions of **SNP** problems, namely *monotonicity*, *monadicity* and *no inequality* (again, see [90]). They show that imposing any two out of this three restrictions, results in a subclass of **SNP** that continues to have polynomially equivalent problems to every problem in **NP**. Nevertheless, by imposing all three restrictions, we obtain the monotone monadic SNP without inequalities (**MMSNP**). They show that **CSP** is strictly contained in **MMSNP** and, furthermore, that every problem in **MMSNP** is polynomially equivalent to a problem in **CSP** (although the reduction from the **CSP** problem to the **MMSNP** one is randomized).

Feder and Vardi's conjecture has inspired various works in the attempt to obtain a dichotomy result for CSP's in the non-Boolean framework. We make a very brief overview here of such attempts. First, let us note that attempts in discerning tractable classes of CSP problems using various notions of *consistency*, both predate and follow Feder and Vardi's conjecture (see e.g. [63,67,132]). This approach far exceeds the scope of this thesis. Another such approach is by Bulatov [34,39,41], that uses *finite algebras* and *groupoids* to obtain results concerning the complexity of CSP's. It should be noted though that Bulatov has provided a dichotomy theorem in case  $\mathcal{D}$  contains 3 elements [36].

Using an approach similar to what we have discussed and general algebraic results concerning (minimal) clones [52,167,189,208], Jeavons and Cohen [133] showed a set of relations  $\mathcal{R}$  over a finite domain  $\mathcal{D}$  can have

specific types of polymorphisms.

We say that  $\mathcal{R}$  is *reduced*, if every unary operation  $f \in \text{Pol}(\mathcal{R})$  is 1-1. Note that if this is not the case, then  $\text{CSP}(\mathcal{R})$  can be defined in a smaller domain  $\mathcal{D}'$  (see [133] for details on this issue). Thus, from now on, we will assume that  $\mathcal{R}$  is reduced, for any  $\mathcal{R} \subseteq \mathcal{R}_{\mathcal{D}}$ . In the non-Boolean framework, a  $k$ -ary  $f \in \mathcal{O}_{\mathcal{D}}$  is *essentially unary* if there is a non-constant unary  $g \in \mathcal{O}_{\mathcal{D}}$  such that:

$$f(a_1, \dots, a_k) := g(a_i), \text{ for some } i \in \{1, \dots, k\}.$$

**Theorem 4.2.3.** *Let  $\mathcal{R} \subseteq \mathcal{R}_{\mathcal{D}}$  be a reduced set of relations over a finite domain  $\mathcal{D}$ . Then, either  $\text{Pol}(\mathcal{R})$  contains essentially unary operations only or there is an  $f \in \text{Pol}(\mathcal{R})$  such that  $f$  is:*

- either a constant operation,
- or a majority operation,
- or a binary idempotent operation which is not a projection,
- or an affine operation,
- or a semi-projection.

The proof of Th. 4.2.3 can again be found in [133].

Now using Schaefer's Dichotomy Theorem (Th. 4.2.2) as a base and proceeding inductively, they proved that:

**Theorem 4.2.4** (Cohen and Jeavons [133]). *For any finite set  $\mathcal{R} \subseteq \mathcal{R}_{\mathcal{D}}$  over a finite domain  $\mathcal{D}$ , if  $\text{Pol}(\mathcal{R})$  contains only essentially unary operations, then  $\text{CSP}(\mathcal{R})$  is NP-complete.*

Furthermore, they proceeded with proving the following sufficient conditions for tractability.

**Theorem 4.2.5.** *For any finite set  $\mathcal{R} \subseteq \mathcal{R}_{\mathcal{D}}$  over a finite domain  $\mathcal{D}$ , if  $\text{Pol}(\mathcal{R})$  contains an operations  $f$  such that  $f$  is:*

- either constant,
- or a majority operation,
- or an affine operation,

then  $\text{CSP}(\mathcal{R})$  is tractable.

Then, they proceed to show that closure under a binary idempotent non-projection operation or a semi-projection is not a sufficient condition for tractability. This, along with the lack of a complete description for  $\mathcal{L}_{\mathcal{D}}^r$  and  $\mathcal{L}_{\mathcal{D}}^o$  in case  $\mathcal{D}$  is not Boolean (see Sec. 4.1), leaves the sought after dichotomy theorem an open problem until today.

We finally discuss two Dichotomy Theorems by Bulatov [38, 40] for subclasses of the CSP's: the *conservative* CSP (c-CSP) and the conservative multi-sorted CSP (c-MCSP). We use these theorems in Ch. 7, to obtain our results.

A set of relations  $\mathcal{R}$  is *conservative*, if every  $B \subseteq \mathcal{D}$  is a relation in  $\mathcal{R}$ . Bulatov showed a Dichotomy Theorem for conservative CSP's over domains of arbitrary cardinalities.

**Theorem 4.2.6** (Bulatov [38]). *If for every Boolean  $B \subseteq \mathcal{D}$  there is a binary polymorphism  $f$  of  $\mathcal{R}$  such that  $f \upharpoonright B \in \{\wedge, \vee\}$  or a ternary polymorphism  $f$  of  $\mathcal{R}$  such that  $f \upharpoonright B \in \{\text{maj}, \oplus\}$ , then  $\text{c-CSP}(\mathcal{R})$  is solvable in polynomial time; otherwise it is NP-complete.*

Recall now the setting of Subsec. 2.1.1 concerning MCSP. Again, let  $\mathfrak{D} = \{\mathcal{D}_i \mid i \in I\}$  be an arbitrary collection of finite sets. A multi-sorted constraint language  $\Gamma$  over  $\mathfrak{D}$  is called *conservative* if for all sets  $\mathcal{D}_i \in \mathfrak{D}$  and all subsets  $B \subseteq \mathcal{D}_i$ , we have that  $B \in \mathcal{R}$  (as a relation over  $\mathcal{D}_i$ ). A  $k$ -ary *multi-sorted polymorphism* for  $\Gamma$  is a collection of  $k$ -ary operators  $F = (f_i)_{i \in I}$ , where  $f_i : \mathcal{D}_i^k \mapsto \mathcal{D}_i$ , such that, for every  $R \in \Gamma$  with signature  $\sigma(R) = (i_1, \dots, i_n)$  and for every  $A = \{\mathbf{a}^1, \dots, \mathbf{a}^k\} \in R^k$ , where  $\mathbf{a}^l = (a_1^l, \dots, a_n^l) \in R$ ,  $l = 1, \dots, k$ , it holds that:

$$(f_{i_1}(\mathbf{a}_1), \dots, f_{i_n}(\mathbf{a}_n)) \in R.$$

Bulatov [37, Theorem 2.16] proved a dichotomy theorem for conservative multi-sorted constraint languages.

**Theorem 4.2.7** (Bulatov [37]). *If for any  $i \in I$  and any two-element subset  $B_i \subseteq \mathcal{D}_i$  there is either a binary multi-sorted polymorphism  $F = (f_i)_{i \in I}$  of  $\Gamma$  such that  $f_i \upharpoonright B_i \in \{\wedge, \vee\}$  or a ternary multi-sorted polymorphism  $F = (f_i)_{i \in I}$  of  $\Gamma$  such that  $f_i \upharpoonright B_i \in \{\text{maj}, \oplus\}$ , then  $\text{c-MCSP}(\Gamma)$  is solvable in polynomial time; otherwise it is NP-complete.*

### 4.2.3 Meta-questions for CSP tractability

As we have seen, depending on the set of relations  $\mathcal{R}$ , the computational complexity of  $\text{CSP}(\mathcal{R})$  can vary, with the tractable and NP-complete cases drawing the most attention and the question of whether there are NP-intermediate CSP problems still open. It is natural to ask whether, given a  $\text{CSP}(\mathcal{R})$ , we can efficiently decide if it is tractable or not. Given the various conditions that have been provided in the bibliography, that either guarantee or exclude tractability (e.g. *bounded width* [15]), we can instead ask for an efficient way to check whether a  $\text{CSP}(\mathcal{R})$  satisfies these conditions or not. This type of questions are known as *meta-questions*.

For example, since we know that checking if a relation  $R$  is one of the Schaefer classes (or 0/1-valid), the meta-question concerning Schaefer's Dichotomy Theorem is tractable. Chen and Larose [51] provide a variety of such results and formalize the connection between the tractability of meta-questions with the existence of efficient *uniform* algorithms for such questions, that is algorithms that uniformly solve the meta-question for whole classes of CSP's.

In this area, we are interested in the results of Bessi re et al. [22] and Carbonnel [47, 48]. Recall that an operator  $f : \mathcal{D}^k \mapsto \mathcal{D}$  is conservative if its output is always a part of its input. The following three Theorems will be used in Ch. 7.

First, we have that deciding whether a constraint language  $\mathcal{R}$  admits a conservative majority or minority polymorphism is tractable. Recall that these are both sufficient conditions for tractability of  $\text{CSP}(\mathcal{R})$ .

**Theorem 4.2.8** (Bessi re et al. [22]). *There is a polynomial-time algorithm for the following problem: given a constraint language  $\mathcal{R}$  on a set  $\mathcal{D}$ , determine whether or not  $\mathcal{R}$  admits a conservative majority polymorphism. Moreover, if such a polymorphism exists, then the algorithm produces one in polynomial time.*

**Theorem 4.2.9** (Carbonnel [48]). *There is a polynomial-time algorithm for the following problem: given a constraint language  $\mathcal{R}$  on a set  $\mathcal{D}$ , determine whether or not  $\mathcal{R}$  admits a conservative minority polymorphism. Moreover, if such a polymorphism exists, then the algorithm produces one in polynomial time.*

Finally, Carbonnel showed that the boundary of the dichotomy for  $c\text{-CSP}(R)$  can be checked in polynomial time.

**Theorem 4.2.10** (Carbonnel [47]). *There is a polynomial-time algorithm for the following problem: given a constraint language  $\mathcal{R}$  on a set  $A$ , determine whether or not for every two-element subset  $B \subseteq A$ , there is a conservative polymorphism  $f$  of  $\mathcal{R}$  such that either  $f$  is binary and  $f \upharpoonright B \in \{\wedge, \vee\}$  or  $f$  is ternary and  $f \upharpoonright B \in \{\text{maj}, \oplus\}$ . Moreover, if such a polymorphism exists, then the algorithm produces one in polynomial time.*





# Chapter 5

## Computational Social Choice Theory

In this chapter we study concepts in the field of Social Choice Theory and, more specifically, Judgment Aggregation. The latter is the setting where our results of Ch. 7 are situated. We begin with some preliminaries in Preference and Judgment Aggregation in Sec. 5.1. In Sec. 5.2 and Sec. 5.3 we study the two main frameworks of Judgment Aggregation where our results are situated; the *abstract framework* and the *integrity constraints* respectively.

### 5.1 Social Choice Theory

In this section we provide some preliminaries in Preference Aggregation (Subsec. 5.1.1) and in Judgment aggregation (Subsec. 5.1.2). In the former framework, we present Arrow’s “General Possibility Theorem”. The latter framework is where our work is situated.

#### 5.1.1 Preference Aggregation

The origin of Social Choice Theory, at least as a robust mathematical theory, is usually traced back to J. K. Arrow’s “General Possibility Theorem” [11] (or “Arrow’s Impossibility Theorem”), that was published in 1951.

Given a population of  $\{1, \dots, k\}$ , which we refer to as *voters* or *agents*, we are interested in their *preferences* over a set of  $l$  alternatives. That is, we are interested in, weak or strict, total, anti-symmetric and transitive orderings  $\sqsubseteq$

or  $\sqsubseteq$  respectively, of the set of alternatives  $\mathcal{D}$ , where  $|\mathcal{D}| = l$ . These orderings are called *linear* and we denote the set of all possible such ordering as  $\mathcal{L}(\mathcal{D})$ . By dropping the anti-symmetry requirement, we obtain the set  $\mathcal{N}(\mathcal{D})$  of total and transitive orderings over  $\mathcal{D}$ .

A (*preference*) *profile* is a vector  $P = (\sqsubseteq_1, \dots, \sqsubseteq_k)$ , where  $\sqsubseteq_i \in \mathcal{L}(\mathcal{D})$  is the preference of agent  $i$ ,  $i = 1, \dots, k$ . To aggregate the individual preferences and obtain the social outcome, we use the so called *social welfare functions* (SWF), which are functions  $f : \mathcal{L}(\mathcal{D})^k \mapsto \mathcal{N}(\mathcal{D})$ . Note that dropping the anti-symmetry requirement in the outcome amounts to allowing ties in the social preference order. We denote the output of  $f$  over a profile by the order  $\sqsubseteq$ . If there are more than one SWF, we use  $\sqsubseteq_f$  to denote the output of each such  $f$ .

Arrow argued that a SWF to be practical as an aggregation rule, it should have four “common sense” properties. The first one is included already at the definition: a SWF should be defined over every element of  $\mathcal{L}(\mathcal{D})^k$ . This property is sometimes called *universality*.

1. A SWF  $f$  is *weakly Paretian* if it holds that: for all  $a, b \in \mathcal{D}$ , if  $a \sqsubseteq_i b$  for  $i = 1, \dots, k$ , then  $a \sqsubseteq b$ . Intuitively, this means that  $f$  should respect all unanimous preferences between two alternatives.
2. A SWF  $f$  is *independent of irrelevant alternatives (IIA)* if the relative social ranking between any two alternatives  $a, b \in \mathcal{D}$  depends only on the individual relative rankings between  $a$  and  $b$ . Formally, let  $P, P'$  be two profiles such that for some  $a, b \in \mathcal{D}$ :

$$a \sqsubseteq_i^P b \text{ if and only if } a \sqsubseteq_i^{P'} b, \text{ for } i = 1, \dots, k.$$

Then,  $a \sqsubseteq^P b$  if and only if  $a \sqsubseteq^{P'} b$ .

3. A SWF is non-dictatorial if there is no agent who determines by himself the social outcome. Formally,  $f$  is a dictatorship if there exists an  $i \in \{1, \dots, l\}$  such that:

$$\text{if } a \sqsubseteq_i b \text{ then } a \sqsubseteq b, \text{ for all } a, b \in \mathcal{D}.$$

At first glance, the majority rule, where  $a \sqsubseteq_{\text{maj}} b$  if and only if  $|\{i \mid a \sqsubseteq_i b\}| < |\{i \mid a \sqsupseteq_i b\}|$ , satisfies all the desired properties. The issue here, as it was known by the so called *Condorcet's paradox* in the late 18th century

(see [65] for a retrieved version of the original work), is that the majority rule is not even a SWF, since it does not satisfy transitivity! Indeed, assume  $\mathcal{D} := \{a, b, c\}$  and that we have three agents, where:

$$\begin{aligned} a &\sqsubseteq_1 b \sqsubseteq_1 c, \\ b &\sqsubseteq_2 c \sqsubseteq_2 a, \\ c &\sqsubseteq_3 a \sqsubseteq_3 b. \end{aligned}$$

The social outcome under the majority rule would be that  $a \sqsubseteq b$ , since agents 1 and 3 prefer  $b$  to  $a$ ,  $b \sqsubseteq c$ , since agents 1 and 2 prefer  $c$  to  $b$  and  $c \sqsubseteq a$ , since agents 2 and 3 prefer  $a$  to  $c$ .

Arrow showed that the problem is far wider.

**Theorem 5.1.1** (Arrow [11]). *For any set  $\mathcal{D}$  of at least three alternatives, any SWF that satisfies universality, IIA and is weakly Paretian, is necessarily a dictatorship.*

Arrow's result has initiated a variety of approaches in order to avoid it with practical SWF. This required for some of the properties described above to be relaxed. Universality, non-dictatorship and being weakly Paretian seem to be considered desirable in almost every scenario. Thus, usually, the condition that is relaxed is the requirement of IIA. We will not get into this issue though here, since our focus is in a slightly different framework, which we consider in the next section. For an interesting discussion of Arrow's impossibility theorem, see Tao [210] whereas for three alternative proofs of, see Geanakopulos [99].

### 5.1.2 Judgment Aggregation

To talk about Judgment Aggregation, one must almost necessarily begin by the *doctrinal paradox* of Kornhauser and Sager [152], or its most recent and abstract form, provided by Pettit and Rabinowicz [182], the *discursive dilemma*. For an interesting book containing a broad range of matters concerning Computational Social Choice Theory and Judgement Aggregation, we refer the reader to Brandt et al. [32].

Assume we have three agents (usually depicted as judges) that need to decide if three propositional formulas, namely  $p$ ,  $q$  and  $p \wedge q$ , are true or not. Assume also that the first agent thinks that both  $p$  and  $q$  are true, the second only  $p$  and the third only  $q$ . For the opinions of the agents to be consistent,

this implies that the first agent accepts  $p \wedge q$  as true, while the other two as false. How should a collective decision be made?

Our usual preference would be to vote by majority rule. But, if so, we have the following paradox. If the agents vote on the premises, then both  $p$  and  $q$  and thus  $p \wedge q$ , should be accepted. On the other hand, if they vote on the conclusion,  $p \wedge q$  should not be accepted. The simplicity of the setting that majority voting cannot handle was perhaps one of the main reasons this paradox had such a strong impact in this field.

We now present the original formal framework of judgment aggregation. Let  $\mathbb{P}$  be the set of all propositional formulas that does not contain any double negated formula. An *agenda*  $\Phi \subseteq \mathbb{P}$  is a subset that is *closed under complementation*, that is,  $\phi \in \Phi$  if and only if  $\neg\phi \in \Phi$ . We denote by  $\Phi^+$  the set of not negated (positive) formulas in  $\Phi$  and by  $\Phi^-$  the negated (negative) ones. Given an agenda  $\Phi$  such that  $|\Phi^+| = n$  (and thus  $|\Phi| = 2n$ ) and a population of  $k$  agents, each agent decides over the validity or not of the formulas of  $\Phi$ , by selecting a *judgment set*  $J \subseteq \Phi$ . In the original framework, we can assume that  $J$  is always *complement free* and *complete*, that is, for all  $\phi \in \Phi$ , exactly one of  $\phi$  or  $\neg\phi$  is in  $J$  (although, see e.g. Terzopoulou et al. [213] for results with incomplete judgment sets).  $J$  is *consistent* if there exists an assignment of values that satisfies all  $\phi \in J$ . We denote by  $\mathcal{J}(\Phi)$  the set of all complete and consistent subsets of  $\Phi$ .

A *coalition*  $C \subseteq \{1, \dots, k\}$  is a subset of the agents. A *profile* is a vector  $\mathbf{J} := (J_1, \dots, J_k) \in \mathcal{J}(\Phi)^k$  containing one complete and consistent judgment set for each agent. For each formula  $\phi \in \Phi$ :

$$N_\phi^{\mathbf{J}} := \{i \in \{1, \dots, k\} \mid \phi \in J_i\}$$

is the subset of agents accepting  $\phi$  as true.

A (*resolute*) *aggregation rule* for  $\Phi$  is a function:

$$f : \mathcal{J}(\Phi)^k \mapsto 2^\Phi.$$

An aggregator is complete, complement-free and/or consistent if  $f(\mathcal{J}(\Phi)^k)$  is complete, complement-free and/or consistent respectively. In general, aggregators are required to satisfy various properties. We present many of them in the following sections.

We end this section by demonstrating the correlation between the preference framework of Subsec. [5.1.1] and the agenda framework. First, observe

that since given a set  $\mathcal{D}$  of alternatives there are no logical consistency assumptions for the preference orderings, there is no “natural” way to express agendas in that framework. On the other hand, we can define a propositional variable  $p^{(a,b)}$  to mean that  $a$  is strictly preferred than  $b$ , for each  $a, b \in \mathcal{D}$ . Now, if  $(a_1, \dots, a_l)$  is a total ordering of the elements of  $\mathcal{D}$ , we can express it with the formula:

$$\bigwedge_{i=1, \dots, l-1} p^{(a_i, a_{i+1})}. \quad (5.1)$$

The main disadvantage here is the size of the agenda. For a set  $\mathcal{D}$  of size  $l$ , we need  $l^2$  propositional variables and  $l! \approx l^l$  formulas of Eq. (5.1).

As mentioned before, Judgment Aggregation was first formalized using agendas. Nevertheless, there are various different, and in some sense equivalent, frameworks that have been developed over the years, like Wilson’s *general attributes* [217], which led to the so called *abstract* framework that was developed by Dokow and Holzman [76, 77], the *property spaces* of Nehring and Puppe [175] and Grandi and Endriss’s *integrity constraints* [113]. In what follows, we take particular interest in the abstract framework (Sec. 5.2) and in the integrity constraints (Sec. 5.3).

Let  $\Phi = \{\phi_1, \dots, \phi_n\}$  be an agenda. For a formula  $\psi$ , let:

$$\psi^x := \begin{cases} \psi & \text{if } x = 1, \\ \neg\psi & \text{else.} \end{cases} \quad (5.2)$$

We say that the *domain* of  $\Phi$  is:

$$X_\Phi := \left\{ \mathbf{x} = (x_1, \dots, x_n) \mid \bigwedge_{i=1}^n \phi_i^{x_i} \text{ is satisfiable} \right\}.$$

That is,  $X_\Phi$  is the set of all consistent judgments of the formulas in the agenda  $\Phi$ . In what follows, we abstract from considering explicit agendas and consider such domains on their own.

## 5.2 Abstract Framework

In the sequel, we have a fixed set  $I = \{1, \dots, n\}$  of issues. Let  $\mathcal{D}$  be a finite set of cardinality at least 2, representing the possible positions (voting options) on the issues. If  $|\mathcal{D}| = 2$  (i.e., if for every issue only a “yes” or

“no” vote is allowed), we say that we are in the *binary* or the *Boolean framework*; otherwise, we say that we are in the *non-binary* or the *non-Boolean framework*.

Let  $X$  be a non-empty subset of  $\mathcal{D}^n$  that represents the feasible voting patterns. We write  $X_j$ ,  $j = 1, \dots, n$  to denote the  $j$ -th projection of  $X$ . We assume that  $|X_j| \geq 2$ , for  $j = 1, \dots, n$  as a non-degeneracy condition. Note that in general, it does not (necessarily) hold that  $X = \prod_{j=1}^n X_j$ .

Let  $k \geq 2$  be an integer representing the number of voters. The elements of  $(\mathcal{D}^n)^k$  can be viewed as  $k \times n$  matrices, whose rows correspond to voters and whose columns correspond to issues. If  $A = \{\mathbf{a}^1, \dots, \mathbf{a}^k\}$  is such a matrix, we write  $x_j^i$  to denote the entry of the matrix in row  $i$  and column  $j$ ; clearly,  $x_j^i$  stands for the vote of voter  $i$  on issue  $j$ . The row vectors of such matrices will be denoted as  $\mathbf{a}^1, \dots, \mathbf{a}^k$ , and the column vectors as  $\mathbf{a}_1, \dots, \mathbf{a}_n$ .

An aggregator of arity  $k$  is a function  $F : (\mathcal{D}^n)^k \mapsto \mathcal{D}^n$  such that  $X$  is closed under  $F$ , meaning that if  $A \in X^k$ , then  $F(A) \in X$ . Requiring that  $X$  is closed under  $F$  reflects the rationality of  $F$ , while requiring that  $F$  is defined on  $(\mathcal{D}^n)^k$  reflects the universality of  $F$ . Actually, universality usually refers to  $F$  being definable on  $X^k$ ; however, it is technically advantageous, and imposes no essential restriction, to assume universality with respect to all possible vectors of votes, even “irrational” ones (for such votes,  $F$  may return an arbitrary value).

An aggregator  $F$  is called *dictatorial on  $X$*  if there is a number  $d \in \{1, \dots, k\}$  such that for every  $A \in X^k$ , we have that  $F(A) = x^d$  (i.e.,  $F(A)$  is equal to the  $d$ -th row of  $A$ ).

A set  $X$  of feasible voting patterns is called a *possibility domain* if, for some number  $k \geq 2$ , there exists a non-dictatorial aggregator of arity  $k$ . Otherwise,  $X$  is called an *impossibility domain*. Thus, a possibility domain is, by definition, one where aggregation is possible for societies of some cardinality, namely, the arity of the non-dictatorial aggregator.

Following List and Puppe [166], we define the notions below concerning properties of aggregators on a domain  $X$ .

An aggregator  $F$  is called *anonymous* if for every two matrices  $A, B \in X^k$  such that the rows of  $\mathbf{b}^1, \dots, \mathbf{b}^k$  of  $B$  are a permutation of the rows  $\mathbf{a}^1, \dots, \mathbf{a}^k$  of  $A$ , we have that  $F(A) = F(B)$ .

An aggregator  $F$  is called *systematic* if for every two issues  $i, j$  and every two matrices  $A, B \in X^k$ , if  $\mathbf{a}_i = \mathbf{b}_j$  (i.e. if the  $i$ -th column of  $A$  coincides with the  $j$ -th column of  $B$ ), then the  $i$ -th coordinate of the vector  $F(A)$  coincides with the  $j$ -th coordinate of the vector  $F(B)$ . In effect, an aggregator is

systematic if it arises from a polymorphism of  $X$ .

Much weaker than the notion of systematicity is the notion of *independence*. An aggregator  $F$  is called *independent of irrelevant alternatives (IIA)* or just *independent* if, for every issue  $j$  and for every two matrices  $A, B \in X^k$ , if  $\mathbf{a}_j = \mathbf{b}_j$  then the  $j$ -th coordinate of the vector  $F(A)$  coincides with the  $j$ -th coordinate of the vector  $F(B)$ . It can be easily shown that an aggregator  $F$  is IIA if and only if for every issue  $j$ , there is a function  $f_j : \mathcal{D}^k \mapsto \mathcal{D}$  such that for every  $A \in X^k$ , the  $j$ -th coordinate of  $F(A)$  is equal to  $f_j(\mathbf{a}_j)$  (see, e.g., [179], Lemma 1). Thus, an aggregator is systematic if and only if it is IIA and all functions  $f_j$  are equal. Note also that an IIA aggregator is anonymous if and only if the output of each function  $f_j$ ,  $1 \leq j \leq n$ , depends only on the multi-set of the input values. Since the values of  $F$  outside  $X$  do not matter, we assume in the sequel that for all IIA aggregators, there are functions  $f_j : \mathcal{D}^k \mapsto \mathcal{D}$  such that for every  $A \in (\mathcal{D}^n)^k$  (and not just for  $A \in X^k$ ), the  $j$ -th component of  $F(A)$  is equal to  $f_j(\mathbf{a}_j)$ . Such aggregators will be denoted by  $F = (f_1, \dots, f_n)$ . In this setting, it is easy to see that an IIA aggregator  $F = (f_1, \dots, f_n)$  is dictatorial if and only if there is a  $d = 1, \dots, k$  so that for every  $j = 1, \dots, n$ , we have that  $f_j = \text{pr}_d^k$ , where  $\text{pr}_d^k$  is the  $k$ -ary projection on the  $d$ -th coordinate (for example  $\text{pr}_2^3(0, 1, 0) = 1$ ).

Observe that, lest we can *always* aggregate non-dictatorially, we have to restrict aggregators to be IIA. Indeed, let  $x_0 \in X$ . Consider the non-IIA aggregator on  $X^2$  defined by  $F(x, y) = x$ , if  $x \neq x_0$ , and  $F(x, y) = y$ , otherwise. Obviously,  $F$  is non-dictatorial if  $X$  contains at least two elements. In other words, the question of the possibility of non-dictatorial aggregation is not meaningful, unless we assume independence. So, in the sequel, we assume that aggregators are IIA. Observe, however, that the aggregator  $F$  just defined is obviously not anonymous. If we seek to characterize domains that admit anonymous aggregators, instead of non-dictatorial ones, then it is meaningful to investigate the case of non-IIA aggregators. This line of research was taken by Nehring and Puppe [175].

An IIA aggregator  $\bar{f} = (f_1, \dots, f_n)$  is *supportive or conservative* if for every  $A \in X^k$ , and for every  $j = 1, \dots, n$ , we have that  $f_j(a_j) \in \{a_j^1, \dots, a_j^k\}$ .

An IIA aggregator  $\bar{f} = (f_1, \dots, f_n)$  is *Paretian* if for every  $A \in X^k$ , and for every  $j = 1, \dots, n$ , if  $a_j^1 = \dots = a_j^k$ , then  $f_j(\mathbf{a}_j) = a_j^1 = \dots = a_j^k$ .

In the Boolean framework, the notions of supportiveness and being Paretian are obviously equivalent. However, if more than two voting options are offered for an issue, then the notion of being supportive is in general stronger than being Paretian, but still is a natural assumption to make.

Throughout this paper, we assume that aggregators are IIA and supportive.

**Example 5.2.1.** *Suppose that  $X$  is a Cartesian product  $X = Y \times Z$ , where  $Y \subseteq \mathcal{D}^l$  and  $Z \subseteq \mathcal{D}^{n-l}$ , with  $1 \leq l < n$ . It is easy to see that  $X$  is a possibility domain.*

*Indeed, for every  $k \geq 2$ , the set  $X$  has non-dictatorial  $k$ -ary aggregators of the form  $(f_1, \dots, f_l, f_{l+1}, \dots, f_n)$ , where for some  $d$  and  $d'$  with  $d \neq d'$ , we have  $f_j = \text{pr}_d^k$ , for  $j = 1, \dots, l$ , and also  $f_j = \text{pr}_{d'}^k$ , for  $j = l+1, \dots, n$ . Thus, every Cartesian product of two sets of feasible patterns is a possibility domain.  $\diamond$*

Since the arity of an aggregator is the arity of its component functions, a ternary aggregator is an aggregator with components of arity three.

**Definition 5.2.1.** *Let  $X$  be a set of feasible voting patterns.*

- *$X$  admits a majority aggregator if it admits a ternary aggregator  $F = (f_1, \dots, f_n)$  such that  $f_j$  is a majority operation on  $X_j$ , for all  $j = 1, \dots, n$ .*
- *$X$  admits a minority aggregator if it admits a ternary aggregator  $F = (f_1, \dots, f_n)$  such that  $f_j$  is a minority operation on  $X_j$ , for all  $j = 1, \dots, m$ .*

Clearly,  $X$  admits a majority (resp. minority) aggregator if and only if there is a ternary aggregator  $F = (f_1, \dots, f_n)$  for  $X$  such that, for all  $j = 1, \dots, n$  and for all two-element subsets  $B_j \subseteq X_j$ , we have that  $f_j \upharpoonright B_j = \text{maj}$  (resp.  $\oplus$ ).

In social choice theory, domains admitting a majority aggregator are often called *median spaces* (see, e.g., [76]). It should also be noted that affine domains play an important role in the work of Dokow and Holzman [76]; furthermore, in social choice theory, non-affine domains are also known as *even-number-negatable* domains (see, e.g., [162]).

**Example 5.2.2.** *The set:*

$$X = \{(a, a, a), (b, b, b), (c, c, c), (a, b, b), (b, a, a), (a, a, c), (c, c, a)\}$$

*admits a majority aggregator.*



To see this, let  $F = (f, f, f)$ , where  $f : \{a, b, c\} \rightarrow \{a, b, c\}$  is as follows:

$$f(u, v, w) = \begin{cases} a & \text{if } u, v, \text{ and } w \text{ are pairwise different;} \\ \text{maj}(u, v, w) & \text{otherwise.} \end{cases}$$

Clearly, if  $B$  is a two-element subset of  $\{a, b, c\}$ , then  $f \upharpoonright B = \text{maj}$ . So, to show that  $X$  admits a majority aggregator, it remains to show that  $F = (f, f, f)$  is an aggregator for  $X$ . In turn, this amounts to showing that  $F$  is supportive and that  $X$  is closed under  $f$ . It is easy to check that  $F$  is supportive. To show that  $X$  is closed under  $f$ , let  $\mathbf{x} = (x_1, x_2, x_3), \mathbf{y} = (y_1, y_2, y_3), \mathbf{z} = (z_1, z_2, z_3)$  be three elements of  $X$ . We have to show that  $(f(x_1, y_1, z_1), f(x_2, y_2, z_2), f(x_3, y_3, z_3))$  is also in  $X$ . The only case that needs to be considered is when  $x, y$ , and  $z$  are pairwise distinct. Several sub-cases need to be considered. For instance, if  $\mathbf{x} = (a, b, b), \mathbf{y} = (a, a, c), \mathbf{z} = (c, c, a)$ , then  $F(x, y, z) = (f(a, a, c), f(b, a, c), f(b, c, a)) = (a, a, a) \in X$ ; the remaining combinations are left to the reader.  $\diamond$

**Example 5.2.3.** The set  $X = \{(a, b, c), (b, a, a), (c, a, a)\}$  admits a minority aggregator.

To see this, let  $F = (f, f, f)$ , where  $f : \{a, b, c\} \rightarrow \{a, b, c\}$  is as follows:

$$f(u, v, w) = \begin{cases} a & \text{if } u, v, \text{ and } w \text{ are pairwise different;} \\ \oplus(u, v, w) & \text{otherwise.} \end{cases}$$

Clearly, if  $B$  is a two-element subset of  $\{a, b, c\}$ , then  $f \upharpoonright B = \oplus$ . So, to show that  $X$  admits a minority aggregator, it remains to show that  $F = (f, f, f)$  is an aggregator for  $X$ . In turn, this amounts to showing that  $F$  is supportive and that  $X$  is closed under  $f$ . It is easy to check that  $F$  is supportive. To show that  $X$  is closed under  $f$ , let  $\mathbf{x} = (x_1, x_2, x_3), \mathbf{y} = (y_1, y_2, y_3), \mathbf{z} = (z_1, z_2, z_3)$  be three elements of  $X$ . We have to show that  $(f(x_1, y_1, z_1), f(x_2, y_2, z_2), f(x_3, y_3, z_3))$  is also in  $X$ . The only case that needs to be considered is when  $\mathbf{x}, \mathbf{y}$ , and  $\mathbf{z}$  are distinct, say,  $\mathbf{x} = (a, b, c), \mathbf{y} = (b, a, a), \mathbf{z} = (c, a, a)$ . In this case, we have that:

$$(f(a, b, c), f(b, a, a), f(c, a, a)) = (a, b, c) \in X.$$

Since  $f$  is not affected by permutations of the input, the proof is complete.  $\diamond$

So far, we have given examples of possibility domains only. Next, we give an example of an impossibility domain in the Boolean framework.

**Example 5.2.4.** Let  $W = \{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}$  be the 1-in-3 relation, i.e., the set of all Boolean tuples of length 3 in which exactly one 1 occurs.

We claim that  $W$  is an impossibility domain. It is not hard to show that  $W$  is not affine and that it does not admit a non-dictatorial binary aggregator. We will see that this implies that  $W$  is an impossibility domain.  $\diamond$

Every logical relation  $X \subseteq \{0, 1\}^m$  gives rise to a generalized satisfiability problem in the context studied by Scheafer [192]. We point out that the property of  $X$  being a possibility domain in the Boolean framework is not related to the tractability of the associated generalized satisfiability problem. Concretely, the set  $W$  in Example 5.2.4 is an impossibility domain and its associated generalized satisfiability problem is the NP-complete problem POSITIVE 1-IN-3-SAT. As discussed earlier, the Cartesian product  $W \times W$  is a possibility domain. Using the results in [192], however, it can be verified that the generalized satisfiability problem arising from  $W \times W$  is NP-complete. At the same time, the set  $\{0, 1\}^n$  is trivially a possibility domain and gives rise to a trivially tractable satisfiability problem. Thus, the property of  $X$  being a possibility domain is not related to the tractability of the generalized satisfiability problem arising from  $X$ .

Nonetheless, in Ch. 7 we establish the equivalence between the stronger notion of  $X$  being a *uniform possibility domain* and the weaker notion of the tractability of the *multi-sorted* generalized satisfiability problem arising from  $X$ , where each issue is taken as a different sort. Actually, we establish this equivalence not only for satisfiability problems but also for constraint satisfaction problems whose variables range over arbitrary finite sets.

There has been a significant body of earlier work on possibility domains. Here, we summarize some of the results that relate the notion of a possibility domain to the notion of a set being *totally blocked*, a notion originally introduced in the context of the Boolean framework by Nehring and Puppe [174]. As stated earlier, a set  $X$  of possible voting patterns is totally blocked if, intuitively, “any position on any issue can be deduced from any position on any issue”; this intuition is formalized by asserting that a certain directed graph  $G_X$  associated with  $X$  is strongly connected.

In the case of the Boolean framework, Dokow and Holzman [76] obtained the following necessary and sufficient condition for a set to be a possibility domain. The necessity of this condition is also contained in Dietrich and List [74].

**Theorem 5.2.1** (Dokow and Holzman [76], Theorem 2.2], Dietrich and List [74], Theorem 2]). *Let  $X \subseteq \{0, 1\}^n$  be a set of feasible voting patterns. The following statements are equivalent.*

- $X$  is a possibility domain.
- $X$  is affine or  $X$  is not totally blocked.

For the non-Boolean framework, Dokow and Holzman [77] found the following connection between the notions of totally blocked and possibility domain.

**Theorem 5.2.2** (Dokow and Holzman [77], Theorem 2]). *Let  $X$  be a set of feasible voting patterns. If  $X$  is not totally blocked, then  $X$  is a possibility domain; in fact, there is a non-dictatorial  $k$ -ary aggregator, for every  $k \geq 2$ .*

Note that, in the case of the Boolean framework, Theorem [5.2.2] was stated and proved as Claim 3.6 in [76]. For the non-Boolean framework, Szegedy and Xu [207] obtained a sufficient and necessary condition for a totally blocked set  $X$  to be a possibility domain.

**Theorem 5.2.3** (Szegedy and Xu [207], Theorem 8]). *Let  $X$  be a set of feasible voting patterns that is totally blocked. The following statements are equivalent.*

- $X$  is a possibility domain.
- $X$  admits a non-dictatorial ternary aggregator.

Note that, in the case of the Boolean framework, Theorem [5.2.3] follows from the preceding Theorems [5.2.1] and [5.2.2], the latter in the Boolean framework.

A binary non-dictatorial aggregator can also be viewed as a ternary one, where one of the arguments is ignored. By considering whether or not  $X$  is totally blocked, Theorems [5.2.2] and [5.2.3] imply the following corollary, which characterizes possibility domains without involving the notion of total blockedness; to the best of our knowledge, this result has not been explicitly stated previously.

**Corollary 5.2.1.** *Let  $X$  be a set of feasible voting patterns. The following statements are equivalent.*

1.  $X$  is a possibility domain.
2.  $X$  admits a non-dictatorial ternary aggregator.

For the rest of this Section *only*, we drop the assumption that aggregators should be supportive (conservative) and we assume them to be just Paretian (unanimous). Note that in the non-Boolean framework, as remarked earlier, these two notions are not equivalent. The following result follows easily from extant results (Theorem 5.2.2 above and Theorem 8 for the Paretian case in [207]), although again, to the best of our knowledge, it has not been explicitly mentioned before.

**Corollary 5.2.2.** *Let  $d = \max\{|X_j| \mid j = 1, \dots, n\}$ . Then the following statements are equivalent:*

1.  $X$  is a possibility domain (in the Paretian sense).
2.  $X$  admits a non-dictatorial and Paretian aggregator of arity at most  $\max(d, 3)$ .

It is known that any Boolean set  $X$  can be described as the set of satisfying assignment of a propositional formula  $\phi$ , in the sense that  $\text{Mod}(\phi) = X$  (see Enderton [80]). In the context of Judgment aggregation, such formulas are called *integrity constraints*.

### 5.3 Integrity Constraints

Recall the notation of Sec. 2.2, Fig. 2.2. In what follows, we will assume, except if specifically noted, that  $n$  denotes the number of variables of a formula  $\phi$  and  $m$  the number of its clauses.

Recall that a Horn clause is a clause with at most one positive literal. A dual Horn is a clause with at most one negative literal. A formula that contains only Horn (dual Horn) clauses is called Horn (dual Horn, respectively). Generalizing the notion of a clause, we will also call clauses sets of literals connected with exclusive OR (or direct sum), the logical connective that corresponds to summation in  $\{0, 1\} \pmod 2$ . Formulas obtained by considering a conjunction of such clauses are called affine. Finally, bijunctive are called the formulas whose clauses, in inclusive disjunctive form, i.e. whose literals are connected with the logical OR, have at most two literals. A domain

$D \subseteq \{0, 1\}^n$  is called Horn, dual Horn, affine or bijunctive respectively, if there is a Horn, dual Horn, affine or bijunctive formula  $\phi$  of  $n$  variables such that  $\text{Mod}(\phi) = D$ .

We have presented the above notions and results without many details, as they are all classical results. For the notions that follow we give more detailed definitions and examples. The first one, as far as we can tell, dates back to 1978 (see Lewis [160]).

**Definition 5.3.1.** *A formula  $\phi$  whose variables are among the elements of the set  $V = \{x_1, \dots, x_n\}$  is called *renamable Horn*, if there is a subset  $V_0 \subseteq V$  so that if we replace every appearance of every negated literal  $l$  from  $V_0$  with the corresponding positive one and vice versa,  $\phi$  is transformed to a Horn formula.*

The process of replacing the literals of some variables with their logical opposite ones, is called a *renaming* of the variables of  $\phi$ . It is straightforward to see that any dual-Horn formula is renamable Horn (just rename all its variables).

**Example 5.3.1.** *Consider the formulas  $\phi_1 = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_3 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee \neg x_5)$  and  $\phi_2 = (\neg x_1 \vee x_2 \vee x_3 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_4 \vee x_5)$ , defined over  $V = \{x_1, x_2, x_3, x_4, x_5\}$ .*

*The formula  $\phi_1$  is renamable Horn. To see this, let  $V_0 = \{x_1, x_2, x_3, x_4\}$ . By renaming these variables, we get the Horn formula  $\phi_1^* = (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (x_2 \vee \neg x_3 \vee \neg x_5)$ . On the other hand, it is easy to check that  $\phi_2$  cannot be transformed into a Horn formula for any subset of  $V$ , since for the first clause to become Horn, at least two variables from  $\{x_2, x_3, x_4\}$  have to be renamed, making the second clause not Horn.  $\diamond$*

It turns out that whether a formula is renamable Horn can be checked in linear time. There are several algorithms that do that in the literature, with the one of del Val [69] being a relatively recent such example. The original non-linear one was given by Lewis [160]. By the construction presented there, it is easy to observe that a bijunctive formula is renamable Horn if and only if it is satisfiable. Indeed, let  $\alpha$  be an assignment satisfying  $\phi$  and rename all the variables  $x \in V$  such that  $\alpha(x) = 1$ . Then, every clause of  $\phi$  either has a positive literal that is renamed, or a negative one that is not renamed.

We now proceed with introducing several syntactic types of formulas:

**Definition 5.3.2.** A formula is called separable if its variables can be partitioned into two non-empty disjoint subsets so that no clause of it contains literals from both subsets.

**Example 5.3.2.** The formula  $\phi_3 = (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_4 \vee x_5)$  is separable. Indeed, for the partition  $V_1 = \{x_1, x_2, x_3\}$ ,  $V_2 = \{x_4, x_5\}$  of  $V$ , we have that no clause of  $\phi_3$  contains variables from both subsets of the partition. On the other hand, there is no such partition of  $V$  for neither  $\phi_1$  nor  $\phi_2$  of the previous example.  $\diamond$

The fact that separable formulas can be recognized in linear time is relatively straightforward (see Proposition [7.2.1](#)).

**Remark 5.3.1.** The notion of separability in Definition [5.3.2](#) is a purely syntactic property. Lang et al. [\[158\]](#) define a semantic version of separability, which, in our framework, reads as follows.

Let  $A \subseteq V$  and  $\phi$  be a formula on  $V$ . Define  $\phi_A$  to be the projection of  $\phi$  to the variables of  $A$ , i.e.  $\phi$  with all appearances of variables not in  $A$  deleted.  $\phi$  is called semantically separable if there is a partition  $(A, B)$  of  $V$  such that  $\text{Mod}(\phi) = \text{Mod}(\phi_A) \times \text{Mod}(\phi_B)$ . Such a partition  $(A, B)$  is called an independent partition of  $V$ .

Obviously, a separable formula satisfies semantic separability. On the other hand, consider that formula:

$$\psi = (x \vee \neg y \vee z) \wedge (\neg x \vee y \vee z) \wedge (x \vee \neg y \vee \neg z) \wedge (\neg x \vee y \vee \neg z).$$

Clearly, this is not a separable formula in the sense of Definition [5.3.2](#). On the other hand, for the partition  $(\{x, y\}, \{z\})$  of its set of variables, it holds that:

$$\text{Mod}(\psi) = \{(0, 0), (1, 1)\} \times \{0, 1\} = \text{Mod}(\psi_{\{x, y\}}) \times \text{Mod}(\psi_{\{z\}}),$$

and thus  $\psi$  is semantically separable.  $\diamond$

We now introduce the following notions:

**Definition 5.3.3.** A formula  $\phi$  is called partially Horn if there is a nonempty subset  $V_0 \subseteq V$  such that (i) the clauses containing only variables from  $V_0$  are Horn and (ii) the variables of  $V_0$  appear only negatively (if at all) in a clause containing also variables not in  $V_0$ .

If a formula  $\phi$  is partially Horn, then any non-empty subset  $V_0 \subseteq V$  that satisfies the requirements of Definition [5.3.3](#) will be called an *admissible set of variables*. Easily, if both  $V_0, V_1 \subseteq V$  are admissible sets of variables, then so is  $V_0 \cup V_1$ . Also the Horn clauses that contain variables only from  $V_0$  will be called *admissible clauses* (the set of admissible clauses might be empty). A Horn clause with a variable in  $V \setminus V_0$  will be called *inadmissible* (the reason for the possible existence of such clauses will be made clear in the following example).

Notice that a Horn formula is, trivially, partially Horn too, as is a formula that contains at least one negative pure literal. It immediately follows that the satisfiability problem remains NP-complete even when restricted to partially Horn formulas (just add a dummy negative pure literal). In Computational Social Choice though, domains are considered to be non-empty as a non-degeneracy condition. Actually, it is usually assumed that the projection of a domain to any one of the  $n$  issues is the set  $\{0, 1\}$ .

**Example 5.3.3.** *We first examine the formulas of the previous examples.  $\phi_1$  is partially Horn, since it contains the negative pure literal  $\neg x_5$ . The Horn formula  $\phi_1^*$  is also trivially partially Horn. On the other hand,  $\phi_2$  and  $\phi_3$  are not, since for every possible  $V_0 \subseteq \{x_1, x_2, x_3, x_4, x_5\}$ , we either get non-Horn clauses containing variables only from  $V_0$ , or variables of  $V_0$  that appear positively in inadmissible clauses.*

*The formula  $\phi_4 = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_3 \vee x_4)$  is partially Horn. Its first three clauses are Horn, though the third has to be put in every inadmissible set, since  $x_3$  appears positively in the fourth clause which is not Horn. The first two clauses though constitute an admissible set of Horn clauses. Finally,  $\phi_5 = (x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_3 \vee x_4)$  is not partially Horn. Indeed, since all its variables appear positively in some clause, we need at least one clause to be admissible. The first two clauses of  $\phi_5$  are Horn, but we will show that they both have to be included in an inadmissible set. Indeed, the second has to belong to every inadmissible set since  $x_3$  appears positively in the third, not Horn, clause. Furthermore,  $x_2$  appears positively in the second clause, which we just showed to belong to every inadmissible set. Thus, the first clause also has to be included in every inadmissible set, and therefore  $\phi_5$  is not partially Horn.  $\diamond$*

Accordingly to the case of renamable Horn formulas, we define:

**Definition 5.3.4.** *A formula is called renamable partially Horn if some of its variables can be renamed (in the sense of Definition [5.3.1](#)) so that it becomes partially Horn.*

Observe that any Horn, renamable horn or partially Horn formula is trivially renamable partially Horn. Also, a formula with at least one pure positive literal is renamable partially Horn, since by renaming the corresponding variable, we get a formula with a pure negative literal.

**Example 5.3.4.** *All formulas of the previous examples are renamable partially Horn:  $\phi_1^*$ ,  $\phi_1$  and  $\phi_4$  correspond to the trivial cases we discussed above, whereas  $\phi_2$ ,  $\phi_3$  and  $\phi_5$  all contain the pure positive literal  $x_4$ .*

*Lastly, we examine two more formulas:  $\phi_6 = (\neg x_1 \vee x_2 \vee x_3 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_4 \vee x_5)$  is easily not partially Horn, but by renaming  $x_4$  and  $x_5$ , we obtain the partially Horn formula  $\phi_6^* = (\neg x_1 \vee x_2 \vee x_3 \vee \neg x_4) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_4 \vee \neg x_5)$ , where  $V_0 = \{x_4, x_5\}$  is the set of admissible variables. On the other hand, the formula  $\phi_7 = (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3)$  is not renamable partially Horn. Indeed, whichever variables we rename, we end up with one Horn and one non-Horn clause, with at least one variable of the Horn clause appearing positively in the non-Horn clause.  $\diamond$*

We prove, by Theorem [7.2.1](#), that checking whether a formula is renamable partially Horn can be done in linear time in the length of the formula.

**Remark 5.3.2.** *Let  $\phi$  be a renamable partially Horn formula, and let  $\phi^*$  be a partially Horn formula obtained by renaming some of the variables of  $\phi$ , with  $V_0$  being the admissible set of variables. Let also  $\mathcal{C}_0$  be an admissible set of Horn clauses in  $\phi^*$ . We can assume that only variables of  $V_0$  have been renamed, since the other variables are not involved in the definition of being partially Horn. Also, we can assume that a Horn clause of  $\phi^*$  whose variables appear only in clauses in  $\mathcal{C}_0$  belongs to  $\mathcal{C}_0$ . Indeed, if not, we can add it to  $\mathcal{C}_0$ .*

*Another technical point is that, contrary to the case of partially Horn formulas, it can be the case that  $V_0$  and  $V_1$  are distinct subsets of variables of a formula which, when renamed, make the formula partially Horn, but  $V_0 \cup V_1$  is not. For example, let*

$$\psi = (\neg x \vee \neg y) \wedge (x \vee z) \wedge (y \vee \neg z),$$



which is easily not partially Horn. Let  $V_0 = \{x\}$  and  $V_1 = \{y, z\}$ . By renaming these subsets of variables, we obtain the two Horn formulas:

$$\begin{aligned}\psi_{\{x\}}^* &= (x \vee \neg y) \wedge (\neg x \vee z) \wedge (y \vee \neg z) \\ \psi_{\{y,z\}}^* &= (\neg x \vee y) \wedge (x \vee \neg z) \wedge (\neg y \vee z),\end{aligned}$$

whereas, by renaming  $V_0 \cup V_1 = \{x, y, z\}$ , we obtain the formula:

$$\psi^* = (x \vee y) \wedge (\neg x \vee \neg z) \wedge (\neg y \vee z),$$

which is not partially Horn. ◇

**Definition 5.3.5.** A formula is called a possibility integrity constraint if it is either separable, or renamable partially Horn or affine.

From the above and the fact that checking whether a formula is affine is easy we get Theorem [7.2.2](#), which states that checking whether a formula is a possibility integrity constraint can be done in polynomial time in the size of the formula.

Now, let  $V, V'$  be two disjoint sets of variables. By further generalizing the notion of a clause of a CNF formula, we say that a  $(V, V')$ -generalized clause is a clause of the form:

$$(l_1 \vee \cdots \vee l_s \vee (l_{s+1} \oplus \cdots \oplus l_t)),$$

where the literal  $l_j$  corresponds to variable  $v_j$ ,  $j = 1, \dots, t$ ,  $s < t$ ,  $v_1, \dots, v_s \in V$  and  $v_{s+1}, \dots, v_t \in V'$ . Such a clause is falsified by exactly those assignments that falsify every literal  $l_i$ ,  $i = 1, \dots, s$  and satisfy an even number of literals  $l_j$ ,  $j = s + 1, \dots, t$ . An affine clause is trivially a  $(V, V')$ -generalized clause, where all its literals correspond to variables from  $V'$ . Consider now the following syntactic type of formulas.

**Definition 5.3.6.** A formula  $\phi$  is a local possibility integrity constraint (lpic) if there are two disjoint subsets  $V_0, V_1 \subseteq V$ , with  $V_0 \cup V_1 = V$ , such that:

1. by renaming some variables of  $V_0$ , we obtain a partially Horn formula  $\phi^*$ , whose set of admissible variables is  $V_0$  and
2. the clauses containing variables from  $V_1$  are  $(V_0, V_1)$ -generalized clauses.

**Example 5.3.5.** *Easily, every (renamable) Horn or affine formula is an lpic. On the other hand, consider the following formula:*

$$\phi_8 = (\neg x_1 \vee \neg x_2) \wedge (x_1 \vee x_2 \vee (x_3 \oplus x_4 \oplus x_5)).$$

$\phi_8$  is not an lpic, since  $V_0$  must contain both  $x_1$  and  $x_2$ , and under any renaming, either at least one of them will appear positively in the second, non-admissible clause, or the first will cease being a Horn clause.  $\diamond$

By Definition 5.3.6, an lpic  $\phi$  over  $V$ , where  $V_0 \neq \emptyset$ , is a renamable partially Horn formula. Otherwise, if  $V_0 = \emptyset$ ,  $\phi$  is affine.

To end this preliminary discussion about propositional formulas, we consider *prime* formulas. Given a clause  $C$  of a formula  $\phi$ , we say that a *sub-clause* of  $C$  is any non-empty clause created by deleting at least one literal of  $C$ . In Quine [187] and Zanuttini and Hébrard [219], we find the following definitions:

**Definition 5.3.7.** *A clause  $C$  of a formula  $\phi$  is a prime implicate of  $\phi$  if no sub-clause of  $C$  is logically implied by  $\phi$ . Furthermore,  $\phi$  is prime if all its clauses are prime implicates of it.*

In Subsec. 7.2.2, we use this notion in order to efficiently construct formulas whose sets of models is a (local) possibility domain.

In the sequel, we will assume that all domains  $\mathcal{D} \subseteq \{0, 1\}^n$  are *non-degenerate*, i.e. for any  $j \in \{1, \dots, n\}$ , it holds that  $\mathcal{D}_j = \{0, 1\}$ . This is a common assumption in Social Choice Theory, which reflects the idea that voting is nonsensical when there is only one option. Consequently, we will also assume that the formulas we consider have non-degenerate domains too.

**Example 5.3.6.** *Theorem 5.2.1 directly implies that the truth set of any affine formula is a possibility domain. Consider now the formula  $\phi_7 = (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3)$  of Example 5.3.4. It holds that:*

$$\text{Mod}(\phi_7) = \{0, 1\}^3 \setminus \{(1, 0, 0), (0, 1, 1)\}.$$

*By checking all  $4^3$  different triples of binary unanimous operators and since  $\text{Mod}(\phi_7)$  is not affine, one can see that  $\text{Mod}(\phi_7)$  is an impossibility domain. On the other hand, let*

$$\phi_9 := (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_4 \vee x_5 \vee x_6) \wedge (x_4 \vee \neg x_5 \vee \neg x_6).$$

Then, we have that:

$$\text{Mod}(\phi_9) = \text{Mod}(\phi_7) \times \text{Mod}(\phi_7),$$

which is a possibility domain, since every Cartesian product is (see Kirousis et al. [144, Example 2.1]). Finally, for:

$$\phi_6 = (\neg x_1 \vee x_2 \vee x_3 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_4 \vee x_5),$$

of Example [5.3.4] we have that:

$$\text{Mod}(\phi_6) = (\text{Mod}(\phi_7) \times \{(0, 0), (0, 1)\}) \cup \left( (\{0, 1\}^3 \setminus \{(1, 0, 0)\}) \times \{(1, 1)\} \right)$$

is a possibility domain, since it admits the binary non-dictatorial aggregator  $(\text{pr}_1^2, \text{pr}_1^2, \text{pr}_1^2, \vee, \vee)$ .  $\diamond$

Nehring and Puppe [175] defined a type of non-dictatorial aggregators they called *locally non-dictatorial*. A  $k$ -ary aggregator  $(f_1, \dots, f_n)$  is locally non-dictatorial if  $f_j \neq \text{pr}_d^k$ , for all  $d \in \{1, \dots, k\}$  and  $j = 1, \dots, n$ .

**Definition 5.3.8.**  $D$  is a local possibility domain (lpd) if it admits a locally non-dictatorial aggregator.

Kirousis et al. [144] introduced these domains as *uniform* non-dictatorial domains, both in the Boolean and non-Boolean framework and provided a characterization for them.

**Theorem 5.3.1** (Kirousis et al. [144], Theorem 5.5).  $D \subseteq \{0, 1\}^n$  is a local possibility domain if and only if it admits a ternary aggregator  $(f_1, \dots, f_n)$  such that  $f_j \in \{\wedge^{(3)}, \vee^{(3)}, \text{maj}, \oplus\}$ , for  $j = 1, \dots, n$ .

**Example 5.3.7.** It holds that neither  $\text{Mod}(\phi_6)$  nor  $\text{Mod}(\phi_7)$  of Example [5.3.4], nor  $\text{Mod}(\phi_9)$  of Example [5.3.6] are local possibility domains, since every aggregator they admit has components that are projection functions. On the other hand, for:

$$\phi_{10} = (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \neg x_3),$$

we have that:

$$\text{Mod}(\phi_{10}) = \{0, 1\}^3 \setminus \{(0, 0, 1), (1, 0, 0)\},$$

that is a possibility domain, since it admits  $(\wedge, \vee, \wedge)$ .  $\diamond$



# Chapter 6

## Algorithmic Lovász Local Lemma

In this chapter, we present a probabilistic approach in designing algorithms that guarantee the existence of and find solutions to CSPs. All that follows is situated in the variable framework, where we have a set of  $l$  random variables  $X_1, \dots, X_l$ , defined on a common probability space and taking values in a finite domain  $\mathcal{D}$  uniformly at random. Let  $E_1, \dots, E_m \subseteq \mathcal{D}^l$  be undesirable events, where  $\text{sc}(E_j) \subseteq \{1, \dots, l\}$  denotes the scope of  $E_j$ , that is the set of random variables it depends on. Our aim is to find a point in  $D^n$  such that none of the events occurs. The algorithms we present are for some of the lopsided notion presented in Ch. 3, but work also for the simple dependency condition of intersecting scopes. Our inspiration comes mainly from the probabilistic RESAMPLE algorithm of Moser [171].

In Sec. 6.1 we present an algorithm for the symmetric version of the LLL under the VDL condition. In Sec. 6.2 and 6.3 we provide algorithms for the, more general, asymmetric version of the LLL and Shearer's lemma, under the d-dependency condition. Finally, in Sec. 6.4 and 6.5 we implement our technique to find acyclic edge colorings and construct separable codes respectively.

### 6.1 Symmetric VDL Lovász Local Lemma

We begin by presenting our algorithm (Subsec. 6.1.1) for the symmetric LLL and describing a forest structure to depict its executions (Subsec. 6.1.2).

Finally, we proceed by analyzing it, to obtain a bound to the number of steps it will perform (Subsec. 6.1.3).

### 6.1.1 The Algorithm

The notion of lopsidedependency we use hereafter in this section is the directed notion of Definition 3.2.7, which we called VDL. The outwards neighborhood  $N_j$  of a vertex  $j$  is thus in terms of  $G^{VDL}$ .

To prove Theorem 3.3.4 in our framework, we give a Moser-like algorithm we call VDL-ALG, algorithm 1 below, and find an upper bound to the probability that it lasts for at least  $n$  rounds, i.e. the number of recursive calls of RESAMPLE.

---

#### Algorithm 1 VDL-ALG

---

- 1: Sample the variables  $X_i$ ,  $i = 1, \dots, l$  and let  $\mathbf{a} = (a_1, \dots, a_l)$  be the resulting assignment.
- 2: **while** there exists an event that occurs under the current assignment, let  $E_j$  be the least indexed such event and **do**
- 3:     RESAMPLE( $E_j$ )
- 4: **end while**
- 5: Output current assignment  $\mathbf{a}$ .

#### RESAMPLE( $E_j$ )

- 1: Resample the variables in  $\text{sc}(E_j)$ .
  - 2: **while** some event whose index is in  $N_j \cup \{j\}$  occurs under the current assignment, let  $E_k$  be the least indexed such event and **do**
  - 3:     RESAMPLE( $E_k$ )
  - 4: **end while**
- 

It is trivial to see that if VDL-ALG ever stops, it returns an assignment for which none of the “undesirable” events  $E_1, \dots, E_m$  holds. We show that at each round, VDL-ALG makes some progress that is preserved in subsequent rounds:

**Lemma 6.1.1.** *Consider an arbitrary call of RESAMPLE( $E_j$ ). Let  $\mathcal{S}$  be the set of events that do not occur at the start of this call. Then, if and when this call terminates, all events in  $\mathcal{S} \cup \{E_j\}$  do not occur.*

*Proof.* Let  $E_k \in \mathcal{S} \cup \{E_j\}$ , assume that  $\text{RESAMPLE}(E_j)$  terminates and that  $E_k$  occurs under the current assignment  $\mathbf{a}$ .

$E_k \neq E_j$ , else the  $\text{RESAMPLE}(E_j)$  call couldn't have exited the while-loop of line 2 and thus wouldn't have terminated.

Consequently,  $E_k \in \mathcal{S}$  thus, under the assignment at the beginning of  $\text{RESAMPLE}(E_j)$ ,  $E_k$  did not occur. Then, it must be the case that at some point during this resample call, some resampling of the variables caused  $E_k$  to occur.

Let  $\text{RESAMPLE}(E_s)$  be the last time  $E_k$  became occurring, and thus remained occurring until the end of  $\text{RESAMPLE}(E_j)$ . During this call, only the variables in  $\text{sc}(E_s)$  were resampled. Thus, it is easy to see that  $k \in N_s$ .

Since  $E_k$  remains occurring,  $\text{RESAMPLE}(E_s)$  couldn't have exited the while-loop of line 2 and thus couldn't have terminated. Consequently, neither could  $\text{RESAMPLE}(E_j)$ . Contradiction.  $\square$

A *root call* of  $\text{RESAMPLE}$  is any call of  $\text{RESAMPLE}$  made when executing line 3 of VDL-ALG and a *recursive call* of  $\text{RESAMPLE}$  is one made from line 2 of another  $\text{RESAMPLE}$  call. The time complexity of the algorithm will be given in terms of the number of rounds it will need to execute.

By Lemma 6.1.1, we know that the events of the root calls of  $\text{RESAMPLE}$  are pairwise distinct. Therefore:

**Corollary 6.1.1.** *There are at most  $m$   $\text{RESAMPLE}$  root calls in any execution of VDL-ALG.*

## 6.1.2 Forests

To depict an execution of VDL-ALG, we will use *rooted forests*, i.e. forests of trees such that each tree has a special node designated as its root.

The nodes of rooted forests are labeled by the events  $E_j$ ,  $j = 1, \dots, m$ , with repetitions of the labels allowed and they are ordered as follows: children of the same node are ordered as their labels are; nodes in the same tree are ordered by preorder (respecting the ordering between siblings) and finally if the label on the root of a tree  $T_1$  precedes the label of the root of  $T_2$ , all nodes of  $T_1$  precede all nodes of  $T_2$ .

The number of nodes of a forest  $\mathcal{F}$  is denoted by  $|\mathcal{F}|$ .

**Definition 6.1.1.** *A labeled rooted forest  $\mathcal{F}$  is called feasible if the following conditions hold:*

1. The labels of the roots of  $\mathcal{F}$  are pairwise distinct.
2. If  $u, v$  are siblings (have a common parent), then the labels of  $u, v$  are distinct.
3. Let  $E_i, E_j$  be the labels of nodes  $u, v$  respectively, where  $u$  is a child of  $v$ . Then,  $i \in N_j \cup \{j\}$ .

Now consider an execution of M-ALGORITHM that lasts for  $n$  rounds. We construct in a unique way, depending on the rounds, a feasible forest with  $n$  nodes as follows:

1. The forest under construction will have as many roots as root calls of RESAMPLE. These roots will be labeled by the event of the corresponding root call.
2. A tree that corresponds to a root call RESAMPLE( $E_j$ ) will have as many non-root nodes as the number of recursive calls of RESAMPLE within RESAMPLE( $E_j$ ). The non-root nodes will be labeled by the events of those recursive calls.
3. The non-root nodes are organized within the tree with root-label  $E_j$  so that a node that corresponds to a call RESAMPLE( $E_k$ ) is parent to a root that corresponds to a call RESAMPLE( $E_l$ ), if RESAMPLE( $E_l$ ) appears immediately on top of RESAMPLE( $E_k$ ) in the recursive stack that implements the root call RESAMPLE( $E_j$ ).

It is straightforward to verify, by inspecting the succession of steps of algorithm [1](#) in an execution, and making use of Lemma [6.1.1](#), that a forest constructed as above from the consecutive rounds of the execution of VDL-ALG, is indeed a feasible forest in the sense of Definition [6.1.1](#). It is not true however that every feasible forest corresponds to the consecutive rounds of some execution of VDL-ALG. For example consider a feasible forest where a node  $w$  has a child  $u$  and a descendent  $v$ , such that: (i)  $v$  is not a descendent of  $u$  and (ii)  $u$  and  $v$  have the same label (thus  $v$  cannot also be a child of  $w$ ). By Lemma [6.1.1](#), this forest could never be constructed as above.

**Definition 6.1.2.** *The witness forest of an execution of algorithm [1](#) is the feasible forest constructed by the process described above. Given a feasible forest  $\mathcal{F}$  with  $n$  nodes, we denote by  $W_{\mathcal{F}}$  the event that in the first  $n$  rounds of VDL-ALG,  $\mathcal{F}$  is constructed. This implies that VDL-ALG lasts for at least  $n$  rounds.*



We also set:

$$P_N := \Pr \left[ \bigcup_{\mathcal{F}:|\mathcal{F}|=N} W_{\mathcal{F}} \right] = \sum_{\mathcal{F}:|\mathcal{F}|=N} \Pr [W_{\mathcal{F}}], \quad (6.1)$$

where the last equality holds, because the events  $W_{\mathcal{F}}$  are disjoint. Obviously now:

$$\Pr[\text{M-Algorithm lasts for at least } n \text{ rounds}] = P_n. \quad (6.2)$$

### 6.1.3 Analysis of VDL-Alg

To find an upper bound for  $P_n$ , consider the following validation algorithm, which we call VALALG:

---

**Algorithm 2** VALALG

---

**Input:** Feasible forest  $\mathcal{F}$  with labels  $E_{j_1}, \dots, E_{j_n}$ .

- 1: Sample the variables  $X_i, i = 1, \dots, l$ .
  - 2: **for**  $s=1, \dots, n$  **do**
  - 3:     **if**  $E_{j_s}$  does not occur under the current assignment **then**
  - 4:         **return failure** and exit.
  - 5:     **else**
  - 6:         Resample the variables in  $\text{sc}(E_{j_s})$
  - 7:     **end if**
  - 8: **end for**
  - 9: **return success.**
- 

The validation algorithm VALALG, takes as input a feasible forest  $\mathcal{F}$  with  $n$  nodes, labeled with the events  $E_{j_1}, \dots, E_{j_n}$  (ordered as their respective nodes) and outputs a Boolean value **success** or **failure**.

Intuitively, with input a feasible forest with labels  $E_{j_1}, \dots, E_{j_n}$ , VALALG initially generates a random sampling of the variables, then checks the current event and, if it holds, resamples its variables and goes to the next event. If the algorithm manages to go through all events, it returns **success**, otherwise, at the first event that does not hold under the current assignment, it returns **failure** and stops. Note that whether there are occurring events under the last assignment generated by VALALG is *irrelevant* to its success of failure.

A round of VDL-ALG-Val is the duration of any **for** loop executed at lines [2-8](#).

We first give a result about the probability distribution of the assignment to the variables  $X_i$  during the execution of VALALG.

**Lemma 6.1.2** (Randomness Lemma). *At the beginning of any round of the algorithm [2](#), the distribution of the current assignment of values to the variables  $X_i$ ,  $i = 1, \dots, l$  is as if all variables have been sampled anew. Therefore the probability of any event occurring at such an instant is bounded from above by  $p$ .*

*Proof.* The result follows from the fact that if at the previous iteration of the **for** loop of line [2](#) the event  $E_{j_s}$  has been checked, only the values of the variables in  $\text{sc}(E_{j_s})$  have been exposed, and these are resampled anew.  $\square$

**Definition 6.1.3.** *Given a feasible forest  $\mathcal{F}$  with  $n$  nodes, we say that  $\mathcal{F}$  is validated by VALALG if the latter returns **success** on input  $\mathcal{F}$ . The event of this happening is denoted by  $V_{\mathcal{F}}$ . We also set:*

$$\hat{P}_n = \sum_{\mathcal{F}: |\mathcal{F}|=n} \Pr[V_{\mathcal{F}}]. \quad (6.3)$$

We now claim:

**Lemma 6.1.3.** *For any feasible forest  $\mathcal{F}$ , the event  $W_{\mathcal{F}}$  implies the event  $V_{\mathcal{F}}$ , therefore:*

$$P_n \leq \hat{P}_n. \quad (6.4)$$

*Proof.* Indeed, if the random choices made by an execution of VDL-ALG that produces as witness forest  $\mathcal{F}$  are made by VALALG on input  $\mathcal{F}$ , then clearly VALALG will return **success**.  $\square$

**Lemma 6.1.4.** *For any feasible forest  $\mathcal{F}$  with  $n$  nodes,  $\Pr[V_{\mathcal{F}}]$  is at most  $p^n$ .*

*Proof.* Immediate corollary of the Randomness Lemma [6.1.2](#).  $\square$

Therefore to find an upper bound for  $\hat{P}_n = \sum_{\mathcal{F}: |\mathcal{F}|=n} \Pr[V_{\mathcal{F}}]$ , it suffices to find an upper bound on the number of feasible forests with  $n$  nodes, a fairly easy exercise in enumerative combinatorics, whose solution we outline below.

First to any feasible forest  $\mathcal{F}$  with  $n$  nodes we add new leaves to get a new labeled forest  $\mathcal{F}'$  (perhaps not feasible anymore) comprising of  $m$  full  $(d+1)$ -ary trees whose internal nodes comprise the set of all nodes (internal or not) of  $\mathcal{F}$  (recall  $m$  is the total number of events  $E_1, \dots, E_m$ ). Specifically, we perform all the following additions of leaves:

1. Add to  $\mathcal{F}$  new trees, each consisting of a single root/leaf labeled with a suitable event, so that the set of labels of all roots of  $\mathcal{F}'$  is equal to the set of all events  $E_1, \dots, E_m$ . In other words, the labels of the added roots/leaves are the events missing from the list of labels of the roots of  $\mathcal{F}$ .
2. Hang from all nodes (internal or not) of  $\mathcal{F}$  new leaves, labeled with suitable events, so that the set of the labels of the children in  $\mathcal{F}'$  of a node  $u$  of  $\mathcal{F}$  labeled with  $E_j$  is equal to the set of the events in  $N_j \cup \{j\}$ . In other words, the labels of the new leaves hanging from  $u$  are the events in  $N_j \cup \{j\}$  that were missing for the labels of the children of  $u$  in  $\mathcal{F}$ .
3. Add further leaves to all nodes  $u$  of  $\mathcal{F}$  so that every node of  $\mathcal{F}$  (internal or not in  $\mathcal{F}$ ) has exactly  $d+1$  children in  $\mathcal{F}'$ ; label all these leaves with the first  $(d+1) - |N_j \cup \{j\}|$  events from the list  $E_1, \dots, E_m$ , so that labels of siblings are distinct, and therefore siblings can be ordered by the index of their labels.

Notice first that indeed, the internal nodes of  $\mathcal{F}'$  comprise the set of all nodes (internal or not) of  $\mathcal{F}$ .

Also, the labels of the nodes of a labeled forest  $\mathcal{F}'$  obtained as above are uniquely determined from the rooted planar forest structure of  $\mathcal{F}'$  when labels are ignored, but the ordering of the nodes imposed by them is retained (a forest comprised of rooted trees is called *rooted planar* if the roots are ordered and if the children of each internal node are ordered). Indeed, the roots of  $\mathcal{F}'$  are labeled by  $E_1, \dots, E_m$ ; moreover, once the the label of an internal node of  $\mathcal{F}'$  is given, the labels of its children are distinct and uniquely specified by the steps of the construction above.

Finally, distinct  $\mathcal{F}$  give rise to distinct  $\mathcal{F}'$ . By the above remarks, to find an upper bound to the number of feasible forests with  $n$  nodes (internal or not), it suffices to find an upper bound on the number of rooted planar forests with  $N$  internal nodes comprised of  $m$  full  $(d+1)$ -ary rooted planar trees.

It is well known that the number  $t_n$  of full  $(d+1)$ -ary rooted planar trees with  $n$  internal nodes is equal to  $\frac{1}{dn+1} \binom{(d+1)n}{n}$ , see e.g. [193, Theorem 5.13]. Now by Stirling's approximation easily follows that for some constant  $A > 1$ ,

depending only on  $d$ , we have:

$$t_N < A \left( \left(1 + \frac{1}{d}\right)^d (d+1) \right)^N. \quad (6.5)$$

Also obviously the number  $f_n$  of rooted planar forests with  $n$  internal nodes that are comprised of  $m$   $(d+1)$ -ary rooted planar trees is given by:

$$f_n = \sum_{\substack{n_1 + \dots + n_m = n \\ n_1, \dots, n_m \geq 0}} t_{n_1} \cdots t_{n_m}. \quad (6.6)$$

From (6.5) and (6.6) we get:

$$f_n < (An)^m \left( \left(1 + \frac{1}{d}\right)^d (d+1) \right)^n < (An)^m (e(d+1))^n. \quad (6.7)$$

So by Lemma 6.1.3, equation (6.3), Lemma 6.1.4 and equation (6.7) we get the following theorem, which is essentially a detailed restatement of Th. 3.3.4:

**Theorem 6.1.1.** *Assuming  $p$  and  $d$  are constants such that  $\left(1 + \frac{1}{d}\right)^d p(d+1) < 1$ , (and therefore if  $ep(d+1) \leq 1$ ), there exists an integer  $N$ , which depends linearly on  $m$ , and a constant  $c \in (0, 1)$  (depending on  $p$  and  $d$ ) such that if  $n/\log n \geq N$  then the probability that VDL-ALG lasts for at least  $n$  rounds is  $< c^n$ .*

Clearly, when VDL-ALG stops we have found an assignment such that none of the events occurs. Since, by the above Theorem, this happens with probability close to 1 for large enough  $n$ , Th. 3.3.4 follows. It is also straightforward to observe that with the same exact arguments, we can prove Th. 6.1.1 for  $G^s$  and  $G^{MT}$ , since  $G^{VDL}$  is sparser than both these dependency graphs.

## 6.2 Asymmetric d-dependency LLL

In the approach of the previous section (and in both approaches by Moser [171] and by Moser and Tardos [172]) we search for an assignment that avoids

the undesirable events by consecutively resampling the variables in the scopes of currently occurring events, by giving priority, when choosing the next event whose variables will be resampled, to the occurring events that belong to the extended neighborhood in the dependency graph of the last resampled event (as it was done in Moser [171]). Thus the failure of the algorithm to return a correct answer within  $n$  steps is depicted by a forest structure, which means, in some sense, that failure to produce results, will create, step after random step, a structure out of randomness, something that cannot last for long, lest the second principle of thermodynamics is violated. This is, very roughly, the intuition behind the entropic method (see Tao [209]). However, as we stressed, we analyze the algorithm by direct computations instead of referring to entropy. One key idea in this section, is to give absolute priority, when searching for the next event to be resampled, to the event itself, if it still occurs, in order to be able to utilize the  $d$ -dependency graph of the events.

Our aim is to prove Th. [3.3.5] for  $G^d$ . It is straightforward to see that the same arguments work for  $G^s$ ,  $G^{MT}$  and  $G^{VDL}$ . Also, since Th. [3.3.4] is a special case of Th. [3.3.5], we also obtain a symmetric directed LLL. We begin by stating our algorithm (Subsec. [6.2.1]) and we then analyze its time complexity (Subsec. [6.2.2]).

### 6.2.1 The Algorithm

DD-ALG, algorithm [6.2.1] below, successively produces random assignments, by resampling the variables in the scopes of occurring events, until it finds one under which no undesirable event occurs. When the variables in the scope of an occurring event  $E_j$  are resampled, the algorithm checks if  $E_j$  still occurs (lines [2] and [3] of the RESAMPLE routine) and, only in case it does not, looks for occurring events in  $E_j$ 's neighborhood in  $G^d$ . Finally, if and when all events in  $E_j$ 's neighborhood cease occurring, the algorithm looks for still occurring events elsewhere.

Obviously, if and when DD-ALG stops, it produces an assignment to the variables for which none of the events occurs. Our aim now is to bound the probability that this algorithm lasts for at least  $n$  steps. We count as a step an execution of the variable resampling command RESAMPLE in line [1] of the subroutine RESAMPLE.

Everywhere below the asymptotics are with respect to  $n$ , the number of steps, whereas the number  $l$  of variables and the number  $m$  of events are

**Algorithm 3** DD-ALG

---

```

1: Sample the variables  $X_i$ ,  $i = 1, \dots, l$  and let  $\mathbf{a}$  be the resulting assignment.
2: while there exists an event that occurs under the current assignment,
   let  $E_j$  be the least indexed such event and do
3:   RESAMPLE( $E_j$ )
4: end while
5: Output current assignment  $\mathbf{a}$ .

   RESAMPLE( $E_j$ )
1: Resample the variables in  $\text{sc}(E_j)$ .
2: if  $E_j$  occurs then
3:   RESAMPLE( $E_j$ )
4: else
5:   while some event whose index is in  $N_j$  occurs under the current
   assignment,
   let  $E_k$  be the least indexed such event and do
6:     RESAMPLE( $E_k$ )
7:   end while
8: end if

```

---

taken to be constants. The rounds, root and recursive calls are defined as in Subsec. [6.1.1](#). Again, we have a corresponding lemma to Lemma [6.1.1](#).

**Lemma 6.2.1.** *Consider an arbitrary call of RESAMPLE( $E_j$ ). Let  $\mathcal{S}_j$  be the set of events that do not occur at the start of this call. Then, if and when this call terminates, all events in  $\mathcal{S}_j \cup \{E_j\}$  do not occur.*

*Proof.* Without loss of generality, say that RESAMPLE( $E_j$ ) is the root call of RESAMPLE, suppose it terminates and that  $\mathbf{a}$  is the produced assignment of values. Furthermore, suppose that  $E_k \in \mathcal{S}_j \cup \{E_j\}$  and that  $E_k$  occurs under  $\mathbf{a}$ .

Let  $E_k \in \mathcal{S}_j$ . Then, under the assignment at the beginning of the main call,  $E_k$  did not occur. Thus, it must be the case that at some point during this call, a resampling of some variables caused  $E_k$  to occur. Let RESAMPLE( $E_s$ ) be the last time  $E_k$  became occurring, and thus remained occurring until the end of the main call.

Since  $E_k$  did not occur at the beginning of RESAMPLE( $E_s$ ), there is an assignment of values  $\mathbf{a}$  such that  $E_s, \overline{E_k}$  occur. Furthermore, for the main

call to have terminated,  $\text{RESAMPLE}(E_s)$  must have terminated too. For this to happen  $\text{RESAMPLE}(E_s)$  must have exited lines [2](#) and [3](#) of its execution. During this time, only variables in  $\text{sc}(E_s)$  were resampled and at the end,  $E_s$  did not occur anymore. Thus,  $E_k$  is in the neighborhood of  $E_s$ . But then, by line [5](#) of the  $\text{RESAMPLE}$  routine,  $\text{RESAMPLE}(E_s)$  couldn't have terminated and thus, neither could the main call. Contradiction.

Thus  $E_k = E_j$ . Since under the assignment at the beginning of the main call,  $E_j$  occurred, by lines [2](#) and [3](#) of the  $\text{RESAMPLE}$  routine, it must be the case that during some resampling of the variables in  $\text{sc}(E_j)$ ,  $E_j$  became non-occurring. The main call could not have ended after this resampling, since  $E_j$  occurs under the assignment  $\mathbf{b}$  produced at the end of this call. Then, there exists some  $r \in N_j$  such that  $\text{RESAMPLE}(E_r)$  is the subsequent  $\text{RESAMPLE}$  call. Thus  $E_j \in \mathcal{S}_r$  and we obtain a contradiction as in the case where  $E_k \in \mathcal{S}_j$  above.  $\square$

An immediate corollary of Lemma [6.2.1](#), is that the events of the root calls of  $\text{RESAMPLE}$  are pairwise distinct, therefore there can be *at most*  $m$  such root calls in any execution of  $\text{DD-ALG}$ .

Feasible forests, along with the event  $W_{\mathcal{F}}$ , are defined as in Subsec. [6.1.2](#). As it happened with  $\text{VDL-ALG}$ ,  $\text{DD-ALG}$  also introduces various dependencies that render the probabilistic calculations essentially impossible. For example, suppose that the  $i$ -th node of a witness forest  $\mathcal{F}$  is labeled by  $E_j$  and its children have labels with indices in  $N_j$ . Then, under the assignment produced at the end of the  $i$ -th round of this execution,  $E_j$  does not occur.

We will use algorithm [2](#), introduced as  $\text{VALALG}$  in Subsec. [6.1.3](#). Recall that  $V_{\mathcal{F}}$  is the event that  $\text{VALALG}$  is successful on input  $\mathcal{F}$ . Lemma [6.1.3](#) again holds for the same reasons, where  $P_n$  now denotes the probability that  $\text{DD-ALG}$  lasts for at least  $n$  steps.

From now on, we will use the following notation:  $\mathbf{n} = \{n_1, \dots, m\}$ , where  $n_1, \dots, n_m \geq 0$  are such that  $\sum_{i=1}^m n_i = n$  and  $\mathbf{n} - (1)_j := (n_1, \dots, n_j - 1, \dots, n_m)$ . In what follows, we prove the following version of Th. [3.3.5](#):

**Theorem 6.2.1** (Algorithmic directed LLL [3.3.5](#)). *Suppose that there exist  $\chi_1, \chi_2, \dots, \chi_m \in (0, 1)$ , such that*

$$\Pr(E_j) \leq \chi_j \prod_{i \in N_j} (1 - \chi_i),$$

for all  $j \in \{1, \dots, m\}$ , where  $N_j$  denotes the outwards neighborhood of  $E_j$  in

the  $d$ -dependency graph. Then, the probability that DD-ALG executes for at least  $n$  rounds is inverse exponential in  $n$ .

*Proof.* We may assume, without loss of generality, that

$$\Pr[E_j] < \chi_j \prod_{i \in N_j} (1 - \chi_i), \text{ for all } j \in \{1, \dots, m\},$$

i.e. that the hypothesis is given in terms of a strict inequality. Indeed, we can otherwise consider an event  $B$ , such that  $B$  and  $E_1, \dots, E_m$ , are mutually independent, where  $\Pr[B] = 1 - \delta$ , for arbitrary small  $\delta > 0$ . We can now perturb the events a little, by considering e.g.  $E_j \cap B$ ,  $j = 1, \dots, m$ . As a consequence, we can also assume without loss of generality that for some other small enough  $\epsilon > 0$ , we have that  $\Pr[E_j] \leq (1 - \epsilon)\chi_j \prod_{i \in N_j} (1 - \chi_i)$ .

By Lemma [6.1.3](#), it suffices to prove that  $\hat{P}_n$  is inverse exponential in  $n$ . Specifically, we show that  $\hat{P}_n \leq (1 - \epsilon)^n$ .

## 6.2.2 Recurrence for dd-Alg

Let  $Q_{\mathbf{n},j}$  be the probability that VALALG is successful when started on a tree whose root is labeled with  $E_j$  and has  $\sum_{i=1}^m n_i = n$  nodes labeled with  $E_1, \dots, E_m$ . Observe that to obtain a bound for  $\hat{P}_n$  we need to add over all possible forests with  $n$  nodes in total. Thus, it holds that:

$$\hat{P}_n \leq \sum_{\mathbf{n}} \sum_{\mathbf{n}^1 + \dots + \mathbf{n}^m = \mathbf{n}} \left( Q_{\mathbf{n}^1,1} \cdots Q_{\mathbf{n}^m,m} \right).$$

Our aim is to show that  $Q_{\mathbf{n},j}$  is exponentially small to  $n$ , for any given sequence of  $\mathbf{n}$  and any  $j \in \{1, \dots, m\}$ . Thus, by ignoring polynomial in  $n$  factors, the same will hold for  $\hat{P}_n$  (recall that the number of variables and the number of events are considered constants, asymptotics are in terms of the number of steps  $n$  only).

Let  $N_j^+ := N_j \cup \{j\}$ , and assume that, for each  $j \in \{1, \dots, m\}$ ,  $|N_j^+| = k_j$ . Observe now that  $Q_{\mathbf{n},j}$  is bounded from above by a function, denoted again by  $Q_{\mathbf{n},j}$  (to avoid overloading the notation), which follows the recurrence:

$$Q_{\mathbf{n},j} = \Pr[E_j] \cdot \sum_{\mathbf{n}^1 + \dots + \mathbf{n}^{k_j} = \mathbf{n} - (1)_j} \left( Q_{\mathbf{n}^1, j_1} + \cdots + Q_{\mathbf{n}^{k_j}, j_{k_j}} \right), \quad (6.8)$$



with initial conditions  $Q_{\mathbf{n},j} = 0$  when  $n_j = 0$  and there exists an  $i \neq j$  such that  $n_i \geq 1$ ; and with  $Q_{\mathbf{0},j} = 1$ , where  $\mathbf{0}$  is a sequence of  $m$  zeroes.

To solve the above recurrence, we introduce, for  $j = 1, \dots, m$ , the *multi-variate generating functions*:

$$Q_j(\mathbf{t}) = \sum_{\mathbf{n}:n_j \geq 1} Q_{\mathbf{n},j} \mathbf{t}^{\mathbf{n}}, \quad (6.9)$$

where  $\mathbf{t} = (t_1, \dots, t_m)$ ,  $\mathbf{t}^{\mathbf{n}} := t_1^{n_1} \dots t_m^{n_m}$ .

By multiplying both sides of (6.8) by  $\mathbf{t}^{\mathbf{n}}$  and adding all over suitable  $\mathbf{n}$ , we get the system of equations  $\mathbf{Q}$ :

$$Q_j(\mathbf{t}) = t_j f_j(\mathbf{Q}), \quad (6.10)$$

where, for  $\mathbf{x} = (x_1, \dots, x_m)$  and  $j = 1, \dots, m$ :

$$f_j(\mathbf{x}) = (1 - \epsilon) \cdot \chi_j \cdot \left( \prod_{i \in N_j} (1 - \chi_i) \right) \cdot \left( \prod_{i \in N_j^+} (x_i + 1) \right). \quad (6.11)$$

To solve the system, we will directly use the result of Bender and Richmond in [19] (Theorem 2). Let  $g$  be any  $m$ -ary projection function on some of the  $m$  coordinates. In the sequel we take  $g := pr_s^m$ , the  $(m)$ -ary projection on the  $s$ -th coordinate. Let also  $\mathcal{B}$  be the set of trees  $B = (V(B), E(B))$  whose vertex set is  $\{0, 1, \dots, m\}$  and with edges directed towards 0. By [19], we get:

$$[\mathbf{t}^{\mathbf{n}}]g((\mathbf{Q})(\mathbf{t})) = \frac{1}{\prod_{j=1}^m n_j} \sum_{B \in \mathcal{B}} [\mathbf{x}^{\mathbf{n}-1}] \frac{\partial(g, f_1^{n_1}, \dots, f_m^{n_m})}{\partial B}, \quad (6.12)$$

where the term for a tree  $B \in \mathcal{B}$  is defined as:

$$[\mathbf{x}^{\mathbf{n}-1}] \prod_{r \in V(B)} \left\{ \left( \prod_{(i,r) \in E(B)} \frac{\partial}{\partial x_i} \right) f_r^{n_r}(\mathbf{x}) \right\}, \quad (6.13)$$

where  $r \in \{0, \dots, m\}$  and  $f_0^{n_0} := g$ .

We consider a tree  $B \in \mathcal{B}$  such that (6.13) is not equal to 0. Thus,  $(i, 0) \notin E(B)$ , for all  $i \neq s$ . On the other hand,  $(s, 0) \in E(B)$ , lest vertex 0 is isolated, and each vertex has out-degree *exactly* one, lest a cycle is formed or connectivity is broken. From vertex 0, we get  $\frac{\partial pr_s^m(\mathbf{x})}{\partial x_s} = 1$ . Since our aim is to prove that  $\hat{P}_n$  is exponentially small in  $n$ , we are interested only

in factors of (6.13) that are exponential in  $n$ , and we can thus ignore the derivatives (except the one for vertex 0), as they introduce only polynomial (in  $n$ ) factors to the product. Thus, we have that (6.13) is equal to the coefficient of  $\mathbf{x}^{\mathbf{n}-1}$  in:

$$\prod_{j=1}^m \left\{ (1 - \epsilon)^{n_j} \cdot \chi_j^{n_j} \cdot \left( \prod_{i \in N_j} (1 - \chi_i)^{n_j} \right) \cdot \left( \prod_{i \in N_j^+} (x_i + 1)^{n_j} \right) \right\}. \quad (6.14)$$

We now group the factors of (6.14) according to the  $i$ 's. We have already argued each vertex  $i$  has out-degree 1. Thus, the exponent of the term  $x_i + 1$  is  $n_i + \sum_{j:i \in N_j} n_j$  and the product of (6.14) is equal to:

$$\prod_{i=1}^m \left\{ (1 - \epsilon)^{n_i} \cdot \chi_i^{n_i} \cdot (1 - \chi_i)^{\sum_{j:i \in N_j} n_j} \cdot (x_i + 1)^{n_i + \sum_{j:i \in N_j} n_j} \right\}. \quad (6.15)$$

Using the binomial theorem and by ignoring polynomial factors, we get that the coefficient of  $\mathbf{x}^{\mathbf{n}-1}$  in (6.15) is:

$$\prod_{i=1}^m \left\{ (1 - \epsilon)^{n_i} \cdot \chi_i^{n_i} \cdot (1 - \chi_i)^{\sum_{j:i \in N_j} n_j} \cdot \binom{n_i + \sum_{j:i \in N_j} n_j}{n_i} \right\}. \quad (6.16)$$

By expanding  $(\chi_i + 1 - \chi_i)^{n_i + \sum_{j:i \in N_j} n_j}$ , we get that (6.16) is at most:

$$\prod_{i=1}^m (1 - \epsilon)^{n_i} = (1 - \epsilon)^{\sum_{i=1}^m n_i} = (1 - \epsilon)^n. \quad (6.17)$$

Thus,  $\hat{P}_n$  is inverse exponential in  $n$ . □

From Th. 6.2.1, the existential Th. 3.3.5 immediately follows.

### 6.3 Shearer's lemma

We now turn our attention to Shearer's lemma. The first algorithmic proof for general probability spaces was given by Kolipaka and Szegedy [150]. Harvey and Vondrák [121] proved a version of the lemma for their lopsided association graphs (again in the generalized framework).

Here, we apply it to the underlying *undirected* graph of the  $d$ -dependency graph we introduced in Section 3.1. Note that as Harris [118] and He et al. [123], Shearer's lemma is tight to the level of generality to which it applies. Since our work is situated in the variable framework, the lopsided version of the lemma we provide is only a necessary condition for the existence of a solution. As in the previous sections, we give a forward argument that directly leads to an exponentially small bound of the probability of the algorithm lasting for at least  $n$  steps.

First, we provide some terminology and the main algorithm that finds the sought after point in the probability space (Subsec. 6.3.1). Then, we design two new validation algorithms and show how their executions are related probabilistically with that of the main algorithm (Subsec. 6.3.2). Finally, we bound the probability of the validation algorithm to succeed by a recurrence relation which we subsequently solve (Subsec. 6.3.3).

### 6.3.1 The MaxSetRes algorithm

Let  $E_1, \dots, E_m$  be events, whose vector of probabilities is  $\mathbf{p} = (p_1, \dots, p_m)$ , that is  $Pr[E_j] = p_j \in (0, 1)$ ,  $j = 1, \dots, m$ . Recall that  $G^{sd}$  is a graph on  $m$  vertices, where we associate each event  $E_j$  with vertex  $j$ ,  $j = 1, \dots, m$  and where

$$E = \{\{i, j\} \mid \text{either } E_j \text{ is } d\text{-dependent on } E_i \text{ or } E_i \text{ is } d\text{-dependent on } E_j\}.$$

A subset  $I \subseteq \{1, \dots, m\}$  of the graph's vertices is an *independent set* if there are no edges between its vertices. Abusing the notation, we will sometimes say that an independent set  $I$  contains events (instead of indices of events). Let  $I(G)$  denote the set of independent sets of  $G$ . For any  $I \in I(G)$ , let  $N(I) := \bigcup_{j \in I} N_j$  be the set of neighbors of the vertices of  $I$ . Following Kolipaka and Szegedy [150], we say that  $I$  *covers*  $J$  if  $J \subseteq I \cup N(I)$ .

A *multiset* is usually represented as a couple  $(A, f)$ , where  $A$  is a set, called the *underlying set*, and  $f : A \mapsto \mathbb{N}_{\geq 1}$  is a function, with  $f(x)$  denoting the multiplicity of  $x$ , for all  $x \in A$ . In our case, the underlying sets of multisets are always subsets of  $\{1, \dots, m\}$ . Thus, to make notation easier to follow, we use couples  $(I, \mathbf{z})$ , where  $I \subseteq \{1, \dots, m\}$  and  $\mathbf{z} = (z_1, \dots, z_m)$  is an  $m$ -ary vector, with  $z_j \in \mathbb{N}$  denoting the multiplicity of  $E_j$  in  $I$ . Note that  $z_j = 0$  if and only if  $j \notin I$ ,  $j = 1, \dots, m$ .

**Algorithm 4** MAXSETRES.

---

```

1: Sample the variables  $X_i$ ,  $i = 1, \dots, l$  and let  $\mathbf{a}$  be the resulting assignment.
2:  $t := 1$ ,  $I_t := \emptyset$ ,  $\bar{z}^t := (0, \dots, 0)$ .
3: repeat
4:   while there exists an event  $E_j \notin I_t \cup N(I_t)$  that occurs under the
      current assignment,
      let  $E_j$  be the least indexed such event and do
5:      $I_t := I_t \cup \{j\}$ ,  $c := 0$ ,  $z_j^t := 1$ .
6:     while  $E_j$  occurs do
7:       Resample the variables in  $sc(E_j)$ .
8:        $c := c + 1$ .
9:     end while.
10:    if there exists an occurring event that is not in  $I_t \cup N(I_t)$  then
11:       $z_j^t := c + 1$ .
12:    else if there exists an occurring event not in  $N_j$  then
13:       $t := t + 1$ ,  $I_t := \emptyset$ .
14:       $z_j^t := c$ .
15:    else
16:      for  $s = 1, \dots, c$  do
17:         $I_{t+s} := \{E_j\}$ .
18:      end for
19:       $t := t + c + 1$ ,  $I_t := \emptyset$ .
20:    end if
21:  end while
22: until  $I_t = \emptyset$ .
23: Output current assignment  $\mathbf{a}$ .

```

---

Our aim is to algorithmically prove Th. 3.3.6 for  $G^{sd}$ . It is straightforward to observe that the proof works for  $G^{MT}$  and  $G^s$  too, but in those cases it could be substantially simplified. The algorithm we use is a variation of the MAXIMAL SET RESAMPLE algorithm, designed by Harvey and Vondrák in [121], which is a slowed down version of the algorithm in [150]. The algorithm constructs multisets whose underlying sets are independent sets of  $G$ , by selecting occurring events that it resamples until they do not occur anymore.

A *step* of MAXSETRES is a single resampling of the variables of an event in line 7, whereas a *phase* is an iteration of **repeat** at lines 3–22, except from

the last iteration that starts and ends with  $I_t = \emptyset$ . Phases are not nested. During each phase, there are at most  $m$  repetitions of the **while**-loop of lines [6-9](#), where  $m$  is the number of events (recall that the number of variables  $n$ , and the number of events  $m$  are considered to be constants). During each phase, a multiset  $(I_t, z^t)$  is created, where the underlying set  $I_t$  is an independent set of  $G$ . There are two scenarios that can happen at the end of a phase. The first is by line [13](#), where MAXSETRES creates a new independent set, containing  $c$  copies of the last event it resampled at lines [6-9](#). The second is by lines [16-19](#), where MAXSETRES proceeds  $c$  phases at once, creating  $c$  singleton sets, containing only the event it lastly resampled at lines [6-9](#). Lines [10-20](#) exist for technical reasons that will become apparent later.

Note that, by lines [4](#) and [22](#), if and when MAXSETRES terminates, it produces an assignment of values under which none of the events occurs.

We now proceed with some results concerning the execution of MAXSETRES. Note that it refers to the underlying independent sets of the multisets created at each phase.

**Lemma 6.3.1.**  $I_t$  covers  $I_{t+1}$ , for all  $t \in \{1, \dots, n-1\}$ .

*Proof.* Let  $E_j$  be an event in  $I_{t+1}$ . Then, at some point during phase  $t+1$ ,  $E_j$  was occurring. We will prove below that  $E_j$  occurs also at the beginning of phase  $t+1$ . This will conclude the proof, since if  $E_j \notin N(I_t) \cup I_t$ , then at the moment when phase  $I_{t+1}$  was to start, the algorithm instead of starting  $I_{t+1}$  would opt to add  $E_j$  to  $I_t$ , a contradiction.

Assume that  $I_{t+1} \neq \{E_j\}$ , lest we have nothing to prove. To prove that  $E_j$  occurs at the beginning of phase  $t+1$ , assume towards a contradiction that it does not. Then it must have become occurring during the repeated resamplings of an event  $E_r$  introduced into  $I_{t+1}$ .

Therefore, under the assignment when  $E_r$  was selected,  $E_r$  occurred and  $E_j$  did not. Furthermore, during the repeated resamplings of  $E_r$ , only variables in  $\text{sc}(E_r)$  had their values changed, and under the assignment at the end of these resamplings,  $E_r$  ceases occurring and  $E_j$  occurs. By Definition [3.2.8](#),  $E_j$  is  $d$ -dependent on  $E_r$  and thus  $E_j \in N(I_{t+1})$ . By lines [4](#), [10](#) and [12](#),  $E_j$  could not have been selected at any point during round  $t+1$ . This concludes the proof.  $\square$

Consider the following definition:

**Definition 6.3.1** (Kolipaka and Szegedy [150]). A stable sequence of events is a sequence of non-empty independent sets  $\mathcal{I} = I_1, \dots, I_n$  such that  $I_t$  covers  $I_{t+1}$ , for all  $t \in \{1, \dots, n-1\}$ .

Stable sequences play the role of witness structures in the present framework. By Lemma 6.3.1, the underlying sets of the sequence of multisets MAXSETRES produces in an execution that lasts for at least  $n$  phases, is a stable sequence of length  $n$ .

We now prove:

**Theorem 6.3.1** (Algorithmic Shearer's lemma 3.3.6 for  $G^{ds}$ ). If it holds that for all  $I \in I(G)$ :

$$q_I(G, \mathbf{p}) = \sum_{J \in I(G): I \subseteq J} (-1)^{|J \setminus I|} \prod_{j \in J} p_j > 0,$$

then the probability  $\Pi_n$  that MAXSETRES lasts for at least  $n$  phases is exponentially small, i.e. for some constant  $c < 1$ ,  $\Pi_n$  is at most  $c^n$ , ignoring polynomial factors.

Again, Th. 3.3.6 follows immediately from Th. 6.3.1.

*Proof.* Let  $\mathbf{z} = (\mathbf{z}^1, \dots, \mathbf{z}^n)$  be an  $(n \times m)$ -matrix, whose rows are  $m$ -ary vectors  $\mathbf{z}^t = (z_1^t, \dots, z_m^t)$  of non-negative integers,  $t = 1, \dots, n$ . Let also

$$(\mathcal{I}, \mathbf{z}) = (I_1, \mathbf{z}^1), \dots, (I_n, \mathbf{z}^n)$$

be a sequence of multisets, whose underlying sequence  $\mathcal{I}$  is a stable sequence. We denote by  $|(\mathcal{I}, \mathbf{z})|$  its length, i.e. the number of pairs  $(\mathcal{I}_t, \mathbf{z}^t)$  it contains. If  $\Pi(\mathcal{I}, \mathbf{z})$  is the probability that an execution of MAXSETRES produced  $(\mathcal{I}, \mathbf{z})$  (which can be zero), it is easy to see that:

$$\Pi_n = \sum_{(\mathcal{I}, \mathbf{z}): |(\mathcal{I}, \mathbf{z})|=n} \Pi(\mathcal{I}, \mathbf{z}), \quad (6.18)$$

where the sum is over all possible pairs of stable sequences  $\mathcal{I}$  of length  $n$  and vectors  $\mathbf{z}$ .

### 6.3.2 The validation algorithms

To bound the rhs of (6.18), consider the validation algorithm `MAXSETVAL` below. `MAXSETVAL`, on input a stable sequence  $\mathcal{I} = I_1, \dots, I_n$ , proceeds to check each event contained in each independent set. If this event does not occur, it fails; else it resamples the variables in its scope. Note that the success or failure of this algorithm has nothing to do with finding an assignment such that none of the events occur.

---

**Algorithm 5** `MAXSETVAL`.

---

**Input:** Stable sequence  $\mathcal{I} = I_1, \dots, I_n$ .

- 1: Sample the variables  $X_i, i = 1, \dots, l$ .
- 2: **for**  $t=1, \dots, l$  **do**
- 3:     **for** each event  $E_j$  of  $I_t$  **do**
- 4:         **if**  $E_j$  does not occur under the current assignment **then**
- 5:             **return failure** and exit.
- 6:         **else**
- 7:             Resample the variables in  $\text{sc}(E_j)$
- 8:         **end if**
- 9:     **end for**
- 10: **end for**
- 11: **return success.**

---

A phase of `MAXSETVAL` is any repetition of lines 2–10. Let  $\hat{\Pi}(\mathcal{I})$  be the probability that `MAXSETVAL` is successful on input  $\mathcal{I}$  and:

$$\hat{\Pi}_n := \sum_{\mathcal{I}:|\mathcal{I}|=n} \hat{\Pi}(\mathcal{I}). \quad (6.19)$$

To obtain our result, we now proceed to show: (i) that  $\Pi_n \leq \hat{\Pi}_n$  and (ii) that  $\hat{\Pi}_n$  is inverse exponential to  $n$ . For the former, consider the validation algorithm `MULTISETVAL`, algorithm 6, below.

`MULTISETVAL` takes as input a sequence  $(\mathcal{I}, \mathbf{z})$  of multisets whose underlying sequence is stable. It then proceeds, for each multiset, to check if its events occur under the current assignment it produces. If not it fails, else it proceeds. When the last copy of an event inside a multiset, apart from the last event, is resampled, it checks if that event still occurs (line 11). If it does, the algorithm fails. If it manages to go through the whole sequence without

**Algorithm 6** MULTISSETVAL.

---

**Input:**  $(\mathcal{I}, \mathbf{z}) = (I_1, z^1), \dots, (I_n, z^n)$ ,  $I_t = \{E_{t_1}, \dots, E_{t_{k_t}}\}$ ,  $t = 1, \dots, n$ .

- 1: Sample the variables  $X_i$ ,  $i = 1, \dots, l$ .
- 2: **for**  $t = 1, \dots, n$  **do**
- 3:     **for**  $s = 1, \dots, k_t - 1$  **do**
- 4:         **for**  $r = 1, \dots, z_{t_s}^t$  **do**
- 5:             **if**  $E_{t_s}$  does not occur under the current assignment **then**
- 6:                 **return failure** and exit.
- 7:             **else**
- 8:                 Resample the variables in  $\text{sc}(E_{t_s})$
- 9:             **end if**
- 10:         **end for**
- 11:         **if**  $E_{t_s}$  occurs under the current assignment **then**
- 12:             **return failure** and exit.
- 13:         **end if**
- 14:     **end for**
- 15:     **if**  $E_{t_{k_t}}$  does not occur under the current assignment **then**
- 16:         **return failure** and exit.
- 17:     **else**
- 18:         Resample the variables in  $\text{sc}(E_{t_{k_t}})$
- 19:     **end if**
- 20: **end for**
- 21: **return success.**

---

failing, it succeeds. Note again that the success or failure of MULTISSETVAL has nothing to do with obtaining an assignment such that none of the events holds.

We call a *phase* of MULTISSETVAL each repetition of lines [2–20](#). Let also  $\tilde{P}(\mathcal{I}, \mathbf{z})$  be the probability that MULTISSETVAL succeeds on input  $(\mathcal{I}, \mathbf{z})$ . We prove two lemmas concerning MULTISSETVAL.

**Lemma 6.3.2.** *For each sequence  $(\mathcal{I}, \mathbf{z})$ ,  $\Pi(\mathcal{I}, \mathbf{z}) \leq \tilde{P}(\mathcal{I}, \mathbf{z})$ . Thus:*

$$\Pi_n \leq \sum_{(\mathcal{I}, \mathbf{z}) : |(\mathcal{I}, \mathbf{z})| = n} \tilde{P}(\mathcal{I}, \mathbf{z}) \quad (6.20)$$

*Proof.* It suffices to prove the first inequality, as the result is then derived by [\(6.18\)](#). Note that the last event in every multiset that MAXSETRES



produces always has multiplicity 1 and that, furthermore, it is not required to be non-occurring after resampling it, in contrast with all the other events in the multiset. It is now straightforward to notice that if `MULTISETVAL` makes the same random choices as `MAXSETRES` did when it created any sequence  $(\mathcal{I}, \mathbf{z})$ , `MULTISETVAL` will succeed on input  $(\mathcal{I}, \mathbf{z})$ .  $\square$

**Lemma 6.3.3.** *For any  $(\mathcal{I}, \mathbf{z})$ , it holds that:*

$$\sum_{(\mathcal{I}, \mathbf{z}): |(\mathcal{I}, \mathbf{z})|=N} \tilde{\mathbb{P}}(\mathcal{I}, \mathbf{z}) = \sum_{\mathcal{I}: |\mathcal{I}|=n} \hat{\mathbb{P}}(\mathcal{I}). \quad (6.21)$$

*Proof.* We will rearrange the sum in the lhs of (6.21). Assume that the stable sequences of length  $n$  in  $G$  are arbitrarily ordered as  $\mathcal{I}_1, \dots, \mathcal{I}_s$ . Then, it holds that:

$$\sum_{(\mathcal{I}, \mathbf{z}): |(\mathcal{I}, \mathbf{z})|=n} \tilde{\mathbb{P}}(\mathcal{I}, \mathbf{z}) = \sum_{\mathbf{z}=(\bar{z}^1, \dots, \bar{z}^n)} \tilde{\mathbb{P}}(\mathcal{I}_1, \mathbf{z}) + \dots + \sum_{\mathbf{z}=(\bar{z}^1, \dots, \bar{z}^n)} \tilde{\mathbb{P}}(\mathcal{I}_s, \mathbf{z}). \quad (6.22)$$

Let  $\mathcal{I} = (I_1, \dots, I_n)$  be a stable sequence and consider the term

$$\sum_{\mathbf{z}=(\bar{z}^1, \dots, \bar{z}^n)} \tilde{\mathbb{P}}(\mathcal{I}, \mathbf{z})$$

of (6.22) corresponding to  $\mathcal{I}$ . It suffices to show that is equal to  $\hat{\mathbb{P}}(\mathcal{I})$ .

Assume again that  $I_t = \{E_{t_1}, \dots, E_{t_{k_t}}\}$  and that  $\bar{z}^t = (z_1^t, \dots, z_m^t)$ , where  $z_j^t \geq 0$ ,  $j = 1, \dots, m$ ,  $t = 1, \dots, n$ . Finally, set:

$$\begin{aligned} \tilde{\mathbb{P}}(I_t, \bar{z}^t) := & \Pr[E_{t_1}]^{z_1^t} \Pr[\bar{E}_{t_1} \cap E_{t_2}] \Pr[E_{t_2}]^{z_2^t-1} \dots \\ & \Pr[E_{t_{k_t-1}}]^{z_{k_t-1}^t-1} \Pr[\bar{E}_{t_{k_t-1}} \cap E_{t_{k_t}}]. \end{aligned}$$

Then, it holds that:

$$\sum_{\mathbf{z}=(\bar{z}^1, \dots, \bar{z}^n)} \tilde{\mathbb{P}}(\mathcal{I}, \mathbf{z}) = \sum_{\mathbf{z}=(z^1, \dots, z^n)} \prod_{t=1}^n \tilde{\mathbb{P}}(I_t, z^t). \quad (6.23)$$

By Lemma 3.2.6, it holds that all the factors  $\Pr[\bar{E} \cap E']$  that appear in (6.23) are less or equal than  $\Pr[\bar{E}] \cdot \Pr[E']$ . Now, by factoring out:

$$\hat{\mathbb{P}}(\mathcal{I}) = \prod_{t=1}^n \left( \Pr[E_{t_1}] \cdots \Pr[E_{t_{k_t}}] \right)$$

from the rhs of (6.23) and by rearranging the terms according to the sets  $\mathcal{I}_t$ , we get:

$$\sum_{\mathbf{z}=(z^1,\dots,z^n)} \tilde{\hat{P}}(\mathcal{I}, \mathbf{z}) = \hat{P}(\mathcal{I}) \cdot \prod_{t=1}^n \left( \sum_{z^t=(z_{k_t}^t, \dots, z_{k_t}^t)} \Pr[E_{t_1}]^{z_{k_t}^t-1} (1 - \Pr[E_{t_1}]) \cdots \Pr[E_{t_{k_t-1}}]^{z_{k_t-1}^t-1} (1 - \Pr[E_{t_{k_t-1}}]) \right). \quad (6.24)$$

The proof is now complete, by noticing that all the factors, except from  $\hat{P}(\mathcal{I})$  in the rhs of (6.24) are equal to 1.  $\square$

Thus, by (6.20), (6.21) and (6.19), we get:

$$\Pi_n \leq \hat{P}_n. \quad (6.25)$$

### 6.3.3 The stable set matrix

What remains is to show that  $\hat{P}_n$  is inverse exponential to  $n$ . Towards this, for  $n \geq 1$  let

$$\hat{P}_{n,I} = \sum_{\substack{\mathcal{I}:|\mathcal{I}|=n \\ \mathcal{I}_1=I}} \hat{P}(\mathcal{I}), \quad (6.26)$$

where  $\mathcal{I}_1$  is its first term of  $\mathcal{I}$ .

Observe now that, for any independent set  $I$ ,  $\hat{P}_{1,I} = \prod_{j \in I} p_j$ . Thus we obtain the following recursion:

$$\hat{P}_{N+1,I} = \begin{cases} \prod_{j \in I} p_j \left( \sum_{J:I \text{ covers } J} \hat{P}_{N,J} \right) & \text{if } N \geq 1, \\ \prod_{j \in I} p_j & \text{if } N = 0. \end{cases} \quad (6.27)$$

If the class of all non-empty independent sets is  $\{I_1, \dots, I_s\}$ , following again the terminology of [150], we define the *stable set matrix*  $M$ , as an  $s \times s$  matrix, whose element in the  $i$ -th row and  $j$ -th column is  $\prod_{j \in I} p_j$  if  $I$  covers  $J$  and 0 otherwise. Furthermore, let  $q_n = (\hat{P}_{n,I_1}, \dots, \hat{P}_{n,I_s})$ . Easily, (6.27) is equivalent to:

$$q_n = M q_{n-1},$$

thus

$$q_n = M^{n-1} q_1. \quad (6.28)$$

Let  $\|\cdot\|_1$  be the 1-norm defined on  $\mathbb{R}^s$ . It is known that any vector norm, and thus 1-norm too, yields a norm for square matrices called the *induced norm* [126] as follows:

$$\|M\|_1 := \sup_{x \neq 0} \frac{\|Mx\|_1}{\|x\|_1} \geq \frac{\|Mq_1\|_1}{\|q_1\|_1}. \quad (6.29)$$

By (6.28) and (6.29), we have that:

$$\|q_n\|_1 = \|M^{n-1}q_1\|_1 \leq \|M^{n-1}\|_1 \cdot \|q_1\|_1. \quad (6.30)$$

Note now that:

$$\hat{P}_n \leq \sum_{i=1}^s \hat{P}_{n,I_i} = \|q_n\|_1 = \|M^{n-1}\|_1 \|q_1\|_1. \quad (6.31)$$

Since  $\|q_1\|_1$  is a constant, it suffices to show that  $\|M^{n-1}\|_1$  is exponentially small in  $n$ . Let  $\rho(M)$  be the *spectral radius* of  $M$  [126], that is:

$$\rho(A) := \max\{|\lambda| \mid \lambda \text{ is an eigenvalue of } A\}.$$

By Gelfand's formula (see again [126]) used for the induced matrix norm  $\|\cdot\|_1$ , we have that:

$$\rho(M) = \lim_{n \rightarrow \infty} \|M^n\|_1^{1/n}. \quad (6.32)$$

Furthermore, in [150] (Theorem 14), it is proved that the following are *equivalent*:

1. For all  $I \in I(G) : q_I(G, \bar{p}) > 0$ .
2.  $\rho(M) < 1$ .

Using (1  $\Rightarrow$  2) we can select an  $\epsilon > 0$  such that  $\rho(M) + \epsilon < 1$ . Then, by (6.32), we have that there exists a  $n_0$  (depending only on  $\epsilon, M$ ) such that, for  $n \geq n_0$ :  $\|M^{n-1}\|_1 \leq (\rho(M) + \epsilon)^{n-1}$ , which, together with (6.31), gives us that  $\hat{P}_n$  is exponentially small in  $n$ .

Thus, by the analysis above, we get that there is a constant  $c < 1$  (depending on  $\|q_1\|, p$  and  $\rho(M) + \epsilon$ ) such that  $\Pi_n \leq c^n$ , for  $n \geq n_0$  and by ignoring polynomial factors. This concludes the proof.  $\square$

## 6.4 A new bound on Acyclic Edge Coloring

In this section we show that  $2\Delta - 1$  colors suffice to acyclically color the edges of a graph with maximum degree  $\Delta$ . In Subsec. [6.4.1](#) we present our main algorithm and in Subsec. [6.4.2](#) we analyze its execution time. Then, in Subsec. [6.4.3](#) we provide the necessary validation algorithm to make our probabilistic analysis feasible and, finally, in Subsec. [6.4.4](#), we provide and solve a recurrence relation that bounds the number of phases of our main algorithms.

### 6.4.1 Main Algorithm

We first present below algorithm `EDGECOLOR`.

---

#### Algorithm 7 `EDGECOLOR`

---

- 1: **for** each  $e \in E$  **do**
- 2:     Choose a color for  $e$  from the palette, independently for each  $e$ , and u.a.r. (not caring for properness)
- 3: **end for**
- 4: **while** there is an edge contained in a cycle of even length  $\geq 6$  and having homochromatic edges of the same parity, let  $e$  be the least such edge and  $C$  be the least such cycle and **do**
- 5:     `RECOLOR`( $e, C$ )
- 6: **end while**
- 7: **return** the current coloring

`RECOLOR`( $e, C$ ), where  $C = C(e) = \{e = e_1^C, \dots, e_{2k}^C\}$ ,  $k \geq 3$ .

- 1: **for**  $i = 1, \dots, 2k - 2$  **do**
  - 2:     Choose a color for each  $e_i^C$  independently and u.a.r. (not caring for properness)
  - 3: **end for**
  - 4: **while** there is an edge in  $\{e_1^C, \dots, e_{2k-2}^C\}$  contained in a cycle of even length  $\geq 6$  and having homochromatic edges of the same parity, let  $e'$  be the least such edge and  $C'$  the least such cycle and **do**
  - 5:     `RECOLOR`( $e', C'$ )
  - 6: **end while**
- 

Notice that `EDGECOLOR` may not halt, and perhaps worse, even if it stops, it may generate a non-proper coloring or a coloring that is proper

but has bichromatic 4-cycles. However, it is obvious, because of the **while**-loops in the main part of `EDGECOLOR` and in the procedure `RECOLOR`, that if the algorithm halts, then it outputs a coloring with no cycles of even length  $\geq 6$  and having homochromatic edges of the same parity (all edges of such cycles might be homochromatic). So in the `MA` that follows, we repeat `EDGECOLOR` until the desired coloring is obtained.

---

**Algorithm 8** `MA`


---

- 1: **while** the color generated by `EDGECOLOR` is not proper or is proper but contains a bichromatic 4-cycle **do**
  - 2:     Execute `EDGECOLOR` anew
  - 3: **end while**
- 

Obviously `MA`, *if and when* it stops, generates a proper acyclic coloring. The rest of the paper is devoted to compute the probability distribution of the number of steps it takes.

A call of the `RECOLOR` procedure from line 5 of the algorithm `EDGECOLOR` is a *root* call of `RECOLOR`, while one made from within the execution of another `RECOLOR` procedure is called a *recursive* call. Each iteration of `RECOLOR` is called a *phase*. We also call phase, the initial one, the **for**-loop of `EDGECOLOR`. In the sequel, we count phases rather than steps of color assignments. Because the number  $m$  of the edges of the graph is constant, this does not affect the form of the asymptotics of the number of steps.

We prove the following progression lemma, which shows that at every time a `RECOLOR`( $e, C$ ) procedure terminates, some progress has indeed been made, which is then preserved in subsequent phases.

**Lemma 6.4.1.** *Consider an arbitrary call of `RECOLOR`( $e, C$ ) and let  $\mathcal{E}$  be the set of edges that at the beginning of the call are not contained in a cycle of even length  $\geq 6$  and having homochromatic edges of the same parity. Then, if and when that call terminates, no such edge in  $\mathcal{E} \cup \{e\}$  exists.*

*Proof.* Suppose that `RECOLOR`( $e, C$ ) terminates and there is an edge  $e' \in \mathcal{E} \cup \{e\}$  contained in a cycle of even length  $\geq 6$  and with homochromatic edges of the same parity. If  $e' = e$ , then by line 4, `RECOLOR`( $e, C$ ) could not have terminated. Thus,  $e' \in \mathcal{E}$ .

Since  $e'$  is not contained in a cycle as described at the beginning of `RECOLOR`( $e, C$ ), it must be the case that at some point during this call, some

cycle, with  $e'$  among its edges, turned into one having homochromatic edges of the same parity because of some call of RECOLOR. Consider the last time this happened and let  $\text{RECOLOR}(e^*, C^*)$  be the causing call. Then, there is some cycle  $C'$  of even length  $\geq 6$  and with  $e' \in C'$ , such that the recoloring of the edges of  $C^*$  resulted in  $C'$  having homochromatic edges of the same parity and staying such until the end of the  $\text{RECOLOR}(e, C)$  call. Then there is at least one edge  $e^*$  contained in both  $C^*$  and  $C'$  that was recolored  $\text{RECOLOR}(e^*, C^*)$ . By line 4 of  $\text{RECOLOR}(e^*, C^*)$ , this procedure could not have terminated, and thus neither could  $\text{RECOLOR}(e, C)$ , a contradiction.  $\square$

By Lemma 6.4.1, we get:

**Lemma 6.4.2.** *There are at most  $m$ , the number of edges of  $G$ , i.e. a constant, repetitions of the **while**-loop of the main part of `EDGECOLOR`.*

However, a **while**-loop of `RECOLOR` or `MA` could last infinitely long. In the next section we analyze the distribution of the number of steps they take.

## 6.4.2 Analysis of the Algorithm

In this section we will prove two things:

- The probability that `EDGECOLOR` lasts for at least  $n$  phases is inverse exponential in  $n$ .
- The probability that the **while**-loop of `MA` is repeated at least  $n$  times is inverse exponential in  $n$ .

From the above two facts, *yet to be proved*, Lemma 6.4.2, and because  $\epsilon$  in the number of colors  $\lceil (2 + \epsilon)(\Delta - 1) \rceil$  of the palette is an arbitrary positive constant, we get the Theorem below and its corollary, our main results. The proof of both the above will be possible by coupling the `EDGECOLOR` algorithm with a validation algorithm that does not have the stochastic dependencies of the color assignments of `EDGECOLOR` and thus is amenable to probabilistic analysis.

**Theorem 6.4.1.** *Assuming  $2\Delta - 1$  colors are available, the probability that `MA`, which if and when it stops produces a proper acyclic edge coloring, lasts for at least  $n^2$  steps is inverse exponential in  $n$ .*

Therefore:

**Corollary 6.4.1.**  $2\Delta - 1$  colors suffice to properly and acyclically color a graph.

We will proceed as follows: we first use a graph structure to depict the action of an execution of `EDGECOLOR` organized in phases. The goal is to present in a structured way any undesirable behavior of `EDGECOLOR` (and the corresponding validation algorithm), and thus get a bound on the probability of it happening. Finally, we will give still another algorithm, to show that can be coupled with `EDGECOLOR` whose probability we will bound.

We will depict the action of execution of `EDGECOLOR` organized in phases with a *rooted forest*, that is an acyclic graph whose connected components (*trees*) all have a designated vertex as their root. We label the vertices of such forests with pairs  $(e, C)$ , where  $e$  is an edge and  $C$  a  $2k$ -cycle containing  $e$ , for some  $k \geq 3$ . If a vertex  $u$  of  $\mathcal{F}$  is labeled by  $(e, C)$ , we will sometimes say that  $e$  is the *edge-label* and  $C$  the *cycle-label* of  $u$ . The number of nodes of a forest is denoted by  $|\mathcal{F}|$ .

**Definition 6.4.1.** A labeled rooted forest  $\mathcal{F}$  is called *feasible*, if the following two conditions hold:

- i. Let  $e$  and  $e'$  be the edge-labels of two distinct vertices  $u$  and  $v$  of  $G$ . Then, if  $u, v$  are both either roots of  $\mathcal{F}$  or siblings (i.e. they have a common parent) in  $\mathcal{F}$ , then  $e$  and  $e'$  are distinct.
- ii. If  $(e, C)$  is the label of a vertex  $u$  that is not a leaf, where  $C$  has half-length  $k \geq 3$ , and  $e'$  is the edge-label of a child  $v$  of  $u$ , then  $e' \in \{e_1^C, \dots, e_{2k-2}^C\}$ .

Given a feasible forest, we order its trees and the siblings of each node according to their edge-labels. By traversing  $\mathcal{F}$  in a depth-first fashion, respecting the ordering of trees and siblings, we obtain the *label sequence*  $\mathcal{L}(\mathcal{F}) = (e_1, C_1), \dots, (e_{|\mathcal{F}|}, C_{|\mathcal{F}|})$  of  $\mathcal{F}$ .

Given an execution of `EDGECOLOR` with at least  $n$  phases, we construct a feasible forest with  $n$  nodes by creating one node  $u$  labeled by  $(e, C)$  for each phase, corresponding to a call (root or recursive) of `RECOLOR`( $e, C$ ). We structure these nodes according to the order their labels appear in the *recursive stack* implementing `EDGECOLOR`: the children of a node  $u$  labeled by  $(e, C)$  correspond to the recursive calls of `RECOLOR` made by line 5 of

RECOLOR( $e, C$ ), with the leftmost child corresponding to the first such call and so on.

Let now  $P_n$  be the probability that EDGECOLOR lasts for at least  $n$  phases, and  $Q_n$  be the probability that EDGECOLOR (i) lasts for strictly less than  $n$  phases, and (ii) the coloring generated when it halts is either not proper or, alternatively, it is proper and has a bichromatic 4-cycle. We are now ready to compute upper bounds for these probabilities.

### 6.4.3 Validation Algorithm

We now give the validation algorithm:

---

#### Algorithm 9 COLORVAL( $\mathcal{F}$ )

---

Input:  $\mathcal{L}(\mathcal{F}) = (e_1, C_1), \dots, (e_{|\mathcal{F}|}, C_{|\mathcal{F}|}) : C_i(e_i) = \{e_i = e_1^{C_i}, \dots, e_{2k_i}^{C_i}\}$ .

- 1: Color the edges of  $G$ , independently and selecting for each a color u.a.r. from  $\{1, \dots, K\}$ .
- 2: **for**  $i = 1, \dots, |\mathcal{F}|$  **do**
- 3:     **if**  $C_i^0(e_i)$  and  $C_i^1(e_i)$  are both monochromatic (have one color) **then**
- 4:         Recolor  $e_1^{C_i}, \dots, e_{2k_i-2}^{C_i}$  by selecting independently colors u.a.r. from  $\{1, \dots, K\}$
- 5:     **else**
- 6:         Recolor  $e_1^{C_i}, \dots, e_{2k_i-2}^{C_i}$  by selecting independently colors u.a.r. from  $\{1, \dots, K\}$
- 7:     **return failure and exit**
- 8:     **end if**
- 9: **end for**
- 10: **return success**

---

We call each iteration of the **for**-loop of lines 2-9 a phase of COLORVAL( $\mathcal{F}$ ).

**Lemma 6.4.3.** *At the end of each phase of COLORVAL the colors are distributed as if they were assigned for a first and single time to each edge, selecting independently for each edge a color u.a.r. from the palette.*

*Proof.* This is because at each phase, the edges  $e_1^{C_i}, \dots, e_{2k_i-2}^{C_i}$ , the only edges for which information is obtained during the phase, are recolored selecting independently a color u.a.r. □



For a feasible forest  $\mathcal{F}$ , let  $V_{\mathcal{F}}$  be the event that  $\text{COLORVAL}(\mathcal{F})$  reports success. First, we compute the probability of  $V_{\mathcal{F}}$ .

**Lemma 6.4.4.** *Let  $\mathcal{F}$  be a feasible forest, with label sequence*

$$\mathcal{L}(\mathcal{F}) = (e_1, C_1), \dots, (e_{|\mathcal{F}|}, C_{|\mathcal{F}|}).$$

*Assume that  $C_i$  has half-length  $k_i \geq 3$ ,  $i = 1, \dots, n$ . Then:*

$$\Pr[V_{\mathcal{F}}] = \prod_{i=1}^{|\mathcal{F}|} \left( \frac{1}{K^{(2k_i-2)}} \right).$$

*Proof.* For each  $i$ , whatever the colors of  $e_{2k_i-1}^{C_i}$  and  $e_{2k_i}^{C_i}$  are, we need the other  $2k_i - 2$  edges of  $C$  to have the same color with one of them. The probability of this being true for each edge is, by Equation (3.14) and Lemma 6.4.3,  $\frac{1}{K}$ .  $\square$

We now let  $\text{COLORVAL}(\mathcal{F})$  be executed *independently* for all feasible forests with at most  $n$  nodes. We let  $\hat{P}_n$  be the probability that  $\text{COLORVAL}(\mathcal{F})$  succeeds for at least one  $\mathcal{F}$  with exactly  $n$  nodes, and let  $\hat{Q}_n$  be the probability that  $\text{COLORVAL}(\mathcal{F})$  succeeds for at least one  $\mathcal{F}$  such that (i)  $\mathcal{F}$  has strictly less than  $n$  nodes, and (ii) the coloring generated when  $\text{COLORVAL}(\mathcal{F})$  comes to an end is either not proper or, alternatively, it is proper and has a bichromatic 4-cycle.

**Lemma 6.4.5.** *We have that  $P_n \leq \hat{P}_n$  and  $Q_n \leq \hat{Q}_n$ .*

*Proof.* Consider an execution of  $\text{EDGECOLOR}$  and let  $\mathcal{F}$  be the feasible forest with  $n$  nodes generated in case the execution lasts for at least  $n$  phases, or be the forest generated until  $\text{EDGECOLOR}$  halts, in case it lasts for strictly less than  $n$  phases. Execute now  $\text{COLORVAL}(\mathcal{F})$  making the random choices of  $\text{EDGECOLOR}$ .  $\square$

**Lemma 6.4.6.** *We have that  $\hat{P}_n \leq \sum_{|\mathcal{F}|=n} \Pr[V_{\mathcal{F}}]$  and that  $\hat{Q}_n \leq 1 - \left(1 - \left(\frac{2}{2+\epsilon}\right)^m\right)$ .*

*Proof.* The first inequality is obvious. Now, just for the sake of notational convenience, let us call below a coloring *strongly proper* if it is proper and 4-acyclic. Also by just *random* we mean a coloring generated by independently coloring all edges once, choosing for each a color u.a.r from the palette.

For the second inequality, first observe that by Lemma [6.4.3](#) we have that  $\hat{Q}_n$ :

$$\begin{aligned} \hat{Q}_n &= \Pr[\text{COLORVAL succeeds for at least one } \mathcal{F} \text{ with } |\mathcal{F}| < n \mid \\ &\quad \text{the generated color is not strongly proper}] \times \\ &\quad \Pr[\text{a random coloring is not strongly proper}] \leq \\ &\quad \Pr[\text{a random coloring is not strongly proper}]. \end{aligned}$$

The given bound then follows from the cornerstone result of Esperet and Parreau given in Lemma [3.4.1](#).  $\square$

The second inequality of the Lemma above has as corollary, by Lemma [6.4.5](#), that the probability that the number of repetitions of the **while**-loop of MA is  $n$ , is inverse exponential in  $n$ . So all that remains to be proved to complete the proof of the main Theorem is to show that  $\sum_{|\mathcal{F}|=n} \Pr[V_{\mathcal{F}}]$  is inverse exponential in  $n$ . We do this in the next subsection, expressing the sum as a recurrence.

#### 6.4.4 Recurrence

We will estimate  $\sum_{|\mathcal{F}|=n} \Pr[V_{\mathcal{F}}]$  by purely combinatorial arguments. Towards this end, we first define the weight of a forest, denoted by  $\|\mathcal{F}\|$ , to be the number

$$\prod_{i=1}^{|\mathcal{F}|} \left( \frac{1}{K^{(2k_i-2)}} \right),$$

(recall that  $|\mathcal{F}|$  denotes the number of nodes of  $\mathcal{F}$ ), and observe that by Lemma [6.4.4](#)

$$\sum_{|\mathcal{F}|=n} \Pr[V_{\mathcal{F}}] = \sum_{|\mathcal{F}|=n} \|\mathcal{F}\|. \quad (6.33)$$

Assume that the empty tree is a feasible forest with number of nodes 0 and weight 1. From the definition of a feasible forest, we have that such a forest is comprised of  $m$  possibly empty trees, in one to one correspondence to the edges, the root of each nonempty one having as edge-label the corresponding edge. For  $j = 1, \dots, m$ , let  $\mathcal{T}_j$  be the set of all possible feasible trees corresponding to the edge  $e_j$  and let  $\mathcal{T}$  be the collection of all  $m$ -ary sequences  $(T_1, \dots, T_m)$  with  $T_j \in \mathcal{T}_j$ .

Now, obviously:

$$\begin{aligned} \sum_{|\mathcal{F}|=n} \|\mathcal{F}\| &= \sum_{\substack{(T_1, \dots, T_m) \in \mathcal{T} \\ |T_1| + \dots + |T_m| = n}} \|T_1\| \cdots \|T_m\| \\ &= \sum_{\substack{n_1 + \dots + n_m = n \\ n_1, \dots, n_m \geq 0}} \left( \left( \sum_{\substack{T_1 \in \mathcal{T}_1: \\ |T_1| = n_1}} \|T_1\| \right) \cdots \left( \sum_{\substack{T_m \in \mathcal{T}_m: \\ |T_m| = n_m}} \|T_m\| \right) \right) \end{aligned} \quad (6.34)$$

We will now obtain a recurrence for each factor of the rhs of (6.34). Let:

$$q = \frac{\Delta - 1}{K} = \frac{\Delta - 1}{\lceil (2 + \epsilon)(\Delta - 1) \rceil}. \quad (6.35)$$

**Lemma 6.4.7.** *Let  $\mathcal{T}^e$  be anyone of the  $\mathcal{T}_j$ . Then:*

$$\sum_{\substack{T \in \mathcal{T}^e \\ |T| = n}} \|T\| \leq R_n, \quad (6.36)$$

where  $R_n$  is defined as follows:

$$R_n := \sum_{k \geq 3} q^{2k-2} \left( \sum_{\substack{n_1 + \dots + n_{2k-2} = n-1 \\ n_1, \dots, n_{2k-2}}} Q_{n_1} \cdots Q_{n_{2k-2}} \right) \quad (6.37)$$

and  $R_0 = 1$ .

*Proof.* Indeed, the result is obvious if  $n = 0$ , because the only possible  $T$  is the empty tree, which has weight 1. Now if  $n > 0$ , observe that there are at most  $\Delta^{2k-2}$  possible cycles with  $2k$  edges, for some  $k \geq 3$ , that can be the cycle-edge of the root of a tree  $T \in \mathcal{T}^e$  with  $|T| > 0$ . Since the probability of each such cycle having homochromatic equal parity sets is  $(\frac{1}{K})^{2k-2}$ , the lemma follows.  $\square$

We will now asymptotically analyze the coefficients of the OGF  $R(z)$  of  $R_n$ .

Multiply both sides of (6.37) by  $z^n$  and sum for  $n = 1, \dots, \infty$  to get

$$R(z) - 1 = \sum_{k \geq 3} \left( q^{2k-2} z R(z)^{2k-2} \right), \quad (6.38)$$

with  $R(0) = 1$ . Setting  $W(z) = R(z) - 1$  we get

$$W(z) = \sum_{k \geq 3} \left( q^{2k-2} z (W(z) + 1)^{2k-2} \right), \quad (6.39)$$

with  $W(0) = 0$ . For notational convenience, set  $W = W(z)$ . Then from (6.39) we get:

$$W = z \sum_{k \geq 2} \left( q^{2k} (W + 1)^{2k} \right) = z \frac{(q(W + 1))^4}{1 - (q(W + 1))^2}. \quad (6.40)$$

Now, set:

$$\phi(x) = \frac{(q(x + 1))^4}{1 - (q(x + 1))^2}, \quad (6.41)$$

to get from (6.40):

$$W = z\phi(W). \quad (6.42)$$

By [94, Proposition IV.5, p. 278], we have that if  $\phi$  is a function analytic at 0 having non-negative Taylor coefficients and such that  $\phi(0) \neq 0$  and if  $r$  is the radius of convergence of the series representing  $\phi$  at 0 and finally if  $\lim_{x \rightarrow r^-} \phi(x) = +\infty$  we get that  $[z^n]R \asymp (1/\rho)^n$ , i.e.  $\limsup ([z^n]R)^{1/n} = 1/\rho$ , where  $\rho = \frac{\tau}{\phi(\tau)}$ , and  $\tau$  is the (necessarily unique) solution of the characteristic equation:

$$\frac{\tau\phi'(\tau)}{\phi(\tau)} = 1 \quad (6.43)$$

within  $(0, r)$  (for the asymptotic notation “ $\asymp$ ” see [94, IV.3.2, p. 243]).

In our case, with  $\phi$  given in (6.41). The above hypotheses are easily satisfied for each  $q \in (0, 1)$  with  $r = (1/q) - 1$ . Also, if  $q = \frac{1}{2}$ , then  $r = 1$ , and the (necessarily unique) solution of the characteristic equation  $\frac{\tau\phi'(\tau)}{\phi(\tau)} = 1$  within  $(0, 1)$  is  $\tau = -2 + \sqrt{5}$ , and therefore  $\rho = \frac{\tau}{\phi(\tau)} = 1$ .

Now, as it is shown in the the proof of [94, Proposition IV.5, p. 278], the function  $\frac{x\phi'(x)}{\phi(x)}$  is increasing with  $x \in (0, r)$ . Therefore, for each  $x \in (0, r)$ , it is also increasing with  $q \in (0, 1/(x + 1))$ , because  $q$  appears in the argument of  $\phi$  as  $q(x + 1)$ . So, if  $q > 1/2$ , the unique solution of  $\frac{\tau\phi'(\tau)}{\phi(\tau)} = 1$  within  $(0, r)$  becomes smaller than  $-2 + \sqrt{5}$ . Now  $\phi'(x)$  is increasing with  $x \in (0, r)$  ( $\phi$ 's Taylor series has positive coefficients), and so  $[z^n]R \asymp (1/\rho)^n$ , for some  $\rho > 1$ , because  $\rho = \frac{\tau}{\phi(\tau)} = \frac{1}{\phi'(\tau)}$ .

By the above, and since there are at most  $n^m$  sequences  $n_1, \dots, n_m$  of integers that add up to  $n$ , we get by Lemma 6.4.6, equations (6.33) and (6.34), and Lemma 6.4.7 that:

$$\hat{P}_n \leq n^m \left(\frac{1}{\rho}\right)^n. \quad (6.44)$$

Thus, by Equation (6.44) and Lemma 6.4.5, we get that:

**Lemma 6.4.8.** *For any  $\epsilon > 0$ , and for any graph  $G$  for which  $l$ ,  $m$  and  $\Delta$  (resp. the number of vertices, the number of edges and the maximum degree of  $G$ ) are considered constant, and given the availability of at least  $(2 + \epsilon)(\Delta - 1)$  colors, there exists a constant  $c \in (0, 1)$ , depending on  $l, m, \Delta$  and  $\epsilon$ , such that the probability that `EDGECOLOR` executes at least  $n$  steps is  $\leq c^n$ .*

This completes the proof of our two probabilistic claims, and the proofs of the Theorem and Corollary, our main results, given there.

## 6.5 Application of the LLL to c-separating codes

In this final section, we utilize our algorithmic approach to the LLL to construct  $c$ -separating codes. First, we show that the LLL implies the existence of such codes (Subsec. 6.5.1). We then explicitly construct such codes and compare our results to other extant works (Subsec. 6.5.2).

### 6.5.1 A lower bound on the rate of c-separating binary codes

We use the Lovász Local Lemma to obtain a lower bound on the rate of  $c$ -separating binary codes, of the same order of magnitude as the bound in Proposition 3.5.1. However, the use of the LLL allows us to move from the existence result to an explicit construction.

Let  $X_{k,l}$ ,  $1 \leq k \leq M$ ,  $1 \leq l \leq N$ , be  $M \cdot N$  independent random variables, following the Bernoulli distribution, where:

$$\Pr(X_{k,l} = 0) = \Pr(X_{k,l} = 1) = \frac{1}{2}.$$

Let also  $\Omega = \{0, 1\}^{MN}$  be the set of all  $(MN)$ -ary binary vectors. It is convenient to think  $\Omega$  as the set of  $M \times N$  matrices with binary entries, as we can then immediately correspond an assignment of values to the variables to an  $(N.M)_2$  binary code. Let us denote such a code by  $\mathcal{C}$ . Recall that  $M$  is the size of the code, i.e. the number of code words it contains and  $N$  is the length of these code words.

It is straightforward to observe that if two sets  $U, V$  of size  $c' < c$  are not separated, then any two supersets  $U' \supset U$  and  $V' \supset V$  of size  $c$ , are also not separated. Thus, it suffices to deal with sets of exactly  $c$  code words. Let  $U^i = \{\mathbf{u}^{i,1}, \dots, \mathbf{u}^{i,c}\}$  be a set of  $c$  distinct code words of  $\mathcal{C}$ , for  $1 \leq i \leq \binom{M}{c}$ , where  $\mathbf{u}^{i,j} = (u_1^{i,j}, \dots, u_n^{i,j})$ ,  $u_l^{i,j} \in \mathbb{F}_2$ ,  $l = 1, \dots, N$ . Let:

$$\mathcal{P} := \{\{U^i, U^j\} \mid |U^i| = |U^j| = c, U^i \cap U^j = \emptyset \text{ and } 1 \leq i, j \leq \binom{M}{c} : i \neq j\},$$

be the set of disjoint sets of size  $c$  of distinct code words of  $\mathcal{C}$ . For each  $\{U^i, U^j\} \in \mathcal{P}$ , we define the event  $E_{i,j}$  to occur when  $U^i, U^j$  are *not* separated. There are  $m = \binom{M}{c} \binom{M-c}{c}$  such events, which we assume to be ordered arbitrarily.

We now prove two lemmas. The first one concerns the probability of each event to occur.

**Lemma 6.5.1.** *The probability of any event  $E_{i,j}$  is:*

$$\Pr[E_{i,j}] = \left(1 - \frac{1}{2^{c-1}}\right)^N. \quad (6.45)$$

*Proof.* Consider the  $s$ -th coordinate of  $\mathbf{u}^{i,1}, \dots, \mathbf{u}^{i,c}$  and  $\mathbf{u}^{j,1}, \dots, \mathbf{u}^{j,c}$ . The probability that  $U^i$  and  $U^j$  are *not* separated in the  $s$ -th coordinate, is equal to 1 minus the probability that  $U^i$  and  $U^j$  are separated in that coordinate. The latter is exactly  $\frac{1}{2^{c-1}}$ , since for separation all values  $\mathbf{u}^{i,1}, \dots, \mathbf{u}^{i,c}$  have to be equal and different from all values  $\mathbf{u}^{j,1}, \dots, \mathbf{u}^{j,c}$ . Thus, since each coordinate of a code word takes values independently, we have that:

$$\Pr[E_{i,j}] = \left(1 - \frac{1}{2^{c-1}}\right)^N$$

and the proof is finished.  $\square$

Assume that  $\mathbf{v}^1, \dots, \mathbf{v}^M$  is some arbitrary ordering of the code words of  $\mathcal{C}$ , where  $\mathbf{v}^k = (v_1^k, \dots, v_N^k)$ ,  $k = 1, \dots, M$ . Then we have that:

$$\text{sc}(E_{i,j}) := \{X_{k,1}, \dots, X_{k,N} \mid \mathbf{v}^k \in U^i \cup V^j\}.$$

Note that  $E_{i,j}$  cannot have some  $X_{k,l} \in \text{sc}(E_{i,j})$ , but not  $X_{k,l'}$ , for any  $l' \neq l$ .

We say that two events are dependent, if they share at least one common random variable  $X_{k,l}$ ,  $1 \leq k \leq M$  and  $1 \leq l \leq N$ . In fact, as is apparent from the above, if two events share  $X_{k,l}$ , they also share all  $X_{k,l'}$  such that  $l' \in \{1, \dots, N\}$ . The next lemma bounds from above the number of dependencies for each event.

**Lemma 6.5.2.** *The number of events depending on  $E_{i,j}$  is at most:*

$$s = \frac{1}{(c-1)!^2} 6M^{2c-1} - 1. \quad (6.46)$$

*Proof.* It is clear that two events are dependent if and only if the corresponding pairs of sets have at least one common code word. To make notation less cumbersome, we will say that a code word belongs in an event if the corresponding random variables are in the scope of the event.

By subtracting from the total number of events the number of events that share no common code word with  $E_{i,j}$ , we get that the number of events that are different from and depend on  $E_{i,j}$  is equal to:

$$\binom{M}{c} \binom{M-c}{c} - \binom{M-2c}{c} \binom{M-3c}{c} - 1. \quad (6.47)$$

For binomial coefficients we have the equality:

$$\binom{M}{c} = \binom{M-1}{c-1} + \binom{M-1}{c} \quad (6.48)$$

We can apply it repeatedly:

$$\begin{aligned} \binom{M-1}{c} &= \binom{M-2}{c-1} + \binom{M-2}{c} \\ \binom{M-2}{c} &= \binom{M-3}{c-1} + \binom{M-3}{c} \\ &\vdots = \vdots \\ \binom{M-(2c-1)}{c} &= \binom{M-2c}{c-1} + \binom{M-2c}{c}. \end{aligned} \quad (6.49)$$

Therefore,

$$\binom{M}{c} = \sum_{i=1}^{2c} \binom{M-i}{c-1} + \binom{M-2c}{c} \quad (6.50)$$

Analogously, we have

$$\binom{M-c}{c} = \sum_{i=1}^{2c} \binom{M-c-i}{c-1} + \binom{M-3c}{c} \quad (6.51)$$

Since for all  $i \geq 0$

$$\binom{M}{c} > \binom{M-i}{c},$$

from (6.50) and (6.51) we have

$$\binom{M}{c} < 2c \binom{M-1}{c-1} + \binom{M-2c}{c} \quad (6.52)$$

and

$$\binom{M-c}{c} < 2c \binom{M-c-1}{c-1} + \binom{M-3c}{c} \quad (6.53)$$

We have that

$$\begin{aligned} \binom{M}{c} \binom{M-c}{c} < \\ \left[ 2c \binom{M-1}{c-1} + \binom{M-2c}{c} \right] \left[ 2c \binom{M-c-1}{c-1} + \binom{M-3c}{c} \right] \end{aligned} \quad (6.54)$$

Noting that

$$\binom{M-1}{c-1} = \frac{c}{M} \binom{M}{c}$$

we have

$$2c \binom{M-1}{c-1} 2c \binom{M-c-1}{c-1} = \frac{4c^3}{M} \binom{M}{c} \binom{M-c-1}{c-1}.$$

Now we can use previous expressions to expand the left side of (6.54). Arranging terms we have (for  $M \geq 2c^2$ )

$$\binom{M}{c} \binom{M-c}{c} < 6c \binom{M}{c} \binom{M}{c-1} + \binom{M-2c}{c} \binom{M-3c}{c} \quad (6.55)$$



Now, it is clear that substituting the left side of (6.55) in (6.47) the term  $\binom{M-2c}{c}\binom{M-3c}{c}$  cancels, so:

$$\binom{M}{c}\binom{M-c}{c} - \binom{M-2c}{c}\binom{M-3c}{c} - 1 < 6c\binom{M}{c}\binom{M}{c-1} - 1 \quad (6.56)$$

Now, by using the approximation  $\binom{n}{k} \leq n^k/k!$ , we have that

$$6c\binom{M}{c}\binom{M}{c-1} \leq 6\frac{1}{(c-1)!(c-1)!}M^{2c-1}.$$

□

Armed with the previous lemmas and Th. 3.3.4, we can state the following result.

**Theorem 6.5.1.** *For every  $n > 0$  there exists a binary  $c$ -separating code of size:*

$$M \leq \left(\frac{(c-1)!^2}{6e}\right)^{\frac{1}{2c-1}} \left(\frac{2^{2c-1}}{2^{2c-1}-1}\right)^{\frac{n}{2c-1}} \quad (6.57)$$

*Proof.* Using values of  $p$  and  $s$  provided by Lemmas 6.5.1 and 6.5.2, we see that for the condition of the symmetric LLL to be satisfied, it must hold that:

$$e\left(1 - \frac{1}{2^{2c-1}}\right)^N \frac{1}{(c-1)!^2} 6M^{2c-1} \leq 1.$$

This is clearly true for the  $M$  in (6.57) and thus, the existence claim follows by Th. 3.3.4. □

By Theorem 6.5.1, we immediately obtain the following result.

**Corollary 6.5.1.** *There exist binary  $c$ -separating codes of asymptotic rate:*

$$\underline{R}(n, c)_2 \geq -\frac{\log_2(1 - 2^{-(2c-1)})}{2c-1} - \frac{1}{n} \left(\frac{\log_2 \frac{(c-1)!^2}{6e}}{2c-1}\right). \quad (6.58)$$

*Proof.* The proof is immediate by using the value of  $M$  in Theorem 6.5.1 in Def. 3.5.2. □

**Remark 6.5.1.** *Note our result is asymptotically equal to the bound given by Barg et al in Proposition [3.5.1](#). For finite values of the code length it falls short by:*

$$\left( \frac{\log_2 \frac{(c-1)!^2}{6e}}{2c-1} \right) = O \left( \frac{2c-2}{2c-1} \log_2(c-1) \right),$$

*which, for a fixed  $c$ , is of little consequence for interesting values of the code length.*

For the case where  $c = 2$ , we have the following corollaries whose proof is straightforward from the above.

**Corollary 6.5.2.** *For every  $n > 0$  there exists a binary 2-separating code of size:*

$$M \leq \frac{1}{\sqrt[3]{5e}} \left( \frac{8}{7} \right)^{N/3}. \quad (6.59)$$

**Corollary 6.5.3.** *There exist binary 2-separating codes of asymptotic rate  $R \approx 0.064$ .*

**Remark 6.5.2.** *It should be noted that the results in both Theorem [6.5.1](#) and Corollary [6.5.2](#) mean that we can find codes of size as large as the greatest integer that is bounded from the corresponding right hand side of Eq. [\(6.57\)](#) and [\(6.59\)](#).*

## 6.5.2 Explicit constructions

Now that we have established a lower bound for the rate of 2-separating binary codes, we turn our attention to obtaining explicit constructions of such codes. We first see that to obtain a code of positive rate, the computational complexity of our algorithm turns out to be exponential in the code length (see Remark [6.5.3](#)). We then show that we can tune the algorithm to be polynomial in the code length, at the cost of having non positive code rate. Nevertheless, the code we construct has a better rate than that of a Simplex code of equivalent length.

**Direct application of the algorithmic LLL for 2-separating codes**  
Consider CODESEP, algorithm [10](#) below.

**Algorithm 10** CODESEP.

- 
- 1: Sample the variables  $X_{k,l}$ ,  $k = 1, \dots, M$ ,  $l = 1, \dots, N$  and let  $\mathbf{C}$  be the resulting code.
  - 2: **while** there exists a pair of non-separated subsets of  $\mathcal{P}$ , let  $\{U^i, U^j\}$  be the first such pair in some arbitrary ordering of  $\mathcal{P}$  and **do**
  - 3:     RESAMPLE( $E_{i,j}$ )
  - 4: **end while**
  - 5: Output current code  $\mathcal{C}$ .

RESAMPLE( $E_{i,j}$ )

- 1: Resample the variables in  $\text{sc}(E_{i,j})$ .
  - 2: **while** there exists a pair of non-separated subsets of  $\mathcal{P}$ , that share at least one code word with either  $U^i$  or  $U^j$ , let  $\{U^{i'}, U^{j'}\}$  be the first such pair in some arbitrary ordering of  $\mathcal{P}$  and **do**
  - 3:     RESAMPLE( $E_{i',j'}$ )
  - 4: **end while**
- 

In what follows, we prove the following result.

**Theorem 6.5.2.** *For every  $n > 0$ , the probability that CODESEP lasts for at least  $n$  rounds is inverse exponential in  $n$ . Upon termination, CODESEP outputs a 2-separating code of size:*

$$M = \left\lceil \left( \frac{(c-1)!^2}{6e} \right)^{\frac{1}{2c-1}} \left( \frac{2^{2c-1}}{2^{2c-1} - 1} \right)^{\frac{N}{2c-1}} \right\rceil. \quad (6.60)$$

*Proof.* First, by the **while**-loop of line [2](#), it follows that if and when CODESEP terminates, it produces a  $c$ -separating code  $\mathbf{C}$ . Thus, we only need to show that it will indeed terminate fast.

A RESAMPLE call made from line [3](#) is a *root* call, while one made from line [3](#) is a *recursive* one. Finally, a *round* is the duration of any RESAMPLE call.

Note that at each round, CODESEP makes some progress, in the sense that any pair  $\{U^{i'}, U^{j'}\}$  that was separated at the beginning of some RESAMPLE( $E_{i,j}$ ), will also be separated at the end of that call, along with the pair  $\{U^i, U^j\}$ . Indeed, by line [3](#) of RESAMPLE( $E_{i,j}$ ), it is obvious that  $U^i$  and  $U^j$  will be separated at the end of this call. Furthermore, any  $\{U^{i'}, U^{j'}\}$  that was not separated at the beginning of the call, will either remain as

such for the duration of this round, or it will cease being separated due to some resampling of the random variables. In the latter case, it must share at least one code word with some other pair  $\{U^{i''}, U^{j''}\}$  that was resampled during  $\text{RESAMPLE}(U^i, U^j)$ . But then, by line 2 of  $\text{RESAMPLE}(E_{i'', j''})$ ,  $\text{RESAMPLE}(E_{i', j'})$  would have been called.

Given an execution of  $\text{CODESEP}$ , we construct a labeled rooted forest  $\mathcal{F}$  (i.e. forest comprised of rooted trees), that we call the *witness forest* of the execution.

- (i) For each  $\text{RESAMPLE}(E_{i,j})$  call, we construct a node labeled by  $E_{i,j}$ .
- (ii) If  $\text{RESAMPLE}(E_{i',j'})$  is called from line 3 of  $\text{RESAMPLE}(E_{i,j})$ , then the corresponding node labeled by  $E_{i',j'}$  is a child of that labeled by  $E_{i,j}$ .

It is not difficult to see that the roots of  $\mathcal{F}$  correspond to root calls of  $\text{RESAMPLE}$ , while the rest of the nodes to recursive calls. Furthermore, since at the end of a  $\text{RESAMPLE}(E_{i,j})$  call,  $\{U^i, U^j\}$  are separated, (i) the labels of the roots are pair-wise distinct and (ii) the same holds for the labels of siblings. Finally, (iii) if a node labeled by  $E_{i',j'}$  is a child of one labeled by  $E_{i,j}$ , then  $\{U^{i'}, U^{j'}\}$  share at least one code word with  $\{U^i, U^j\}$ . Thus, the children of an internal node of  $\mathcal{F}$  are at most  $s + 1$ .

We call any forest  $\mathcal{F}$  that satisfies (i), (ii) and (iii), *feasible*. The number of its nodes is denoted by  $|\mathcal{F}|$ . Note that the class of feasible forests is broader than that of witness forests. We order the nodes of a feasible forest in the following way: (i) trees and siblings are ordered according to the order of their labels (ii) the nodes of a tree are ordered in pre-order, respecting the ordering of siblings. Following this ordering, we can obtain the *label-sequence*  $\mathcal{L}(\mathcal{F}) = (E^1, \dots, E^n)$  of a feasible forest  $\mathcal{F}$  with  $n$  nodes, where each  $E^r$  is some  $E_{i,j}$ .

Let  $P_n$  be the probability that  $\text{CODESEP}$  lasts for at least  $n$  rounds and  $W_{\mathcal{F}}$  the event that  $\mathcal{F}$  is the witness forest of an execution of  $\text{CODESEP}$ . Then, it holds that:

$$P_N = \Pr \left[ \bigcup_{\mathcal{F}: |\mathcal{F}|=n} W_{\mathcal{F}} \right] \leq \sum_{\mathcal{F}: |\mathcal{F}|=n} \Pr[W_{\mathcal{F}}], \quad (6.61)$$

where the last inequality is by union-bound.

**Lemma 6.5.3.** *For any feasible forest  $\mathcal{F}$  with  $n$  nodes,  $\Pr[W_{\mathcal{F}}] \leq p^n$ .*

*Proof.* Assume that  $E_{i,j}$  is the  $r$ -th event in the label sequence  $\mathcal{L}(\mathcal{F})$  of a feasible forest  $\mathcal{F}$  with  $n$  nodes. For  $\mathcal{F}$  to be constructed, we have  $n$  independent random samplings of the code  $\mathbf{C}$ , where, at the  $r$ -th such sampling,  $r = 1, \dots, n$ , it must be the case that:

- if  $r = 1$ , then  $\{U^i, U^j\}$  is the first pair of not separable sets, otherwise,
- if  $E_{i,j}$  is the label of a root, then  $\{U^i, U^j\}$  is the first pair of not separable sets and no pair of sets that share a code word with those of the last leaf can occur and
- if  $E_{i,j}$  is an internal node, then  $\{U^i, U^j\}$  must be the first non-separated pair from those that share a code word with the parent label.

Thus, we can bound  $\Pr[W_{\mathcal{F}}]$  by the product:

$$\prod_{r=1}^n \Pr[\text{the pair of sets corresponding to } E^r \text{ is not separated}].$$

Now the result follows immediately.  $\square$

Thus, to bound the rhs of Eq. (6.61), we need to count the number of feasible forests with  $n$  internal nodes. To do that, we first hang from every node of a feasible forest  $\mathcal{F}$  leaves, labeled suitably such that each node labeled by  $E_{i,j}$  of  $\mathcal{F}$ , now has exactly  $s + 1$  children, labeled with the corresponding events  $E_{i',j'}$ , such that  $\{U^i, U^j\}$  and  $\{U^{i'}, U^{j'}\}$  share at least one common code word. Also, trees comprised of a single root/leaf are build, such that the sets of root-labels and the set of undesirable events coincide. By ordering the nodes of these new forests in the same way as before and making them planar, we can easily see that that there is a one-to-one and onto correspondence with the feasible forests.

Thus, we can count the number  $f_n$  of of *rooted planar forests* with  $n$  internal nodes, comprised of  $m$  *full*  $(s + 1)$ -ary rooted planar trees. Denote the number of full  $(s + 1)$ -ary rooted planar trees with  $n$  internal nodes by  $t_n$ . It holds that  $t_n = \frac{1}{sn+1} \binom{(s+1)n}{n}$  (see [193, Theorem 5.13]), which, by Stirling's approximation gives that that there is some constant  $A$ , depending only on  $s$ , such that:

$$t_n < A \left( \left(1 + \frac{1}{s}\right)^s (s + 1) \right)^n. \quad (6.62)$$

Finally, by (6.62), we get:

$$f_n = \sum_{\substack{n_1 + \dots + n_m = n \\ n_1, \dots, n_m \geq 0}} t_{n_1} \cdots t_{n_m} < (An)^m \left( \left(1 + \frac{1}{s}\right)^s (s+1) \right)^n. \quad (6.63)$$

Thus, taking Eq. (6.61) and (6.63), we have that:

$$P_n < (An)^m \left( \left(1 + \frac{1}{s}\right)^s (s+1)p \right)^n, \quad (6.64)$$

which concludes the proof.  $\square$

**Remark 6.5.3.** Consider line 2 of CODESEP. For the algorithm to find the least indexed event, it must go over all the approximately  $M^{2c}$  elements of  $\mathcal{P}$  and check if they are separated. Accordingly, in line 2 of a RESAMPLE( $E_{i,j}$ ) call of CODESEP, the algorithm must check all the approximately  $M^c$  events in the neighborhood of  $E_{i,j}$ . Given the bound we proved for  $M$ , it is easy to see that in both cases, the number of events that need to be checked is exponentially large in  $N$ . In what follows, we will deal, in a way, with this problem.

**Constructions of polynomial complexity** In Remark 6.5.3 above, we have exposed the drawback of applying in a straightforward way the algorithmic version of the LLL. Since we are aiming for a code with asymptotic positive rate, this means that the number of code words has to be exponential in the code length, which implies that the algorithmic complexity is exponential in the code length too. If we insist in building positive rate 2-separating codes and use the algorithmic version of the LLL for it, this exponential dependence seems to be unavoidable.

In light of this we tackle the problem of removing the exponential dependence of the construction, at the cost of producing 2-separating codes with non-positive, but still relatively good, rate.

One can see from Theorem 6.5.2 that the size of the 2-separating code we produce, directly affects the complexity of CODESEP, since the number of checks at the **while**-loops of lines 2 and 2 depend on it. Furthermore, the size of the produced code depends on the probability of the undesirable events.

Following this line of thought, we are going to plug in the LLL condition a probability for bad events that is actually higher than the real one. Of course we will only be able to prove the existence of not so “optimal” objects, but we do so in polynomial time. This might seem somewhat unorthodox. What we do essentially, is restrict the size of the code we sample and resample. Thus, we allow CODESEP to make much less checks and corrections in order to produce a 2-separating code  $\mathcal{C}$ .

**Lemma 6.5.4.** *Let  $N > 0$  and any  $\alpha > 0$  be integers such that*

$$\frac{1}{N^\alpha} > \left(1 - \frac{1}{2^{2c-1}}\right)^N. \quad (6.65)$$

*Then, there exists a binary  $c$ -separating code of size:*

$$M \leq \left(\frac{c!c!}{16c^2}\right)^{\frac{1}{2c-1}} N^{\frac{\alpha}{2c-1}} \quad (6.66)$$

*Proof.* The proof is identical to that of Theorem 6.5.1, by taking  $p = \frac{1}{N^\alpha}$ .  $\square$

For explicit binary 2-separating codes, the interested reader is referred to [13] and the references therein. Known constructions are somewhat particular and rare. For instance, there exists a 2-separating binary [35,6] code, there exists a 2-separating binary [126,14] code, and of course as stated in Lemma 3.5.1 the Simplex code is also 2-separating.

We now take a step into constructing 2-separating binary codes with rate better than the Simplex code for any code length and in polynomial time to their length. The following Corollary is a weaker result than Corollary 6.5.2 and follows from the fact that in Theorem 3.3.4 one only needs an upper bound of the probability of the bad events.

**Corollary 6.5.4.** *For every  $N > 0$  and any  $\alpha > 0$ , there exists a binary 2-separating code of size:*

$$M \leq \frac{1}{\sqrt[3]{5}e} N^{\alpha/3}. \quad (6.67)$$

*Proof.* Immediate consequence of Lemma 6.5.4.  $\square$

Recall that the rate of a Simplex code of length  $N$  is

$$R_{\text{Simplex}}(N) = \frac{\log_2(n+1)}{N}. \quad (6.68)$$

With little algebraic manipulation one can see that for any  $N = 2^k - 1$  with  $k > 0$ , and any  $\alpha > 0$  such that:

$$N^\alpha > 5e(n+1)^3,$$

the codes in Corollary 6.5.4 have better rate than the Simplex code of the same length and with a polynomial (in  $N$ ) number of code words.

As an illustration, note the following numerical example. We take  $N = 255$ . The corresponding Simplex code has dimension  $k = 8$ , and rate

$$R_{\text{Simplex}}(255) = \frac{\log_2(256)}{255} = 0.031.$$

According to Corollary 6.5.4, for length  $N = 255$  we can take  $\alpha = 6$ . Then, there exists a code of size  $M = 25640$ , that is, of rate

$$R(255) = \frac{\log_2(25640)}{255} = 0.057.$$

Recall the observations made in Remark 6.5.3. In CODESEP we have to:

- go over the approximately  $2M^4$  elements of  $\mathcal{P}$  in line 2 and
- check all the approximately  $5M^3$  events in the neighborhood of  $E_j$  in line 2 of a RESAMPLE( $E_j$ ) call.

Observe that, with the value of  $M \leq \frac{1}{\sqrt[3]{5e}} N^{\alpha/3}$  given by Corollary 6.5.4, this can be done in polynomial time in the code length. Furthermore, by taking this value for  $M$ , by Theorem 6.5.2 applies verbatim. Thus, we have proven the following:

**Theorem 6.5.3.** *For every  $n > 0$ , the probability that CODESEP lasts for at least  $n$  rounds is inverse exponential in  $n$ . Upon termination, CODESEP outputs a 2-separating code of length  $n$  and size:*

$$M \leq \frac{1}{\sqrt[3]{5e}} N^{\alpha/3},$$

*with rate larger than the Simplex code of the same length. The computational complexity is  $O(N^{4\alpha/3})$  for any  $\alpha$  such that  $N^\alpha > 5e(N+1)^3$ .*



**Using code concatenation** It is an established fact that in order to obtain infinite families of codes with a certain property one can resort to code concatenation (see Forney [95]). This construction consists of two codes. An *inner* code over a small alphabet  $q$ , say  $C_i = (N, M_i)_q$ . The size of the inner code is precisely the size of *outer* code's alphabet  $C_o = (N_o, M_o)_Q$  with  $|\mathbf{Q}| = M_i$ . There is a bijection  $\psi : \mathbf{Q} \rightarrow C_i$  between the inner code words and the elements of the outer code's alphabet. Take an outer code's code word and apply  $\psi$  to each symbol. This will give us an  $NN_o$ -tuple. We denote by  $C_i \circ C_o$  the  $(NM, M_o)_q$  the code formed by all such  $NN_o$ -tuples. Note that  $C_i \circ C_o$  is a code over the alphabet  $q$ .

It is straightforward to prove that if the rates of the composing codes are  $R_i$  and  $R_o$ , then rate of the concatenated code is  $R_i \cdot R_o$ .

**Lemma 6.5.5.** *Let  $C$  be an  $(N_o, M)_Q$  code with minimum distance  $d$ . If*

$$d > N_o - \frac{N_o}{c^2} \quad (6.69)$$

*then  $C$  is  $c$ -separating.*

In fact any code satisfying the distance constrain of the lemma possesses two stronger properties, namely the  $c$ -Identifiable Parent Property (IPP) and the  $c$ -Traceability (TA) property (see Staddon et al. [203]).

**Lemma 6.5.6.** *Let  $C_i = (N, M_i)_2$  be a  $c$ -separating binary code of rate  $R_i$  and let  $C_o = (N_o, M_o)_Q$  a  $c$ -separating code of rate  $R_o$  with  $|\mathbf{Q}| = M_i$ . Then  $C_i \circ C_o$  is a binary  $c$  separating code of rate  $R_i R_o$ .*

Our goal is to construct a family of asymptotically good  $c$ -separating binary codes in polynomial time. For the inner code, we are going to use our algorithmic constructions. For the outer code we will rely on algebraic-geometric codes.

We will need the following lemmas.

**Lemma 6.5.7.** *A  $c$ -separating AG  $(N_o, M)_Q$  code over an alphabet  $\mathbf{Q}$  can be constructed for rates*

$$R < \frac{1}{c^2} - \frac{1}{\sqrt{|\mathbf{Q}|} - 1}. \quad (6.70)$$

By the results in Shum et al. [197], these codes can be constructed with polynomial complexity  $O((N_o \log_Q N_o)^3)$ .

*Proof.* The lemma follows from (6.69) and (3.20). Since, there exist an AG code with  $R = 1 - \frac{d}{N_o} - \frac{1}{\sqrt{|\mathbf{Q}|-1}}$  and separation implies  $\frac{d}{N_o} > 1 - \frac{1}{c^2}$ .  $\square$

**Lemma 6.5.8.** *Setting the code length  $N$  to be  $2^{2c}c \ln M$  in Eq. (6.57), satisfies Theorem 6.5.1 for codes of size  $M$ .*

*Proof.* We can express (6.57) as

$$\frac{\ln\left(\frac{M^{2c-1}6e}{(c-1)!^2}\right)}{\ln\left(\frac{2^{2c-1}}{2^{2c-1}-1}\right)} < N.$$

Since,

$$\ln\left(\frac{1}{1 - \frac{1}{2^{2c-1}}}\right) = -\ln\left(1 - \frac{1}{2^{2c-1}}\right),$$

by using the well known approximation  $-\ln\left(1 - \frac{1}{2^{2c-1}}\right) \geq \frac{1}{2^{2c-1}}$  we have

$$\frac{\ln\left(\frac{M^{2c-1}16c^2}{c!c!}\right)}{\ln\left(\frac{2^{2c-1}}{2^{2c-1}-1}\right)} \leq 2^{2c-1} \ln\left(\frac{6eM^{2c-1}}{(c-1)!^2}\right) < 2^{2c}c \ln M.$$

So we can take  $N$  such that:

$$N \geq 2^{2c}c \ln M$$

and the proof is completed.  $\square$   $\square$

Our main result is the following.

**Theorem 6.5.4.** *There are explicit constructions of binary  $c$ -separating code of rate  $R = O\left(\frac{1}{2^{2c}c^3}\right)$ .*

*Proof.* We are going to use code concatenation for the construction. For the outer code  $C_o$  we will take a  $c$ -separating code as in Lemma 6.5.7, with  $|\mathbf{Q}| = c^{2\beta}$ , where  $\beta > 2$  is a positive integer. The rate of this code is  $\Theta\left(\frac{1}{c^2}\right)$ . This is because, if we express (6.70) as  $R = \frac{1}{\alpha} \left(\frac{1}{c^2} - \frac{1}{\sqrt{|\mathbf{Q}|-1}}\right)$  for an appropriate  $\alpha > 0$ , then both

$$\lim_{c \rightarrow \infty} \frac{\frac{1}{\alpha} \left(\frac{1}{c^2} - \frac{1}{c^{\beta-1}}\right)}{\frac{1}{c^2}} < \infty \text{ and } \lim_{c \rightarrow \infty} \frac{\frac{1}{c^2}}{\frac{1}{\alpha} \left(\frac{1}{c^2} - \frac{1}{c^{\beta-1}}\right)} > 0$$

and the proof is finished. □

For the inner code  $C_i$  we will take a  $c$  binary separating code of size  $c^\beta$ . By the claim above, the code has rate  $O(\frac{1}{2^{2^c}})$ . The theorem follows by Lemma 6.5.6. □

**Comparison with previous works and applications to fingerprinting**

Let us show explicit rates of our results. We

Take for instance values for  $c = 2$ ,  $c = 4$  and  $c = 6$ . According to Lemma 6.5.7 and the proof of Theorem 6.5.4, for the outer code, we take  $|\mathbf{Q}| = c^6$ . That means that sizes f

There are not many attempts in the literature to construct separating codes. To see the relevance of the results in this paper, check for instance [170] where explicit constructions of secure and almost secure frameproof codes. Secure frameproofness is a name given to separation in the crypto literature.

In [170] for separating codes of size  $M = 10^2$  the authors obtain codes with rates  $R_2 = 2.2967E-6$ ,  $R_4 = 1.4018E-10$  and  $R_6 = 1.5210E-14$ . For separating codes of size  $M = 10^8$  they obtain codes of rate  $R_2 = 5.7417E-7$ ,  $R_4 = 3.5045E-11$  and  $R_6 = 3.8026E-15$ . For almost secure frameproof codes the rates are  $R_2 = 2.1598E-3$ ,  $R_4 = 9.1699E-6$  and  $R_6 = 6.3793E-8$ . According to Theorem 6.5.4 we have  $R_2 = 0.00390625$ ,  $R_4 = 1.5258E-5$  and  $R_6 = 1.88380E-7$

Code	$c = 2$	6	8
Theorem 6.5.2 (exponential complexity)	6.422 - 2	9.161 - 3	1.616 - 3
Theorem 6.5.4 (polynomial complexity)	9.258 - 4	6.348 - 6	1.004 - 7
Separating (Moreira et al. [170])	1.422 - 1	1.703 - 2	3.001 - 3
Almost secure frameproof (Moreira et al. [170])	2.075 - 1	6.422 - 2	2.328 - 2

Table 6.1: Lower bounds on the rate of some  $q$ -ary codes.

The extensive work by Barg et al. [13], is a deep discussion of fingerprinting codes using separating codes. In that paper there is a large number of existence results about fingerprinting codes, but unfortunately there are almost no explicit constructions other than for the case  $t = 2$ .

To see how our results fit in we take, for instance, the following results.

**Corollary 6.5.5** (Corollary 4.3 [13]). *For any fixed  $c$ , any  $q \geq (c^2 + 2)^2$  that is an even power of a prime, and any rate  $R < R_{2c}R_0(W)$ , where  $R_0(W)$  is the root of the equation*

$$R(W) \log_2 q = D \left( \frac{1}{c} - c \left( R(W) - \frac{1}{\sqrt{q} - 1} \right) \parallel \frac{c-1}{q-1} \right)$$

and

$$R_{2c} = -\frac{\log_2(1 - 2^{-(2c-1)})}{2c-1}$$

a binary fingerprinting code of length  $N$  and size  $2^{RN}$  constructed by concatenating a fixed inner  $(m, q)$  code  $V$  with the  $c$ -separating property and AG codes  $W$  of rate  $R(W) \leq R_0(W)$  and growing length  $N_0$ , identifies one traitor with exponentially falling error probability.

**Corollary 6.5.6** (Corollary 5.2 [13]). : *Let  $R_c^{(s)}$  be the maximum achievable rate of  $c$ -separating codes. For any rate  $R$ ,  $0 < R < \frac{R_c^{(s)}}{c^3}$ , there exists a sequence of  $c$ -secure fingerprinting codes of length  $n$  and size  $2^{RN}$  that allow polynomial-time identification with error probability falling exponentially with the code length  $N$ .*

The algorithms presented in Section [6.5.2] provide constructions of polynomial complexity to approach these results.

# Chapter 7

## Aggregating Abstract and Implicitly given Domains

In this chapter, we present our results in the field of Judgement Aggregation. We begin with characterizing various abstract domains in terms of the aggregators they admit in Sec. [7.1](#). In Sec. [7.2](#) we syntactically characterize integrity constraints whose domains have interesting properties. Finally, in Sec. [7.1](#), we discuss the computational complexity of deciding if a domain (whether it is abstract or is given implicitly) can be aggregator non-dictatorially.

### 7.1 Characterizations for Abstract Domains

In this section, we characterize possibility domains in the Boolean and non-Boolean domains (Subsec. [7.1.1](#)). Then, we proceed to characterize totally blocked domains (Subsec. [7.1.2](#)) and, finally, we define and characterize uniform possibility domains (Subsec. [7.1.3](#)).

#### 7.1.1 Characterization of Possibility Domains

Our first result is a necessary and sufficient condition for a set of feasible voting patterns to be a possibility domain.

**Theorem 7.1.1.** *Let  $X$  be a set of feasible voting patterns. The following statements are equivalent.*

1.  $X$  is a possibility domain.
2.  $X$  admits a non-dictatorial binary aggregator or it admits a majority aggregator or it admits a minority aggregator.

Theorem [7.1.1](#) is stronger than Corollary [5.2.1](#) because, unlike Corollary [5.2.1](#), it gives explicit information about the nature of the components  $f_j$  of non-dictatorial ternary aggregators  $F = (f_1, \dots, f_n)$ , when the components are restricted to a two-element subset  $B_j \subseteq X_j$  of the set of positions on issue  $j$ , information that is necessary to relate results in aggregation theory with complexity theoretic results (besides the three projections, there are 61 supportive ternary functions on a two element set). Observe also that if  $F = (f_1, \dots, f_n)$  is a binary aggregator, then every component  $f_j$  is necessarily a projection function or the function  $\wedge$  or the function  $\vee$ , when restricted to a two-element subset  $B_j \subseteq X_j$  (identified with the set  $\{0, 1\}$ ). So, for binary aggregators, the information about the nature of their components is given *gratis*.

Only the direction  $1 \implies 2$  of Theorem [7.1.1](#) requires proof. Towards this goal, we first introduce a new notion, that of monomorphic aggregators, and give three lemmas, which we then use to prove Theorem [7.1.1](#).

Let  $X$  be a set of feasible voting patterns and let  $F = (f_1, \dots, f_n)$  be a  $k$ -ary aggregator for  $X$ .

**Definition 7.1.1.** *We say that  $F$  is locally monomorphic if for all indices  $i$  and  $j$  with  $1 \leq i, j \leq n$ , for all two-element subsets  $B_i \subseteq X_i$  and  $B_j \subseteq X_j$ , for every bijection  $g : B_i \mapsto B_j$ , and for all column vectors  $\mathbf{x}_i = (x_i^1, \dots, x_i^k) \in B_i^k$ , we have that*

$$f_j(g(x_i^1), \dots, g(x_i^k)) = g(f_i(x_i^1, \dots, x_i^k)).$$

Intuitively, the above definition says that, no matter how we identify the two elements of  $B_i$  and  $B_j$  with 0 and 1, the restrictions  $f_i \upharpoonright B_i$  and  $f_j \upharpoonright B_j$  are equal as functions. Notice that in the definition we are allowed to have  $i = j$ , which implies that if in a specific  $B_j$  we interchange the values 0 and 1 in the arguments of  $f_j \upharpoonright B_j$ , then the bit that gives the image of  $f_j \upharpoonright B_j$  is flipped.

It follows immediately from the definitions that if an aggregator is dictatorial, then it is locally monomorphic. For binary aggregators, the converse is true. Indeed, assume that  $F = (f_1, \dots, f_n)$  is a binary locally monomorphic aggregator for  $X$ . We claim that  $F = (f_1, \dots, f_n)$  is dictatorial on  $X$ .

To see this, fix a coordinate  $f_i$  and consider a pair  $(a, b) \in X_i^2$  with  $a \neq b$ . By conservativeness, either  $f_i(a, b) = a$  or  $f_i(a, b) = b$ . We claim that if  $f_i(a, b) = a$ , then  $(f_1, \dots, f_n) = (\text{pr}_1^2, \dots, \text{pr}_1^2)$ , while if  $f_i(a, b) = b$ , then  $(f_1, \dots, f_n) = (\text{pr}_2^2, \dots, \text{pr}_2^2)$ . To see this, consider a coordinate  $f_j$  and a pair  $(a', b') \in X_j^2$  with  $a' \neq b'$ . Let  $g : \{a, b\} \rightarrow \{a', b'\}$  be the bijection  $g(a) = a'$  and  $g(b) = b'$ . Since  $F = (f_1, \dots, f_n)$  is locally monomorphic, we have that  $f_j(a', b') = f_j(g(a), g(b)) = g(f_i(a, b)) = g(a) = a'$ , hence  $(f_1, \dots, f_n) = (\text{pr}_1^2, \dots, \text{pr}_1^2)$ . The case where  $f_i(a, b) = b$  is entirely analogous. As we shall see next, a ternary locally monomorphic aggregator need not be dictatorial. In fact, majority aggregators and minority aggregators are locally monomorphic, but, of course, they are not dictatorial.

**Example 7.1.1.** *Let  $X$  be a set of feasible voting patterns that admits a ternary aggregator  $F = (f_1, \dots, f_n)$  that is either a majority or a minority aggregator. Then  $F$  is locally monomorphic.*

*Indeed, suppose that  $F$  is a minority aggregator, i.e. for every  $j$  with  $1 \leq j \leq n$  and every two-element set  $B_j \subseteq X_j$ , we have that  $f_j \upharpoonright_{B_j} = \oplus$ . Let  $i, j$  be such that  $1 \leq i, j \leq n$ , let  $B_i = \{a, b\} \subseteq X_i$ , and let  $B_j = \{c, d\} \subseteq X_j$  (we make no assumption for the relation, if any, between  $a, b, c, d$ ). There are exactly two bijections  $g$  and  $g'$  from  $B_i$  to  $B_j$ , namely,*

$$g(a) = c \text{ and } g(b) = d$$

$$g'(a) = d \text{ and } g'(b) = c$$

*Suppose that  $(x, y, z)$  is a triple with  $x, y, z \in B_i$ . Since  $|B_i| = |B_j| = 2$ , it holds that  $f_i \upharpoonright_{B_i} = \oplus$  and  $f_j \upharpoonright_{B_j} = \oplus$ . Without loss of generality, suppose that  $x = a, y = z = b$ . Then*

$$\begin{aligned} f_j(g(x), g(y), g(z)) &= f_j(c, d, d) \\ &= \oplus(c, d, d) = c \\ &= g(a) = g(\oplus(a, b, b)) \\ &= g(f_i(x, y, z)). \end{aligned}$$

*An analogous statement holds for  $g'$ . Since  $i, j$  were arbitrary, we conclude that  $\bar{f}$  is locally monomorphic.*

*The proof for the case when  $\bar{f}$  is a majority aggregator is similar.  $\diamond$*

We now present the first lemma needed in the proof of Theorem 7.1.1, which gives a sufficient condition for all aggregators of all arities to be locally monomorphic.

**Lemma 7.1.1.** *Let  $X$  be a set of feasible voting patterns. If every binary aggregator for  $X$  is dictatorial on  $X$ , then, for every  $k \geq 2$ , every  $k$ -ary aggregator for  $X$  is locally monomorphic.*

*Proof.* Under the hypothesis that all binary aggregators are dictatorial, the conclusion is obviously true for binary aggregators. For the induction step, suppose that the conclusion is true for all  $(k-1)$ -ary aggregators, where  $k \geq 3$ . Consider a  $k$ -ary aggregator  $F = (f_1, \dots, f_n)$  and a pair  $(B_i, B_j)$  of two-element subsets  $B_i \subseteq X_i$  and  $B_j \subseteq X_j$ . To render the notation less cumbersome, we will take the liberty to denote the two elements of both  $B_i$  and  $B_j$  as 0 and 1. Assume now, towards a contradiction, that there are a column-vector  $(a^1, \dots, a^k)$  with  $a^i \in \{0, 1\}$ ,  $1 \leq i \leq k$ , a “copy” of this vector belonging to  $B_i^k$ , another copy belonging to  $B_j^k$ , such that  $f_i(a^1, \dots, a^k) \neq f_j(a^1, \dots, a^k)$ . Since  $k \geq 3$ , by the pigeonhole principle applied to two holes and at least three pigeons, there is a pair of coordinates of  $(a^1, \dots, a^k)$  that coincide. Without loss of generality, assume that these two coordinates are the two last ones, i.e.,  $a^{k-1} = a^k$ . We now define an  $(k-1)$ -ary aggregator  $G = (g_1, \dots, g_m)$  as follows: given  $k-1$  voting patterns  $(x_1^i, \dots, x_n^i)$ ,  $i = 1, \dots, k-1$ , define  $k$  voting patterns by just repeating the last one and then for all  $k = 1, \dots, n$ , define

$$g_k(x_l^1, \dots, x_l^{k-1}) = f_l(x_l^1, \dots, x_l^{k-1}, x_l^{k-1}).$$

It is straightforward to verify that  $G$  is an  $(k-1)$ -ary aggregator on  $X$  that is not locally monomorphic, which contradicts the inductive hypothesis.  $\square$

**Remark 7.1.1.** *The preceding argument generalizes to arbitrary cardinalities in the following way: if every aggregator of arity at most  $s$  on  $X$  is dictatorial, then every aggregator on  $X$  is  $s$ -locally monomorphic, meaning that for every  $t \leq s$  and for all sets  $B_j \subseteq X_j$  of cardinality  $t$ , the functions  $f_j \upharpoonright B_j$  are all equal up to bijections between the  $B_j$ 's.*

Next, we state a technical lemma whose proof was inspired by a proof in Dokow and Holzman [77, Proposition 5].

**Lemma 7.1.2.** *Assume that for all integers  $k \geq 2$  and for every  $k$ -ary aggregator  $F = (f_1, \dots, f_n)$ , there is an integer  $d \leq k$  such that for every integer  $j \leq n$  and every two-element subset  $B_j \subseteq X_j$ , the restriction  $f_j \upharpoonright B_j$  is equal to  $\text{pr}_d^k$ , the  $k$ -ary projection on the  $d$ -th coordinate. Then for all integers  $k \geq 2$  and for every  $k$ -ary aggregator  $F = (f_1, \dots, f_n)$  and for all*



$s \geq 2$ , there is an integer  $d \leq k$  such that for every integer  $j \leq n$  and every subset  $B_j \subseteq X_j$  of cardinality at most  $s$ , the restriction  $f_j \upharpoonright B_j$  is equal to  $\text{pr}_d^k$ .

*Proof.* The proof will be given by induction on  $s$ . The induction basis  $s = 2$  is given by hypothesis. Before delving into the inductive step of the proof and for the purpose of making the intuition behind it clearer, let us mention the following fact whose proof is left to the reader. This fact illustrates the idea for obtaining a non-dictatorial aggregator of lower arity from one of higher arity.

Let  $A$  be a set and let  $f : A^3 \mapsto A$  be a supportive function such that if among  $x_1, x_2, x_3$  at most two are different, then  $f(x_1, x_2, x_3) = x_1$ . Assume also that there exist pairwise distinct  $a_1, a_2, a_3$  such that  $f(a_1, a_2, a_3) = a_2$ ; in the terminology of universal algebra,  $f$  is a *semi-projection*, but not a projection. Define  $g(x_1, x_2) = f(x_1, f(x_1, x_2, a_3), a_3)$ . Then, by distinguishing cases as to the value of  $f(x_1, x_2, a_3)$ , it is easy to verify that  $g$  is supportive; however,  $g$  is not a projection function because  $g(a_1, a_2) = a_2$ , whereas  $g(a_1, a_3) = a_1$ .

For the inductive step of the proof of Lemma [7.1.2](#), we assume that for every  $k \geq 2$  and every  $k$ -ary aggregator  $F = (f_1, \dots, f_n)$ , there is a  $d \leq k$  such that for every integer  $j \leq n$  and every subset  $B_j \subseteq X_j$  with at most  $s - 1$  elements, the restriction  $f_j \upharpoonright B_j$  is equal to  $\text{pr}_d^k$ . Fix such an  $k$ -ary aggregator  $F$  and fix an integer  $d$ , obtained by applying the induction hypothesis to  $s - 1$  and  $F$ . Assume, without loss of generality that  $d = 1$ . We will show that for every  $j \leq n$  and for every subset  $B_j \subseteq X_j$  of cardinality at most  $s$ , we have that  $f_j \upharpoonright B_j = \text{pr}_1^k$ , the  $k$ -ary projection function on  $d = 1$ . We may assume that  $s \leq k$ , lest the induction hypothesis applies.

Assume towards a contradiction that there exists an integer  $j_0 \leq n$  and row vectors  $\mathbf{a}^1, \dots, \mathbf{a}^k$  in  $X$  such that the set  $B_{j_0} = \{a_{j_0}^1, \dots, a_{j_0}^k\}$  has cardinality  $s$  and

$$f_{j_0}(a_{j_0}^1, \dots, a_{j_0}^k) \neq a_{j_0}^1. \quad (7.1)$$

By supportiveness, there exists  $i_0 \in \{2, \dots, k\}$  such that

$$f_{j_0}(a_{j_0}^1, \dots, a_{j_0}^k) = a_{j_0}^{i_0}. \quad (7.2)$$

Let  $\{l_1, \dots, l_s\}$  be a subset of  $\{1, \dots, k\}$  of cardinality  $s$  such that  $a_{j_0}^{l_1}, \dots, a_{j_0}^{l_s}$  are pairwise distinct. Obviously, if  $i \notin \{l_1, \dots, l_s\}$ , then there exists  $t \in \{1, \dots, s\}$  such that  $a_{j_0}^i = a_{j_0}^{l_t}$ . So, by renumbering we may assume that  $l_1 = 1, \dots, l_s = s$  and  $i_0 = 2$ . Recall that  $s \geq 3$ . Let  $B_{j_0}^- = \{a_{j_0}^1, \dots, a_{j_0}^{s-1}\}$ .

We define an  $(s-1)$ -ary aggregator  $F^- = (f_1^-, \dots, f_n^-)$  as follows: for  $j = 1, \dots, n$  and  $(x_j^1, \dots, x_j^{s-1}) \in X_j^{s-1}$ , first define  $(y_j^1, \dots, y_j^k) \in X_j^k$  as follows:

$$y_j^i = \begin{cases} x_j^i & \text{for } i = 1, \dots, s-1, \\ a_j^s & \text{if } i = s, \\ a_j^s & \text{if } i > s \text{ and } a_{j_0}^i = a_{j_0}^s, \\ x_j^t & \text{for the least } t < s \text{ such that } a_{j_0}^i = a_{j_0}^t, \text{ if } i > s \text{ and } a_{j_0}^i \neq a_{j_0}^s, \end{cases} \quad (7.3)$$

then we set:

$$\hat{y}_j^i = \begin{cases} y_j^i & \text{if } a_{j_0}^i \neq a_{j_0}^2, \\ f_j(y_j^1, \dots, y_j^k) & \text{if } a_{j_0}^i = a_{j_0}^2, \end{cases} \quad (7.4)$$

and finally define:

$$f_j^-(x_j^1, \dots, x_j^{s-1}) = f_j(\hat{y}_j^1, \dots, \hat{y}_j^k).$$

First observe that  $F^-$  is supportive. Indeed this follows from the observation that because of the inductive hypothesis,  $f_j^-$  can never take the value  $a_{j_0}^s$ , if  $(x_j^1, \dots, x_j^{s-1}) \in X_j^{s-1}$ , but  $a_{j_0}^s \neq x_j^1$ . Then observe that  $F^- = (f_1^-, \dots, f_n^-)$  is an aggregator on  $X$ , because, in case  $x^i \in X$  for all  $i = 1, \dots, s-1$  then all row vectors  $y^1, \dots, y^k$  defined above belong to  $X$  (each is either some  $x^i$  or some  $a^i$ ).

It is obvious that

$$f_{j_0}^-(a_{j_0}^1, \dots, a_{j_0}^{s-1}) = f_{j_0}(a_{j_0}^1, \dots, a_{j_0}^k) = a_{j_0}^2.$$

Also, let  $x_{j_0}^1, \dots, x_{j_0}^{s-1} \in B_{j_0}^-$  be such that  $x_{j_0}^1 \neq x_{j_0}^2$  and

$$2 \leq |\{x_{j_0}^1, \dots, x_{j_0}^{s-1}\}| \leq s-2.$$

It is easy to see that for the corresponding  $\hat{y}_{j_0}^i$ , it holds that

$$2 \leq |\{\hat{y}_{j_0}^1, \dots, \hat{y}_{j_0}^k\}| \leq s-1.$$

It follows that

$$f_{j_0}^-(x_{j_0}^1, \dots, x_{j_0}^{s-1}) = f_{j_0}(\hat{y}_{j_0}^1, \dots, \hat{y}_{j_0}^k) = \hat{y}_{j_0}^1 = x_{j_0}^1 \neq x_{j_0}^2.$$

Therefore,  $f_{j_0}^- \upharpoonright B_{j_0}^-$  cannot be a projection function, which contradicts the inductive hypothesis (assumed to hold for every  $F$ ); this concludes the proof of Lemma [7.1.2](#).  $\square$

The proof of the next lemma is straightforward.

**Lemma 7.1.3.** *Let  $X$  be a set of feasible voting patterns. For every  $j$  with  $1 \leq j \leq n$  and every subset  $B_j \subseteq X_j$ , the set  $\mathcal{C}_{B_j}$  of the restrictions  $f_j \upharpoonright B_j$  of the  $j$ -th components of aggregators  $F = (f_1, \dots, f_m)$  for  $X$  is a clone on  $B_j$ .*

As we saw in Ch. 4, one of Post's main findings is that if  $\mathcal{C}$  is a clone of conservative functions on a two-element set, then either  $\mathcal{C}$  contains only projection functions or  $\mathcal{C}$  contains one of the following operations: the binary operation  $\wedge$ , the binary operation  $\vee$ , the ternary operation  $\oplus$ , the ternary operation  $\text{maj}$ .

Using all of the above, we are now ready to prove Theorem 7.1.1.

*Proof of Theorem 7.1.1.* As stated earlier, only the direction  $1 \implies 2$  requires proof. In the contrapositive, we will prove that if  $X$  does not admit a majority or a minority aggregator, and it does not admit a non-dictatorial binary aggregator, then  $X$  does not have an  $k$ -ary non-dictatorial aggregator, for any  $k$ . Towards this goal, and assuming that  $X$  is as stated, we will first show that the hypothesis of Lemma 7.1.2 holds. Once this is established, the conclusion will follow from Lemma 7.1.2 by taking  $s = \max\{|X_j| : 1 \leq j \leq n\}$ .

Given  $j \leq n$  and a two-element subset  $B_j \subseteq X_j$ , consider the clone  $\mathcal{C}_{B_j}$ . If  $\mathcal{C}_{B_j}$  contained one of the binary operations  $\wedge$  or  $\vee$ , then  $X$  would have a binary non-dictatorial aggregator, a contradiction. If, on the other hand,  $\mathcal{C}_{B_j}$  contained the ternary operation  $\oplus$  or the ternary operation  $\text{maj}$ , then, by Lemma 7.1.1,  $X$  would admit a minority or a majority aggregator, a contradiction as well. So, by Post's result, all elements of  $\mathcal{C}_{B_j}$ , no matter what their arity is, are projection functions. By Lemma 7.1.1 again, since  $X$  has no binary non-dictatorial aggregator, we have that for every  $k$  and for every  $k$ -ary aggregator  $F = (f_1, \dots, f_n)$ , there exists an integer  $d \leq m$  such that for every  $j \leq n$  and every two-element set  $B_j \subseteq X_j$ , the restriction  $f_j \upharpoonright B_j$  is equal to  $\text{pr}_d^k$ , the  $k$ -ary projection on the  $d$ -th coordinate. This concludes the proof of Theorem 7.1.1.  $\square$

In the case of the Boolean framework, Theorem 7.1.1 takes the stronger form of Theorem 7.1.2 below. Although this result for the Boolean framework is implicit in Dokow and Holzman [76], we give an independent proof.

**Theorem 7.1.2** (Dokow and Holzman). *Let  $X \subseteq \{0, 1\}^n$  be a set of feasible voting patterns. The following statements are equivalent.*

1.  $X$  is a possibility domain.

2.  $X$  is affine (i.e.,  $X$  admits a minority aggregator) or  $X$  admits a non-dictatorial binary aggregator.

*Proof.* Only the direction  $1 \implies 2$  requires proof. Assume that  $X$  is a possibility domain in the Boolean framework. By Theorem 7.1.1,  $X$  admits either a majority or a minority aggregator or  $X$  has non-dictatorial binary aggregator. Since we are in the Boolean framework, this means that  $X$  is affine or  $X$  is bijunctive or  $X$  has a non-dictatorial binary aggregator. If  $X$  has at most two elements, then  $X$  is closed under  $\oplus$ , hence  $X$  is affine. So, it suffices to show that if  $X$  is bijunctive and has at least three elements, then  $X$  has a non-dictatorial binary aggregator. To prove the latter, fix an element  $\mathbf{a} = (a_1, \dots, a_n) \in X$ . Define the following binary aggregator, where  $\mathbf{x} = (x_1, \dots, x_n)$  and  $\mathbf{y} = (y_1, \dots, y_n)$  are arbitrary elements of  $X$ :

$$F^{\mathbf{a}}(\mathbf{x}, \mathbf{y}) = (\text{maj}(x_1, y_1, a_1), \dots, \text{maj}(x_n, y_n, a_n)). \quad (7.5)$$

First, observe that  $F^{\mathbf{a}}$  is indeed an aggregator for  $X$ . Since  $X$  is closed under  $\text{maj}$ , all we have to prove is that  $F^{\mathbf{a}}$  is supportive. But this is obvious, because, for  $j \leq n$ , if  $x_j = a_j$  or  $y_j = a_j$ , then  $\text{maj}(x_j, y_j, a_j) = x_j$  or  $\text{maj}(x_j, y_j, a_j) = y_j$ . If  $x_j \neq a_j$  and  $y_j \neq a_j$ , then  $x_j = y_j$ , hence  $\text{maj}(x_j, y_j, a_j) = x_j = y_j$ .

Now, consider an  $\mathbf{a} \in X$  such that at some coordinate  $i$ ,  $a_i = 1$  and at some coordinate  $j$ ,  $a_j = 0$  (such  $\mathbf{a}$  exists, because  $X$  has at least three elements). Observe that  $F_i^{\mathbf{a}} = \vee$ , by equation 7.5 and because  $a_i = 1$ . Similarly,  $F_j^{\mathbf{a}} = \wedge$ , because  $a_j = 0$ . Therefore  $F_i^{\mathbf{a}} \neq F_j^{\mathbf{a}}$ .  $\square$

## 7.1.2 Characterization of Total Blockedness

As discussed in the preceding section, much of the earlier work on possibility domains used the notion of a set being totally blocked. Our next result characterizes this notion in terms of binary aggregators and, in many respects, “explains” the role of this notion in the earlier results about possibility domains.

We begin by giving the precise definition of what it means for a set  $X$  of feasible voting patterns to be totally blocked. We will follow closely the notation and terminology used by Dokow and Holzman [77].

Let  $X$  be a set of feasible voting patterns.

- Given subsets  $B_j \subseteq X_j, j = 1, \dots, n$ , the product  $B = \prod_{j=1}^n B_j$  is called a *sub-box*. It is called a *2-sub-box* if  $|B_j| = 2$ , for all  $j$ .

Elements of a box  $B$  that belong also to  $X$  will be called *feasible evaluations within  $B$*  (in the sense that each issue  $j = 1, \dots, n$  is “evaluated” within  $B$ ).

- Let  $K$  be a subset of  $\{1, \dots, n\}$  and let  $\mathbf{x}$  be a tuple in  $\prod_{j \in K} B_j$ .

We say that  $\mathbf{x}$  is a *feasible partial evaluation within  $B$*  if there exists a feasible evaluation  $\mathbf{y}$  within  $B$  that extends  $\mathbf{x}$ , i.e.,  $x_j = y_j$ , for all  $j \in K$ ; otherwise, we say that  $\mathbf{x}$  is an *infeasible partial evaluation within  $B$* .

We say that  $\mathbf{x}$  is a  *$B$ -Minimal Infeasible Partial Evaluation ( $B$ -MIPE)* if  $\mathbf{x}$  is an infeasible partial evaluation within  $B$  and if for every  $j \in K$ , there is a  $b_j \in B_j$  such that changing the  $j$ -th coordinate of  $\mathbf{x}$  to  $b_j$  results into a feasible partial evaluation within  $B$ .

- We define a directed graph  $G_X$  as follows.

The vertices of  $G_X$  are the pairs of *distinct* elements  $u, u'$  in  $X_j$ , for all  $j = 1, \dots, n$ . Each such vertex is denoted by  $uu'_j$ .

Two vertices  $uu'_s, vv'_t$  with  $s \neq t$  are connected by a directed edge from  $uu'_s$  to  $vv'_t$  if there exists a 2-sub-box  $B = \prod_{j=1}^n B_j$ , a set  $K \subseteq \{1, \dots, n\}$ , and a  $B$ -MIPE  $x = (x_j)_{j \in K}$  such that  $s, t \in K$  and  $B_s = \{u, u'\}$  and  $B_t = \{v, v'\}$  and  $x_s = u$  and  $x_t = v'$ . Each such directed edge is denoted by  $uu'_s \xrightarrow[B, \mathbf{x}, K]{} vv'_t$  (or just  $uu'_s \rightarrow vv'_t$ , in case  $B, \mathbf{x}, K$  are understood from the context).

Notice that  $uu'_s \rightarrow vv'_t$  if and only if  $v'v_t \rightarrow u'u_s$ .

We now give the following definition:

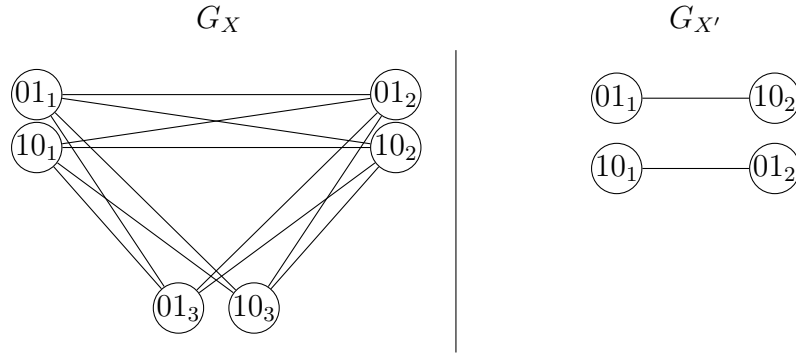
**Definition 7.1.2** (Dokow and Holzman [\[77\]](#)). *We say that  $X$  is totally blocked if the graph  $G_X$  is strongly connected, i.e., every two distinct vertices  $uu'_s, vv'_t$  are connected by a directed path (this must hold even if  $s = t$ ).*

**Example 7.1.2.** *Let  $X = \{(0, 0, 1), (0, 1, 0), (1, 0, 0), (1, 1, 1)\}$  and  $X' = \{(0, 1), (1, 0)\}$ .*

*Both  $G_X$  and  $G_{X'}$  have two vertices for each issue  $j$ , namely  $01_j$  and  $10_j$ , where  $G_X$  has  $j = 1, 2, 3$  and  $G_{X'}$  has  $j = 1, 2$ . In the figures below, we use undirected edges between two vertices  $uu'_s$  and  $vv'_t$  to denote the existence of both  $uu'_s \rightarrow vv'_t$  and  $vv'_t \rightarrow uu'_s$ .*

Since  $X$  is in the Boolean framework, the only 2-sub-box  $B$  is  $X_1 \times X_2 \times X_3$ . The  $B$ -MIPES of  $X$  are  $(0, 0, 0)$ ,  $(0, 1, 1)$ ,  $(1, 0, 1)$  and  $(1, 1, 0)$ . Consider  $01_1$ ,  $01_2$  of  $G_X$ . Due to  $(0, 1, 1)$ ,  $01_1 \rightarrow 01_2$  and due to  $(1, 0, 1)$ ,  $01_2 \rightarrow 01_1$ . Also, due to  $(0, 0, 0)$ ,  $01_i \rightarrow 10_j$ ,  $\forall i, j \in \{1, 2, 3\} : i \neq j$ . The rest of the cases are left to the reader.

Observe that  $G_X$  is strongly connected and that it admits no binary non-dictatorial aggregator, as expected by Theorem 7.1.3 below, but it admits the ternary minority aggregator.



On the other hand, the only 2-sub-box  $B'$  of  $X'$  is  $X'_1 \times X'_2$ , since it is also in the Boolean framework. The  $B'$ -MIPES of  $X'$  are  $(0, 0)$  and  $(1, 1)$ . Easily now,  $01_1 \leftrightarrow 10_2$  and  $10_1 \leftrightarrow 01_2$ .

Observe that  $G_{X'}$  is not strongly connected (it is not even connected) and, as expected by Theorem 7.1.3 below, it admits binary non-dictatorial aggregators, namely,  $(\wedge, \vee)$  and  $(\vee, \wedge)$ .  $\diamond$

This notion is a generalization to the case where the domain  $\mathcal{D}$  is allowed to have an arbitrary cardinality of a corresponding notion for the Boolean framework, originally given in [174].

To give the intuition behind the above rather technical definition, assume that we are in the Boolean case, so the only 2-sub-box is  $\prod_{j=1}^n X_j$ . Observe that a MIPES  $\mathbf{x}$  is a vector of votes on *some* of the issues, which however *cannot* be extended to a vector of votes on all issues in  $X$ , i.e. cannot be extended to a “rational” voting pattern on all issues, and is a minimal partial such vector, in the sense that deleting the vote of a single issue from  $\mathbf{x}$ , we get a partial vector of votes that can be extended to a total rational one. Now observe that if  $uu'_s, vv'_t$  with  $s \neq t$  are connected by an edge, then there

is a minimal way to fix the votes on some issues other than  $s, t$  so that any rational total voting pattern that takes these fixed values and has the value  $u$  on issue  $s$ , should take the value  $v$  on issue  $t$ . Therefore as Dokow and Holzman [75] write “Roughly speaking, it requires that the limitations on feasibility embodied in the set  $X$  make it possible to deduce any position on any issue from any position on any issue, via a chain of deductions.”

We are now ready to state the following result, which — quite remarkably we believe— characterizes total blockedness as a weak form of impossibility, for any number of issues and any possible set of votes for each issue.

**Theorem 7.1.3.** *Let  $X$  be a set of feasible voting patterns. The following statements are equivalent.*

1.  $X$  is totally blocked.
2.  $X$  has no non-dictatorial binary aggregator.

Observe that Theorem [7.1.2] is also an immediate consequence of Theorem [5.2.1] and Theorem [7.1.3]. In view of Theorem [5.2.2] by Dokow and Holzman [77], only the direction  $1 \implies 2$  of Theorem [7.1.3] requires proof. Nevertheless, we prove both directions of Theorem [7.1.3] for completeness.

*Proof.* We start with direction  $1 \implies 2$ .

Consider at first two vertices  $uu'_s, vv'_t$  of  $G_X$  (with  $s \neq t$ ) connected by an edge  $uu'_s \rightarrow vv'_t$ . Then there exists a 2-sub-box  $B = \prod_{j=1}^n B_j$  with  $B_s = \{u, u'\}$  and  $B_t = \{v, v'\}$  and a  $B$ -MIPE  $\mathbf{x} = (x_j)_{j \in K}$  such that  $\{s, t\} \subseteq K$  and  $x_s = u, x_t = v'$ .

**Claim 7.1.1.** *For every binary aggregator  $F = (f_1, \dots, f_n)$  of  $X$ , it holds that if  $f_s(u, u') = u$ , then  $f_t(v, v') = v$ .*

To prove the Claim, first observe that by the minimality of  $\mathbf{x}$  within  $B$  if we flip  $x_s$  from  $u$  to  $u'$  or if we flip  $x_t$  from  $v'$  to  $v$ , then we get, in both cases, respective feasible evaluations within  $B$ . Therefore, there are two total evaluations  $e$  and  $e'$  in  $X \cap B$  such that

- $e_s = u'$  and
- $e_p = x_p$  for  $p \in K, p \neq s$  (in particular  $e_t = v'$ ),

and

- $e'_t = v$  and
- $e'_p = x_p$  for  $p \in K, p \neq t$  (in particular  $e'_s = u$ ).

If we assume, towards a contradiction, that  $f_s(u, u') = u$  and  $f_t(v, v') = v'$ , we immediately have that the evaluation

$$F(e, e') := (f_1(e_1, e'_1), \dots, f_n(e_n, e'_n))$$

extends  $(x_j)_{j \in K}$ , contradicting the latter's infeasibility within  $B$ . This completes the proof of the Claim and we now return to the proof of Theorem [7.1.3](#).

From the Claim we get that if  $uu'_s \rightarrow\rightarrow vv'_t$  and  $f_s(u, u') = u$ , then  $f_t(v, v') = v$  (even if  $s = t$ ), where  $uu'_s \rightarrow\rightarrow vv'_t$  means that there is path from  $uu'_s$  to  $vv'_t$  in the graph  $G_X$ . Also, since by supportiveness  $f_t(v, v') \in \{v, v'\}$ , we have that if  $vv'_t \rightarrow\rightarrow uu'_s$  and  $f_s(u, u') = u'$ , then  $f_t(v, v') = v'$ . From this, it immediately follows that if  $G_X$  is strongly connected, then every binary aggregator of  $X$  is dictatorial.

We will now prove Direction 2  $\implies$  1 of Theorem [7.1.3](#), namely, that if  $X$  is not totally blocked, then there is a non-dictatorial binary aggregator (this part is contained in [\[77\]](#), Theorem 2] – Theorem [5.2.2](#) above). Since  $G_X$  is not strongly connected, there is a partition of the vertices of  $G_X$  into two mutually disjoint and non-empty subsets  $V_1$  and  $V_2$  so that there is no edge from a vertex of  $V_1$  towards a vertex in  $V_2$ . We now define a  $F = (f_1, \dots, f_n)$ , where  $f_s : A_s^2 \mapsto A_s$ , as follows:

$$f_s(u, u') = \begin{cases} u & \text{if } uu'_s \in V_1 \text{ and } u \neq u', \\ u' & \text{if } uu'_s \in V_2 \text{ and } u \neq u', \\ u & \text{if } u = u'. \end{cases} \quad (7.6)$$

In other words, for two differing values  $u$  and  $u'$  in  $X_s$ , the function  $f_s$  is defined as the projection on the first coordinate if  $uu'_s \in V_1$ , and as the projection onto the second coordinate if  $uu'_s \in V_2$ ; we also define  $f_k(u, u) = u$  if  $u = u'$ .

Notice that  $F$  is non-dictatorial, because  $V_1$  and  $V_2$  are not empty.

All that remains to be shown is that  $X$  is closed under  $F$ , i.e., if  $e = (e_1, \dots, e_n), e' = (e'_1, \dots, e'_n) \in X$  are two total feasible evaluations, then

$$\bar{f}(e, e') := (f_1(e_1, e'_1), \dots, f_n(e_n, e'_n)) \in X. \quad (7.7)$$



Let

$$L = \{j = 1, \dots, n \mid e_j \neq e'_j\}.$$

For an arbitrary  $j \in L$ , define  $\text{vertex}_j(e, e')$  to be the vertex  $uu'_j$  of  $G_X$ , where  $u = e_j$  and  $u' = e'_j$ .

If now  $F(e, e') = e$  or if  $F(e, e') = e'$ , then obviously (7.7) is satisfied. So assume that

$$F(e, e') \neq e \text{ and } F(e, e') \neq e'. \quad (7.8)$$

Also, towards showing (7.7) by contradiction, assume

$$F(e, e') \notin X. \quad (7.9)$$

Define now a 2-sub-box  $B = (B_j)_{j=1, \dots, n}$  as follows:

$$B_j = \begin{cases} \{e_j, e'_j\} & \text{if } e_j \neq e'_j, \\ \{e_j, a_j\} & \text{otherwise,} \end{cases} \quad (7.10)$$

where  $a_j$  is an arbitrary element  $\neq e_j$  of  $X_j$  (the latter choice is only made to ensure that  $|B_j| = 2$  in all cases).

Because of (7.9) and (7.10), we have that  $F(e, e')$  is a total evaluation infeasible within  $B$ . Towards constructing a  $B$ -MIPE, delete one after the other (and as far as it can go) coordinates of  $F(e, e')$ , while taking care not to destroy infeasibility within  $B$ . Let  $K \subseteq \{1, \dots, n\}$  be the subset of coordinate indices that remain at the end of this process. Then the partial evaluation

$$x := (f_j(e_j, e'_j))_{j \in K} \quad (7.11)$$

is infeasible within  $B$ . Therefore, lest  $e$  or  $e'$  extends  $\mathbf{x} = (f_j(e_j, e'_j))_{j \in K}$  (not permissible because the latter partial evaluation is infeasible), there exist  $s, t \in K$  such that

$$e_s \neq e'_s \text{ and } e_t \neq e'_t \quad (7.12)$$

and also

$$f_s(e_s, e'_s) = e_s \text{ and } f_t(e_t, e'_t) = e'_t. \quad (7.13)$$

But then if we set

$$u = e_s, u' = e'_s, v = e_t, v' = e'_t, \quad (7.14)$$

we have, by (7.6), (7.12), (7.13) and (7.14), that

$$\text{vertex}_s(e, e') = uu'_s \in V_1 \text{ and } \text{vertex}_t(e, e') = vv'_t \in V_2 \quad (7.15)$$

and, by (7.6), (7.13) and (7.14), we get that

$$uu'_s \xrightarrow{B, \mathbf{x}, K} vv'_t$$

which by (7.15) is a contradiction, because we get an edge from  $V_1$  to  $V_2$ . This completes the proof of Theorem 7.1.3.  $\square$

Before proceeding further, we point out that the three types of non-dictatorial aggregators in Theorem 7.1.1 are, in a precise sense, independent of each other.

**Example 7.1.3.** Consider the set  $X = \{0, 1\}^3 \setminus \{(1, 1, 0)\}$  of satisfying assignments of the Horn clause  $(\neg x \vee \neg y \vee z)$ .

It is easy to see that  $X$  is closed under the binary operation  $\wedge$ , but it is not closed under the ternary majority operation  $\text{maj}$  or the ternary minority operation  $\oplus$ .

Thus,  $X$  is a possibility domain admitting a non-dictatorial binary aggregator, but not a majority aggregator or a minority aggregator.  $\diamond$

**Example 7.1.4.** Consider the set  $X = \{(0, 0, 1), (0, 1, 0), (1, 0, 0), (1, 1, 1)\}$  of solutions of the equation  $x + y + z = 1$  over the two-element field.

It is easy to see that  $X$  is closed under the ternary minority operation  $\oplus$ , but it is not closed under the ternary majority operation  $\text{maj}$ . Moreover, Dokow and Holzman [76, Example 3] pointed out that  $X$  is totally blocked, hence Theorem 7.1.3 implies that  $X$  does not admit a non-dictatorial binary aggregator.

Thus,  $X$  is a possibility domain admitting a minority aggregator, but not a majority aggregator or a non-dictatorial binary aggregator.  $\diamond$

**Example 7.1.5.** Consider the set  $X = \{(0, 1, 2), (1, 2, 0), (2, 0, 1), (0, 0, 0)\}$ .

This set was studied in [77, Example 4]. It can be shown that  $X$  admits a majority aggregator. To see this, consider the ternary operator  $F = (f_1, f_2, f_3)$  such that  $f_j(x, y, z)$  is the majority of  $x, y, z$ , if at least two of the three values are equal, or it is 0 otherwise. Notice that in the latter case the value 0 must be one of the  $x, y, z$ , so this operator is indeed supportive. It is easy to verify that  $X$  is closed under  $(f_1, f_2, f_3)$ . Moreover, if one of the  $f_j$ 's is restricted to a two-element domain (i.e., to one of  $\{0, 1\}$ ,  $\{(1, 2)\}$ ,  $\{0, 2\}$ ), then it must be the majority function by its definition, so  $F$  is indeed a majority aggregator on  $X$ .

*Dokow and Holzman argued that  $X$  is totally blocked, hence Theorem [7.1.3](#) implies that  $X$  does not admit a non-dictatorial binary aggregator.*

*Next, we claim that  $X$  does not admit a minority aggregator. Towards a contradiction, assume it admits the minority aggregator  $G = (g_1, g_2, g_3)$ . By applying  $G$  to the triples  $(0, 1, 2)$ ,  $(1, 2, 0)$ ,  $(0, 0, 0)$  in  $X$ , we infer that the triple  $(g_1(0, 1, 0), g_2(1, 2, 0), g_3(2, 0, 0))$  must be in  $X$ . By the assumption that this aggregator is the minority operator on two-element domains, we have that  $g_1(0, 1, 0) = 1$  and  $g_3(2, 0, 0) = 2$ , so  $X$  contains a triple of the form  $(1, g_2(1, 2, 0), 2)$ ; however,  $X$  contains no triple whose first coordinate is 1 and its third coordinate is 2, so we have arrived at a contradiction.*

*Thus,  $X$  is a possibility domain admitting a majority aggregator, but not a minority aggregator or a non-dictatorial binary aggregator.  $\diamond$*

Observe that the possibility domains in Examples [7.1.3](#) and [7.1.4](#) are in the Boolean framework, while the possibility domain in Example [7.1.5](#) is not. This is no accident, because it turns out that, in the Boolean framework, if a set admits a majority aggregator, then it also admits a non-dictatorial binary aggregator. This property is shown as a claim in the proof of Theorem [7.1.2](#). Note also that this explains why admitting a majority aggregator is not part of the characterization of possibility domains in the Boolean framework in Theorem [7.1.2](#).

### 7.1.3 Uniform Possibility Domains

In this Section, we connect aggregation theory with multi-sorted constraint satisfaction problems. Towards this goal, we introduce a new type of aggregators, namely *uniform* non-dictatorial aggregators, which is a stronger notion of non-dictatorial aggregators.

**Definition 7.1.3.** *We say that an aggregator  $F = (f_1, \dots, f_n)$  for  $X$  is uniform non-dictatorial if for every  $j = 1, \dots, n$  and every two-element subset  $B_j \subseteq X_j$ , we have that  $f_j \upharpoonright B_j$  is not a projection function.*

Obviously, such an aggregator is not dictatorial. In the literature, several other ways to strengthen the notion of being non-dictatorial have been suggested, mainly for the Boolean case. We mention these below and outline their relation to the notion of uniform non-dictatorial aggregator.

A  $k$ -ary aggregator  $F = (f_1, \dots, f_m)$ , in the Boolean framework, is *locally dictatorial* if there exists a  $j \in \{1, \dots, n\}$  and a  $d \in \{1, \dots, k\}$  such that

$f_j = pr_d^k$  (see [175]). It can be easily seen that, in the Boolean framework,  $F$  is locally non-dictatorial if and only if it is uniform non-dictatorial. As mentioned in Section 2, an aggregator  $F = (f_1, \dots, f_n)$  is *anonymous* if the output of each  $f_j$  depends only on the multi-set of its input values. It is easy to see that an anonymous aggregator is also uniform non-dictatorial; the converse however is not necessarily true. The same holds for the *StrongDem* aggregator, defined by Szegedy and Xu in [207]: for all  $j \in \{1, \dots, n\}$ , for all  $B_j \subseteq X_j$  and for all  $i \in \{1, \dots, k\}$ , there exist  $a_j^1, \dots, a_j^k \in B_j$ , such that, the value of  $f_j(a_j^1, \dots, a_j^{i-1}, x_j^i, a_j^{i+1}, \dots, a_j^k)$  does not depend on the value of  $x_j^i$  (this aggregator is defined in the non-Boolean framework). It is again easy to see that such an aggregator is uniform non-dictatorial. On the other hand, consider the uniform non-dictatorial (and systematic) aggregator  $F = (f, \dots, f)$  such that  $f$  is a minority operator.  $F$  is not StrongDem: consider  $f$  on input  $a, b, c \in \{0, 1\}$ . If  $a = b$ , then if  $c \neq a$ ,  $f(a, b, c) = c$ ; else it is equal to  $a$ . If  $a \neq b$ , then, if  $c = a$ ,  $f(a, b, c) = b$ , else it is equal to  $a$ . Thus, there are no values for  $a, b$  in order for  $f$  to be independent of  $c$ .

Finally, it is interesting to compare aggregators that are not uniform non-dictatorial with *generalized dictatorships*, defined by Grandi and Endriss [111, 113] (or *rolling dictatorships*; see [49]). A  $k$ -ary aggregator  $F = (f_1, \dots, f_n)$  for  $X$  is a generalized dictatorship if there exists a function  $g : X^k \mapsto \{1, \dots, k\}$  such that, for any  $\mathbf{x} = (x^1, \dots, x^k) \in X^k$ ,

$$\bar{f}(\mathbf{x}) = x^{g(\mathbf{x})}.$$

The notions of generalized dictatorships and not uniform non-dictatorial aggregators differ. Informally, note that: (i) a generalized dictatorship refers to elements  $\mathbf{x} \in X^k$ , whereas not uniform non-dictatorial aggregators to issues  $j$  and (ii) for an aggregator  $F$  *not* to be a generalized dictatorship, we only need one element  $\mathbf{x}$  such that  $F$  is not dictatorial on this element, whereas for an aggregator to be uniform non-dictatorial, we need all issues to be aggregated in a non-dictatorial way. Formally, the class of aggregators that are not generalized dictatorships is neither a subclass nor a superclass of the uniform non-dictatorial ones. Indeed, recall the aggregator of Example 5.2.1. It is not a generalized dictatorship, since for each  $\mathbf{x} \in X^k$ , it introduces two dictators (the  $d$ -th voter for the first  $l$  issues and the  $d'$ -th for the rest) and, obviously, it is not a uniform non-dictatorial aggregator, since all of its components are projections. On the other hand, let  $X = \{(0, 0), (1, 1)\} \subseteq \{0, 1\}^2$  and let  $F = (\text{maj}, \text{maj})$ , which is obviously a uniform non-dictatorial aggregator. It is not hard to see that  $f$  is a generalized dictatorship.

We now turn our attention to sets of feasible voting patterns that admit uniform non-dictatorial aggregators.

**Definition 7.1.4.** *Let  $X$  be a set of feasible voting patterns. We say that  $X$  is a uniform possibility domain if  $X$  admits a uniform non-dictatorial aggregator of some arity.*

The next example shows that the notion of a uniform possibility domain is strictly stronger than the notion of a possibility domain.

**Example 7.1.6.** *Let  $W = \{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}$  be the 1-in-3 relation, considered in Example 5.2.4. As seen earlier, the Cartesian product  $W \times W$  is a possibility domain. We claim that  $W \times W$  is not a uniform possibility domain in the sense of Definition 7.1.4. Indeed, since  $W$  is an impossibility domain, it follows easily that for every  $k$ , all  $k$ -ary aggregators of  $W \times W$  are of the form*

$$(pr_d^k, pr_d^k, pr_d^k, pr_{d'}^k, pr_{d'}^k, pr_{d'}^k), \text{ for } d, d' \in \{1, \dots, k\}. \quad \diamond$$

It is obvious that every set  $X$  that admits a majority aggregator or a minority aggregator is a uniform possibility domain. The next example states that uniform possibility domains are closed under Cartesian products.

**Example 7.1.7.** *If  $X$  and  $Y$  are uniform possibility domains, then so is their Cartesian product  $X \times Y$ .*

*Assume that  $X \subseteq \mathcal{D}^l$  and  $Z \subseteq \mathcal{D}^{n-l}$ , where  $1 \leq l < n$ . Let  $(f_1, \dots, f_l)$  be a uniform non-dictatorial aggregator for  $X$  and let  $(f_{l+1}, \dots, f_n)$  be a uniform non-dictatorial aggregator for  $Y$ . Then*

$$(f_1, \dots, f_l, f_{l+1}, \dots, f_n)$$

*is a uniform non-dictatorial aggregator for  $X \times Y$ .*  $\diamond$

We now provide a useful characterization for uniform possibility domains.

**Theorem 7.1.4.** *Let  $X$  be a set of feasible voting patterns. The following statements are equivalent.*

1.  $X$  is a uniform possibility domain.
2. For every  $j = 1, \dots, n$  and for every two-element subset  $B_j \subseteq X_j$ , there is an aggregator  $F = (f_1, \dots, f_n)$  (that depends on  $j$  and  $B_j$ ) of some arity such that  $f_j \upharpoonright B_j$  is not a projection function.

3. There is a ternary aggregator  $F = (f_1, \dots, f_n)$  such that for all  $j = 1, \dots, n$  and all two-element subsets  $B_j \subseteq X_j$ , we have that  $f_j \upharpoonright B_j$  is one of the ternary operations  $\wedge^{(3)}$ ,  $\vee^{(3)}$ ,  $\text{maj}$ ,  $\oplus$  (to which of these four ternary operations the restriction  $f_j \upharpoonright B_j$  is equal to depends on  $j$  and  $B_j$ ).
4. There is a ternary aggregator  $F = (f_1, \dots, f_n)$  such that for all  $j = 1, \dots, n$  and all  $x, y \in X_j$ , we have that  $f_j(x, y, y) = f_j(y, x, y) = f_j(y, y, x)$ .

Before the proof of Theorem [7.1.4](#), we give several preliminaries.

We start with the following lemma:

**Lemma 7.1.4** (Superposition of aggregators). *Let  $F = (f_1, \dots, f_n)$  be a  $k$ -ary aggregator and let*

$$H^1 = (h_1^1, \dots, h_n^1), \dots, H^k = (h_1^k, \dots, h_n^k)$$

*be  $k$   $l$ -ary aggregators (all on  $n$  issues). Then the  $n$ -tuple of  $l$ -ary functions  $(g_1, \dots, g_n)$  defined by:*

$$g_j(x_1, \dots, x_l) = f_j(h_j^1(x_1, \dots, x_l), \dots, h_j^k(x_1, \dots, x_l)), j = 1, \dots, n$$

*is also an aggregator.*

*Proof.* Let  $x_j^s$ ,  $s = 1, \dots, l$ ,  $j = 1, \dots, n$  be an  $l \times n$  matrix whose rows are in  $X$ . Since the  $H^i$ ,  $i = 1, \dots, k$  are  $l$ -ary aggregators, we conclude that for all  $i = 1, \dots, k$ ,

$$(h_1^i(x_1^1, \dots, x_l^1), \dots, h_n^i(x_n^1, \dots, x_n^l)) \in X.$$

We now apply the aggregator  $F = (f_1, \dots, f_n)$  to the  $k \times n$  matrix

$$h_j^i(x_j^1, \dots, x_j^l), i = 1, \dots, k, j = 1, \dots, n,$$

which concludes the proof. □

Using the above lemma we will assume below, often tacitly, that various tuples of functions obtained by superposition of aggregators with other aggregators, like projections, are aggregators as well.

We now prove three lemmas:

**Lemma 7.1.5.** *Let  $\mathcal{D}$  be an arbitrary set and  $f : \mathcal{D}^3 \mapsto \mathcal{D}$  a ternary supportive operation on  $\mathcal{D}$ , and  $B$  a two-element subset of  $\mathcal{D}$  taken as  $\{0, 1\}$ . Then  $f|B$  is commutative if and only if  $f|B \in \{\wedge^{(3)}, \vee^{(3)}, \text{maj}, \oplus\}$ .*

*Proof.* Only the sufficiency of commutativity of  $f|B$  for its being one of  $\wedge^{(3)}, \vee^{(3)}, \text{maj}, \oplus$  is not entirely trivial. Since  $f$  is supportive,  $f(0, 0, 0) = 0$  and  $f(1, 1, 1) = 1$ . Assume  $f|B$  is commutative. Let

$$f(1, 0, 0) = f(0, 1, 0) = f(0, 0, 1) := a, \text{ and}$$

$$f(0, 1, 1) = f(1, 0, 1) = f(1, 1, 0) := b.$$

By supportiveness,  $a, b \in \{0, 1\}$ . If  $a = b = 0$ , then  $f = \wedge^{(3)}$ ; if  $a = b = 1$ ,  $f = \vee^{(3)}$ ; if  $a = 0$  and  $b = 1$ ,  $f = \text{maj}$ ; and if  $a = 1$  and  $b = 0$ ,  $f = \oplus$ .  $\square$

**Lemma 7.1.6.** *Let  $\mathcal{D}$  be an arbitrary set and  $f, g : \mathcal{D}^3 \mapsto \mathcal{D}$  two ternary supportive operations on  $\mathcal{D}$ . Define the supportive as well ternary operation*

$$h(x, y, z) = f(g(x, y, z), g(y, z, x), g(z, x, y)).$$

*If  $B$  is a two-element subset of  $\mathcal{D}$  then  $h|B$  is commutative if either  $f|B$  or  $g|B$  is commutative.*

*Proof.* The result is entirely trivial if  $g|B$  is commutative, since in this case, by supportiveness of  $f$ ,  $h|B = g|B$ . If on the other hand  $f|B$  is commutative then easily from the definition of  $h$  follows that for any  $x, y, z \in B$ ,  $h(x, y, z) = h(y, z, x) = h(z, x, y)$ . This form of superposition of  $f$  and  $g$  appears also in Bulatov [38], Section 4.3.  $\square$

For notational convenience, we introduce the following definition:

**Definition 7.1.5.** *Let  $F$  and  $G$  be two aggregators on  $X$ . Let  $F \diamond G$  be the ternary aggregator  $H = (h_1, \dots, h_m)$  defined by:*

$$h_j(x, y, z) = f_j(g_j(x, y, z), g_j(y, z, x), g_j(z, x, y)), j = 1, \dots, m,$$

*(The fact that  $H$  is indeed an aggregator follows from Lemma 7.1.4 and the fact that a tuple of functions comprised of the same projections is an aggregator.)*

**Lemma 7.1.7.** *Let  $H$  and  $G$  be two aggregators on  $X$ . Let  $i, j \in \{1, \dots, n\}$  two arbitrary issues (perhaps identical) and  $B_i, B_j$  two-element subsets of  $X_i$  and  $X_j$ , respectively. If  $f_i \upharpoonright B_i$  and  $g_j \upharpoonright B_j$  are commutative (i.e., by Lemma 7.1.5 if each is one of the  $\wedge^{(3)}, \vee^{(3)}, \text{maj}, \oplus$ ) then both  $F \diamond G \upharpoonright B_i$  and  $F \diamond G \upharpoonright B_j$  are commutative (i.e., each is one of the  $\wedge^{(3)}, \vee^{(3)}, \text{maj}, \oplus$ ).*

*Proof.* Immediate by Lemmas 7.1.5 and 7.1.6.  $\square$

We now prove the characterization of uniform possibility domains. Some of the techniques employed in the proof of Theorem 7.1.4 and the preceding lemmas had been used by Bulatov (see 37, Proposition 3.1] 38, Proposition 2.2]; these results however consider only operations of arity two or three.) $\square$

*Proof of Theorem 7.1.4.*

The directions (1)  $\implies$  (2) and (3)  $\implies$  (1) are obvious. Also the equivalence of (3) and (4) immediately follows from Lemma 7.1.5. It remains to show (2)  $\implies$  (3). For a two-element subset  $B_j \subseteq X_j$ , let  $\mathcal{C}_{B_j}$  be the clone (Lemma 7.1.3) of the restrictions  $f_j \upharpoonright B_j$  of the  $j$ -th components of aggregators  $F = (f_1, \dots, f_n)$ . By Post 186], we can easily get that  $\mathcal{C}_{B_j}$  contains one of the operations  $\wedge, \vee, \text{maj}$  and  $\oplus$ . Therefore, easily, for all  $j, B_j$  there is a ternary aggregator  $F = (f_1, \dots, f_n)$  (depending on  $j, B_j$ ) such that  $f_j \upharpoonright B_j$  is one of the  $\wedge^{(3)}, \vee^{(3)}, \text{maj}$  and  $\oplus$ . Now let  $F^1, \dots, F^N$  be an arbitrary enumeration of all ternary aggregators each of which on some issue  $j$  and some two-element  $B_j$  is one of the  $\wedge^{(3)}, \vee^{(3)}, \text{maj}$  and  $\oplus$  and such that the  $F^l$ 's cover all possibilities for  $j, B_j$ . As a ternary operation  $H$  such that uniformly for each  $j, B_j$ , the restriction  $h_j \upharpoonright B_j$  belongs to the set  $\{\wedge^{(3)}, \vee^{(3)}, \text{maj}, \oplus\}$  we can take, by Lemma 7.1.7,

$$(\dots (F^1 \diamond F^2) \diamond \dots \diamond F^N),$$

which concludes the proof.  $\square$

We now prove a result that connects locally non-dictatorial and anonymous aggregators in the Boolean framework. Nehring and Puppe 175, Theorem 2] proved that a domain admits a *monotone* and locally non-dictatorial aggregator if and only if it admits a monotone anonymous one. An  $n$ -ary aggregator  $F = (f_1, \dots, f_m)$  is monotone if, for all  $j \in \{1, \dots, n\}$  and for all  $i \in \{1, \dots, k\}$  it satisfies the following condition:

$$f_j(x_j^1, \dots, x_j^{i-1}, x_j^i, x_j^{i+1}, \dots, x_j^k) = 1 \implies f_j(x_j^1, \dots, x_j^{i-1}, 1, x_j^{i+1}, \dots, x_j^k) = 1.$$

<sup>1</sup>This came to the attention of the authors only after the work reported here had been essentially completed.



As a corollary of our Theorem [7.1.4](#), we show not only that the monotonicity requirement can be dropped but also that in the “only if” direction, the anonymous aggregator can be proved to be ternary. Specifically, we prove:

**Corollary 7.1.1.** *Let  $X$  be a set of feasible voting patterns in the Boolean framework. The following statements are equivalent.*

1.  $X$  admits a locally non-dictatorial aggregator of some arity  $k$ .
2.  $X$  admits an anonymous ternary aggregator.

*Proof.* Recall that, in the Boolean framework, the notions of locally and uniform non-dictatorial aggregators coincide. Thus, condition [1](#) of the Corollary is equivalent with  $X$  being a uniform possibility domain. The result now follows by condition [4](#) of Theorem [7.1.4](#)  $\square$

Recall the setting of Subsec. [2.1.1](#) and [4.2.2](#). If  $X \subseteq \mathcal{D}^n$  is a set of feasible voting patterns, then  $X$  can be considered as multi-sorted relation with signature  $(1, \dots, n)$  (one sort for each issue). We write  $\mathcal{R}_X^{\text{cons}}$  to denote the multi-sorted conservative constraint language consisting of  $X$  and all subsets of  $\mathcal{D}$ .

In this framework, and in case all  $\mathcal{D} = \{0, 1\}$ , we have that  $\text{CSP}(\mathcal{R}_X^{\text{cons}})$  coincides with the problem  $\text{SAT}_C(\{X\})$ . Note that the presence of the sets  $\{0\}$  and  $\{1\}$  in the constraint language amounts to allowing constants, besides variables, in the constraints.

Bulatov’s Dichotomy Theorem for  $c$ -MCSP(Th. [4.2.7](#)), in our setting reads:

**Theorem 7.1.5.** *If for any  $j = 1, \dots, n$  and any two-element subset  $B_j \subseteq X_j$  there is either a binary aggregator  $F = (f_1, \dots, f_n)$  such that  $f_j \upharpoonright B_j \in \{\wedge, \vee\}$  or a ternary aggregator  $F = (f_1, \dots, f_m)$  such that  $f_j \upharpoonright B_j \in \{\text{maj}, \oplus\}$ , then  $c$ -MCSP( $\mathcal{R}_X$ ) is solvable in polynomial time; otherwise it is NP-complete.*

We now state the following dichotomy theorem.

**Theorem 7.1.6.** *If  $X$  is a uniform possibility domain, then  $c$ -MCSP( $\mathcal{R}_X$ ) is solvable in polynomial time; otherwise it is NP-complete.*

*Proof.* The tractability part of the statement follows from Bulatov’s Dichotomy Theorem and item [3](#) of Theorem [7.1.4](#) (observing that  $x \wedge y = \wedge^{(3)}(x, x, y)$  and similarly for  $\vee$  and using Lemma [7.1.4](#)), whereas the completeness part follows from Bulatov’s Dichotomy Theorem and item [2](#) of Theorem [7.1.4](#)  $\square$

We end this section with the following example:

**Example 7.1.8.** Let  $Y = \{0, 1\}^3 \setminus \{(1, 1, 0)\}$  be the set of satisfying assignments of the clause  $(\neg x \vee \neg y \vee z)$  and let:

$$Z = \{(1, 1, 0), (0, 1, 1), (1, 0, 1), (0, 0, 0)\}$$

be the set of solutions of the equation  $x + y + z = 0$  over the two-element field.

We claim that  $Y$  and  $Z$  are uniform possibility domains, hence, by Example 7.1.7, the Cartesian product  $X = Y \times Z$  is also a uniform possibility domain. From Theorem 7.1.6, it follows that  $c\text{-MCSP}(\mathcal{R}_X)$  is solvable in polynomial time. However, the generalized satisfiability problem  $c\text{-CSP}(\mathcal{R}_X)$  is NP-complete.

Indeed, in Schaefer's [192] terminology, the set  $Y$  is Horn (equivalently, it is coordinate-wise closed under  $\wedge$ ); however, it is not dual Horn (equivalently, it is not coordinate-wise closed under  $\vee$ ), nor affine (equivalently, it does not admit a minority aggregator) nor bijunctive (equivalently, it does not admit a majority aggregator). Therefore, by coordinate-wise closure under  $\wedge$ , we have that  $Y$  is a uniform possibility domain. Also,  $Z$  is affine, but not Horn, nor dual Horn neither bijunctive. So, being affine,  $Z$  is a uniform possibility domain. The NP-completeness of  $c\text{-CSP}(\mathcal{R}_X)$  follows from Schaefer's dichotomy theorem [192], because  $X$  is not Horn, dual Horn, affine, nor bijunctive.  $\diamond$

## 7.2 Syntactic Characterizations via Integrity Constraints

In this section, we begin (Subsec. 7.2.1) by showing that we can efficiently recognize the syntactic types of the integrity constraints we defined in Sec. 5.3. We then proceed to give syntactic characterizations for Boolean (local) possibility domains via the integrity constraints that describe them (Subsec. 7.2.2). Afterwards, we show how, given a Boolean domain  $\mathcal{D}$ , we can efficiently construct such integrity constraints that describe it (Subsec. 7.2.3). Finally, we turn our attention to other forms of non-dictatorial aggregators and provide corresponding characterizations (Subsec. 7.2.4).

### 7.2.1 Identifying (local) possibility integrity constraints

In this section, we show that identifying (local) possibility integrity constraints can be done in time linear in the length of the input formula. By Definitions 5.3.5 and 5.3.6, it suffices to show that for separable formulas, renamable partially Horn formulas and lpic's, since the corresponding problem for affine formulas is trivial.

In all that follows, we assume that we have a set of variables  $V := \{x_1, \dots, x_n\}$  and a formula  $\phi$  defined on  $V$  that is a conjunction of  $m$  clauses  $C_1, \dots, C_m$ , where  $C_j = (l_{j_1}, \dots, l_{j_{k_j}})$ ,  $j = 1, \dots, m$ , and  $l_{j_s}$  is a positive or negative literal of  $x_{j_s}$ ,  $s = 1, \dots, k_j$ . We denote the set of variables corresponding to the literals of a clause  $C_j$  by  $\text{vbl}(C_j)$ .

We begin with the result for separable formulas:

**Proposition 7.2.1.** *There is an algorithm that, on input a formula  $\phi$ , halts in time linear in the length of  $\phi$  and either returns that the formula is not separable, or alternatively produces a partition of  $V$  in two non-empty and disjoint subsets  $V_1, V_2 \subseteq V$ , such that no clause of  $\phi$  contains variables from both  $V_1$  and  $V_2$ .*

*Proof.* We construct a graph on the variables of  $\phi$ , where two such vertices are connected if they appear consecutively in a common clause of  $\phi$ . The result is then obtained by showing that  $\phi$  is separable if and only if  $G$  is not connected.

Suppose the variables of each clause are ordered by the indices of their corresponding literals in the clause. Thus, we say that  $x_{j_s}, x_{j_t}$  are consecutive in  $C_j$ , if  $t = s + 1$ ,  $s = 1, \dots, k_j - 1$ .

Given a formula  $\phi$ , construct an undirected graph  $G = (V, E)$ , where :

- $V$  is the set of variables of  $\phi$ , and
- two vertices are connected if they appear consecutively in a common clause of  $\phi$ .

It is easy to see that each clause  $C_j$ , where  $\text{vbl}(C_j) = \{x_{j_1}, \dots, x_{j_{k_j}}\}$  induces the path  $\{x_{j_1}, \dots, x_{j_{k_j}}\}$  in  $G$ .

For the proof of linearity, notice that the set of edges can be constructed in linear time with respect to the length of  $\phi$ , since we simply need to read once each clause of  $\phi$  and connect its consecutive vertices. Also, there are

standard techniques to check connectivity in linear time in the number of edges (e.g. by a *depth-first search* algorithm).

The correctness of the algorithm is derived by noticing that two connected vertices of  $G$  cannot be separated in  $\phi$ . Indeed, consider a path  $P := \{x_r, \dots, x_s\}$  in  $G$  (this need not be a path induced by a clause). Then, each couple  $x_t, x_{t+1}$  of vertices in  $P$  belongs in a common clause of  $\phi$ ,  $t = r, \dots, s-1$ . Thus,  $\phi$  is separable if and only if  $G$  is not connected.  $\square$

To deal with renamable partially Horn formulas, we will start with Lewis' idea [160] of creating, for a formula  $\phi$ , a 2SAT formula  $\phi'$  whose satisfiability is equivalent to  $\phi$  being renamable Horn. However, here we need to (i) look for a renaming that might transform only some clauses into Horn and (ii) deal with inadmissible Horn clauses, since such clauses can cause other Horn clauses to become inadmissible too.

**Proposition 7.2.2.** *For every formula  $\phi$ , there is a formula  $\phi'$  such that  $\phi$  is renamable partially Horn if and only if  $\phi'$  is satisfiable.*

Before delving into the proof, we introduce some notation. Assume that after a renaming of some of the variables in  $V$ , we get the partially Horn formula  $\phi^*$ , with  $V_0$  being the admissible set of variables. Let  $\mathcal{C}_0$  be an admissible set of clauses for  $\phi^*$ . We assume below that only a subset  $V^* \subseteq V_0$  has been renamed and that all Horn clauses of  $\phi^*$  with variables exclusively from  $V_0$  belong to  $\mathcal{C}_0$  (see Remark [5.3.2]). Also, let  $V_1 := V \setminus V_0$ . The clauses of  $\phi^*$ , which are in a one to one correspondence with those of  $\phi$ , are denoted by  $C_1^*, \dots, C_m^*$ , where  $C_j^*$  corresponds to  $C_j$ ,  $j = 1, \dots, m$ .

*Proof.* For each variable  $x \in V$ , we introduce a new variable  $x'$ . Intuitively, setting  $x = 1$  means that  $x$  is renamed (and therefore  $x \in V^*$ ), whereas setting  $x' = 1$  means that  $x$  is in  $V_0$ , but is not renamed. Finally we set both  $x$  and  $x'$  equal to 0 in case  $x$  is not in  $V_0$ . Obviously, we should not allow the assignment  $x = x' = 1$  (a variable in  $V_0$  cannot be renamed and not renamed). Let  $\bar{V} = V \cup \{x' \mid x \in V\}$ .

Consider the formula  $\phi'$  below, with variable set  $\bar{V}$ . For each clause  $C$  of  $\phi$  and for each  $x \in \text{vbl}(C)$ : if  $x$  appears positively in  $C$ , introduce the literals  $x$  and  $\neg x'$  and if it appears negatively, the literals  $\neg x$  and  $x'$ .  $\phi'$  is the conjunction of the following clauses: for each clause  $C$  of  $\phi$  and for each two variables  $x, y \in \text{vbl}(C)$ ,  $\phi'$  contains the disjunctions of the positive with the negative literals introduced above. Thus:

- (i) if  $C$  contains the literals  $x, y$ , then  $\phi'$  contains the clauses  $(x \vee \neg y')$  and  $(\neg x' \vee y)$ ,
- (ii) if  $C$  contains the literals  $x, \neg y$ , then  $\phi'$  contains the clauses  $(x \vee \neg y)$  and  $(\neg x' \vee y')$  (accordingly if  $C$  contains  $\neg x, y$ ) and
- (iii) if  $C$  contains the literals  $\neg x, \neg y$ , then  $\phi'$  contains the clauses  $(\neg x \vee y')$  and  $(x' \vee \neg y)$ .

Finally, we add the following clauses to  $\phi'$ :

- (iv)  $(\neg x_i \vee \neg x'_i)$ ,  $i = 1, \dots, n$  and
- (v)  $\bigvee_{x \in \bar{V}} x$ .

The clauses of items (i)–(iv) correspond to the intuition we explained in the beginning. For example, consider the case where a clause  $C_j$  of  $\phi$  has the literals  $x, \neg y$ . If we add  $x$  to  $V_0$  without renaming it, we should not rename  $y$ , since we would have two positive literals in a clause of  $\mathcal{C}_0$ . Also, we should not add the latter to  $V_1$ , since we would have a variable of  $V_0$  appearing positively in a clause containing a variable of  $V_1$ . Thus, we have that  $x' \rightarrow y'$ , which is expressed by the equivalent clause  $(\neg x' \vee y')$  of item (ii). The clauses of item (iv) exclude the assignment  $x = x' = 1$  for any  $x \in V$ . Finally, since we want  $V_0$  to be non-empty, we need at least one variable of  $\bar{V}$  to be set to 1.

To complete the proof of Proposition [7.2.2](#), we now proceed as follows.

( $\Rightarrow$ ) First, suppose  $\phi$  is renamable partially Horn. Let  $V_0, V_1, V^*$  and  $\bar{V}$  as above. Suppose also that  $V_0 \neq \emptyset$ .

Set  $a = (a_1, \dots, a_{2n})$  to be the following assignment of values to the variables of  $\bar{V}$ :

$$a(x) = \begin{cases} 1, & \text{if } x \in V^*, \\ 0, & \text{else,} \end{cases} \quad \text{and} \quad a(x') = \begin{cases} 0, & \text{if } x \in V^* \cup V_1, \\ 1, & \text{else,} \end{cases}$$

for all  $x \in V$ . To obtain a contradiction, suppose  $a$  does not satisfy  $\phi'$ .

Obviously, the clauses of items (iv) and (v) above are satisfied, by the definition of  $a$  and the fact that  $V_0$  is not empty.

Now, consider the remaining clauses of items (i)–(iii) above and suppose for example that some  $(\neg x \vee y')$  is not satisfied. By the definition of  $\phi'$ , there exists a clause  $C$  which, before the renaming takes place, contains the literals  $\neg x, \neg y$  (see item (iii)). Since the clause is not satisfied,  $a(x) = 1$  and

$a(y') = 0$ , which in turn means that  $x \in V^*$  and  $y \in V^* \cup V_1$ . If  $y \in V_1$ ,  $C^*$  contains, after the renaming, a variable in  $V_1$  and a positive appearance of a variable in  $V_0$ . If  $y \in V^*$ ,  $C^*$  contains two positive literals of variables in  $V_0$ . Contradiction. The remaining cases can be proven analogously and are left to the reader.

( $\Leftarrow$ ) Suppose now that  $a = (a_1, \dots, a_{2n})$  is an assignment of values to the variables of  $\bar{V}$  that satisfies  $\phi'$ . We define the following subsets of  $\bar{V}$ :

- $V^* = \{x \mid a(x) = 1\}$ ,
- $V_0 = \{x \mid a(x) = 1 \text{ or } a(x') = 1\}$  and
- $V_1 = \{x \mid a(x) = a(x') = 0\}$ .

Let  $\phi^*$  be the formula obtained by  $\phi$ , after renaming the variables of  $V^*$ .

Obviously,  $V_0$  is not empty, since  $a$  satisfies the clause of item (v).

Suppose that a clause  $C^*$ , containing only variables from  $V_0$ , is not Horn. Then,  $C^*$  contains two positive literals  $x, y$ . If  $x, y \in V_0 \setminus V^*$ , then neither variable was renamed and thus  $C$  also contains the literals  $x, y$ . This means that, by item (i) above,  $\phi'$  contains the clauses  $(x \vee \neg y')$  and  $(\neg x' \vee y)$ . Now, since  $x, y \in V_0 \setminus V^*$ , it holds that  $a(x) = a(y) = 0$  and  $a(x') = a(y') = 1$ . Then,  $a$  does not satisfy these two clauses. Contradiction. In the same way, we obtain contradictions in cases that at least one of  $x$  and  $y$  is in  $V^*$ .

Finally, suppose that there is a variable  $x \in V_0$  that appears positively in a clause  $C^* \notin \mathcal{C}_0$ . Let  $y \in V_1$  be a variable in  $C^*$  (there is at least one such variable, lest  $C^* \in \mathcal{C}_0$ ). Suppose also that  $y$  appears positively in  $C^*$ .

Assume  $x \in V^*$ . Then,  $C$  contains the literals  $\neg x, y$ . Thus, by item (ii),  $\phi'$  contains the clause  $(\neg x \vee y)$ . Furthermore, since  $x \in V^*$ ,  $a(x) = 1$  and since  $y \in V_1$ ,  $a(y) = 0$ . Thus the above clause is not satisfied. Contradiction. In the same way, we obtain contradictions in all the remaining cases.  $\square$

To compute  $\phi'$  from  $\phi$ , one would need quadratic time in the length of  $\phi$ . Thus, we introduce the following linear algorithm that decides if a formula  $\phi$  is renamable partially Horn, by tying a property of a graph constructed based on  $\phi$ , with the satisfiability of  $\phi'$ .

**Theorem 7.2.1.** *There is an algorithm that, on input a formula  $\phi$ , halts in time linear in the length of  $\phi$  and either returns that  $\phi$  is not renamable partially Horn or alternatively produces a subset  $V^* \subseteq V$  such that the formula  $\phi^*$  obtained from  $\phi$  by renaming the literals of variables in  $V^*$  is partially Horn.*

To prove Theorem [7.2.1](#), we define a *directed bipartite* graph  $G$ , i.e. a directed graph whose set of vertices is partitioned in two sets such that no vertices belonging in the same part are adjacent. Then, by computing its *strongly connected components (scc)*, i.e. its maximal sets of vertices such that every two of them are connected by a directed path, we show that at least one of them is not *bad* (does not contain a pair of vertices we will specify below) *if and only if*  $\phi$  is renamable partially Horn.

For a directed graph  $G$ , we will denote a directed edge from a vertex  $u$  to a vertex  $v$  by  $(u, v)$ . A (directed) path from  $u$  to  $v$ , containing the vertices  $u = u_0, \dots, u_s = v$ , will be denoted by  $(u, u_1, \dots, u_{s-1}, v)$  and its existence by  $u \rightarrow v$ . If both  $u \rightarrow v$  and  $v \rightarrow u$  exist, we will sometimes write  $u \leftrightarrow v$ .

Recall that given a directed graph  $G = (V, E)$ , there are known algorithms that can compute the scc of  $G$  in time  $O(|V| + |E|)$ , where  $|V|$  denotes the number of vertices of  $G$  and  $|E|$  that of its edges (see e.g. Tarjan [\[211\]](#)). By identifying the vertices of each scc, we obtain a *directed acyclic graph (DAG)*. An ordering  $(u_1, \dots, u_n)$  of the vertices of a graph is called *topological* if there are no edges  $(u_i, u_j)$  such that  $i \geq j$ , for all  $i, j \in \{1, \dots, n\}$ .

*Proof.* Given  $\phi$  defined on  $V$ , whose set of clauses is  $\mathcal{C}$  and let again  $\bar{V} = V \cup \{x' \mid x \in V\}$ . We define the graph  $G$ , with vertex set  $\bar{V} \cup \mathcal{C}$  and edge set  $E$  such that, if  $C \in \mathcal{C}$  and  $x \in \text{vbl}(C)$ , then:

- if  $x$  appears *negatively* in  $C$ ,  $E$  contains  $(x, C)$  and  $(C, x')$ ,
- if  $x$  appears *positively* in  $C$ ,  $E$  contains  $(x', C)$  and  $(C, x)$  and
- $E$  contains no other edges.

Intuitively, if  $x, y \in \bar{V}$ , then a path  $(x, C, y)$  corresponds to the clause  $x \rightarrow y$  which is logically equivalent to  $(\neg x \vee y)$ . The intuition behind  $x$  and  $x'$  is exactly the same as in Proposition [7.2.2](#). We will thus show that the bipartite graph  $G$  defined above, contains all the necessary information to decide if  $\phi'$  is satisfiable, with the difference that  $G$  can be constructed in time linear in the length of the input formula. To that end, observe that to construct  $G$ , we need constant-time access to the formula  $\phi$ . Indeed, one only needs to read  $\phi$  once, from left to right, constructing one vertex for each clause and connecting the vertices of the clause with it in the way described above.

There is a slight technicality arising here since, by the construction above,  $G$  always contains either the path  $(x, C, x')$  or  $(x', C, x)$ , for any clause  $C$  and

$x \in \text{vbl}(C)$ , whereas neither  $(\neg x \vee x')$  nor  $(x \vee \neg x')$  are ever clauses of  $\phi'$ . Thus, from now on, we will assume that no path can contain the vertices  $x$ ,  $C$  and  $x'$  or  $x'$ ,  $C$  and  $x$  *consecutively*, for any clause  $C$  and  $x \in \text{vbl}(C)$ .

Observe that by construction, (i)  $(x, C)$  or  $(C, x)$  is an edge of  $G$  if and only if  $x \in \text{vbl}(C)$ ,  $x \in \bar{V}$  and (ii)  $(x, C)$  (resp.  $(x', C)$ ) is an edge of  $G$  if and only if  $(C, x')$  (resp.  $(C, x)$ ) is one too.

We now prove several claims concerning the structure of  $G$ . To make notation less cumbersome, assume that for an  $x \in V$ ,  $x'' = x$ . Consider the formula  $\phi'$  of Proposition [7.2.2](#).

**Claim 7.2.1.** *Let  $x, y \in \bar{V}$ . For  $z_1, \dots, z_k \in \bar{V}$  and  $C_1, \dots, C_{k+1} \in \mathcal{C}$ , it holds that  $(x, C_1, z_1, C_2, \dots, z_k, C_{k+1}, y)$  is a path of  $G$  if and only if  $(\neg x \vee z_1)$ ,  $(\neg z_i \vee z_{i+1})$ ,  $i = 1, \dots, k - 1$  and  $(\neg z_k \vee y)$  are all clauses of  $\phi'$ .*

*Proof of Claim.* Can be easily proved inductively to the length of the path, by recalling that a path  $(u, C, v)$  corresponds to the clause  $(\neg u \vee v)$ , for all  $u, v \in \bar{V}$  and  $C \in \mathcal{C}$ .  $\square$

**Claim 7.2.2.** *Let  $x, y \in \bar{V}$ . If  $x \rightarrow y$ , then  $y' \rightarrow x'$ .*

*Proof of Claim.* Since  $x \rightarrow y$ , there exist variables  $z_1, \dots, z_k \in \bar{V}$  and clauses  $C_1, \dots, C_{k+1} \in \mathcal{C}$ , such that  $(x, C_1, z_1, C_2, \dots, z_k, C_{k+1}, y)$  is a path of  $G$ . By Claim [7.2.1](#),  $(\neg x \vee z_1)$ ,  $(\neg z_i \vee z_{i+1})$ ,  $i = 1, \dots, k - 1$  and  $(\neg z_k \vee y)$  are all clauses of  $\phi'$ . By Proposition [7.2.2](#), so do  $(\neg y' \vee z'_k)$ ,  $(\neg z'_{i+1} \vee z'_i)$ ,  $i = 1, \dots, k - 1$  and  $(\neg z'_1 \vee x')$  and the result is obtained by using Claim [7.2.1](#) again.  $\square$

We can obtain the scc's of  $G$  using a variation of a *depth-first search (DFS)* algorithm, that, whenever it goes from a vertex  $x$  (resp.  $x'$ ) to a vertex  $C$ , it cannot then go to  $x'$  (resp.  $x$ ) at the next step. Since the algorithm runs in time linear in the number of the vertices and the edges of  $G$ , it is also linear in the length of the input formula  $\phi$ .

Let  $S$  be a scc of  $G$ . We say that  $S$  is *bad*, if, for some  $x \in V$ ,  $S$  contains both  $x$  and  $x'$ . We can decide if each of the scc's is bad or not again in time linear in the length of the input formula.

**Claim 7.2.3.** *Let  $S$  be a bad scc of  $G$  and  $y \in \bar{V}$  be a vertex of  $S$ . Then,  $y'$  is in  $S$ .*

*Proof of Claim.* Since  $S$  is bad, there exist two vertices  $x, x'$  of  $\bar{V}$  in  $S$ . If  $x = y$  we have nothing to prove, so we assume that  $x \neq y$ . Then, we have



that  $y \rightarrow x$ , which, by Claim [7.2.2](#) implies that  $x' \rightarrow y'$ . Since  $x \rightarrow x'$ , we get that  $y \rightarrow y'$ . That  $y' \rightarrow y$  can be proven analogously.  $\square$

Let the scc's of  $G$ , in *reverse topological order*, be  $S_1, \dots, S_t$ . We describe a process of assigning values to the variables of  $\bar{V}$ :

1. Set every variable that appears in a bad scc of  $G$  to 0.
2. For each  $j = 1, \dots, t$  assign value 1 to every variable of  $S_j$  that has not already received one (if  $S_j$  is bad no such variable exists). If some  $x \in \bar{V}$  of  $S_j$  takes value 1, then assign value 0 to  $x'$ .
3. Let  $a$  be the resulting assignment to the variables of  $\bar{V}$ .

Now, the last claim we prove is the following:

**Claim 7.2.4.** *There is at least one variable  $z \in \bar{V}$  that does not appear in a bad scc of  $G$  if and only if  $\phi'$  is satisfiable.*

*Proof of Claim.* ( $\Rightarrow$ ) We prove that every clause of type (i)–(v) is satisfied. First, by the construction of  $a$ , every clause  $\neg x_i \vee \neg x'_i$ ,  $i = 1, \dots, n$ , of type (iv) is obviously satisfied. Also, since by the hypothesis,  $z$  is not in a bad scc, it holds, by step 2 above, that either  $z$  or  $z'$  are set to 1. Thus, the clause  $\bigvee_{x \in \bar{V}} x$  of type (v) is also satisfied.

Now, suppose some clause  $(x \vee \neg y')$  (type (i)) of  $\phi'$  is not satisfied. Then  $a(x) = 0$  and  $a(y') = 1$ . Furthermore, there is a vertex  $C$  such that  $(y', C)$  and  $(C, x)$  are edges of  $G$ . By the construction of  $G$ ,  $(x', C)$  and  $(C, y)$  are also edges of  $G$ .

Since  $a(x) = 0$ , it must hold either that  $x$  is in a bad scc of  $G$ , or that  $a(x') = 1$ . In the former case, we have that  $x \rightarrow x'$ , which, together with  $(y', C, x)$  and  $(x', C, y)$  gives us that  $y' \rightarrow y$ . Contradiction, since then  $a(y')$  should be 0. In the latter case, we have that there are two scc's  $S_p, S_r$  of  $G$  such that  $x \in S_p$ ,  $x' \in S_r$  and  $p < r$  in their topological order. But then, there is some  $q : p \leq q \leq r$  such that  $C$  in  $S_q$ . Now, if  $p = q$ , we obtain a contradiction due to the existence of  $(x', C)$ , else, due to  $(C, x)$ .

The proof for the rest of the clauses of types (i)–(iii) are left to the reader.

( $\Leftarrow$ ) First, recall that for two propositional formulas  $\phi, \psi$ , we say that  $\phi$  *logically entails*  $\psi$ , and write  $\phi \models \psi$ , if any assignment that satisfies  $\phi$ , satisfies  $\psi$  too.

Now observe that, if  $x, y$  are two vertices in  $\bar{V}$  such that  $x \rightarrow y$ , then  $\phi' \models (\neg x \vee y)$ . Indeed, suppose  $\beta$  is an assignment of values that satisfies

$\phi'$ . If  $\beta(y) = 1$ , we have nothing to prove. Thus, assume that  $\beta(y) = 0$ . By Claim [7.2.1](#), if  $(x, C_1, z_1, C_2, z_2, \dots, z_k, C_{k+1}, y)$  is the path  $x \rightarrow y$ , then  $(\neg x \vee z_1)$ ,  $(\neg z_i \vee z_{i+1})$ ,  $i = 1, \dots, k-1$  and  $(\neg z_k \vee y)$  are all clauses of  $\phi'$  and are thus satisfied by  $\beta$ . Since  $\beta(y) = 0$ , we have  $\beta(z_k) = 0$ . Continuing in this way,  $\beta(z_i) = 0$ ,  $i = 1, \dots, k$  and thus  $\beta(x) = 0$  too, which implies that  $\beta(\neg x \vee y) = 1$ .

Now, for the proof of the claim, suppose again that  $\phi'$  is satisfiable, and let  $\beta$  be an assignment (possibly different than  $\alpha$ ) that satisfies  $\phi'$ . Since  $\beta$  satisfies  $\phi'$ , it satisfies  $\bigvee_{x \in \bar{V}} x$ . This means that there exists some  $x \in \bar{V}$  such that  $\beta(x) = 1$ . But  $\beta$  also satisfies  $(\neg x \vee \neg x')$ , so we get that  $\beta(x') = 0$ . Thus  $\beta(\neg x \vee x') = 0$ , which means that  $\phi'$  does not logically entail  $\neg x \vee x'$ . By the discussion above, there exists no path from  $x$  to  $x'$ , so  $x$  is not in a bad scc of  $G$ .  $\square$

By Proposition [7.2.2](#), we have seen that  $\phi$  is renamable partially Horn if and only if  $\phi'$  is satisfiable. Also, in case  $\phi'$  is satisfiable, a variable  $x \in V$  is renamed if and only if  $a(x) = 1$ .

Thus, by the above and Claim [7.2.4](#),  $\phi$  is renamable partially Horn if and only if there is some variable  $x$  that does not appear in a bad scc of  $G$ . Furthermore, the process described in order to obtain assignment  $a$  is linear in the length of the input formula, and  $a$  provides the information about which variables to rename.  $\square$

Because checking whether a formula is affine can be trivially done in linear time, we get:

**Theorem 7.2.2.** *There is an algorithm that, on input a formula  $\phi$ , halts in linear time in the length of  $\phi$  and either returns that  $\phi$  is not a possibility integrity constraint, or alternatively, (i) either it returns that  $\phi$  is affine or (ii) in case  $\phi$  is separable, it produces two non-empty and disjoint subsets  $V_1, V_2 \subseteq V$  such that no clause of  $\phi$  contains variables from both  $V_1$  and  $V_2$  and (iii) in case  $\phi$  is renamable partially Horn, it produces a subset  $V^* \subseteq V$  such that the formula  $\phi^*$  obtained from  $\phi$  by renaming the literals of variables in  $V^*$  is partially Horn.*

We proceed by showing that we can recognize lpic's efficiently.

**Theorem 7.2.3.** *There is an algorithm that, on input a formula  $\phi$ , halts in linear time in the length of  $\phi$  and either returns that  $\phi$  is not a local possibility constraint, or alternatively, produces the sets  $V_0, V_1$  described in Definition [5.3.6](#).*

*Proof.* We describe the algorithm in a high level way.

1. Read once each clause  $C$  of  $\phi$ . If  $C$  does not contain any  $\oplus$  connectives, put the variables of  $\text{vbl}(C)$  to  $V_0$ . Else, put the variables connected by  $\oplus$  to  $V_1$ .
2. Check the variables in  $V_0$  and  $V_1$ . If  $V_1 = V$ ,  $\phi$  is affine and thus an lpic. Else, if  $V_0 \cap V_1 \neq \emptyset$  or  $V_0 \cup V_1 \neq V$ ,  $\phi$  is not an lpic. Also, if  $V_0 = V$ ,  $\phi$  is an lpic if and only if it is a renamable Horn formula, something that can be checked in linear time by the algorithm of del Val [69].
3. Assume that  $(V_0, V_1)$  is a partition of  $V$ . We use the following variation of the linear algorithm presented in Theorem 7.2.1:
  - (a) Build the graph  $G = (\bar{V}, E)$  in the same way and compute its scc. Let  $\bar{V}_0 = V_0 \cup \{x' \mid x \in V_0\}$  and  $\bar{V}_1 = V_1 \cup \{x' \mid x \in V_1\}$ .
  - (b) If there is any variable of  $\bar{V}_0$  in a bad scc of  $G$ ,  $G$  is not an lpic. Else, set all variables of  $\bar{V}_1$  to 0.
  - (c) Let the scc's of  $G$ , in reverse topological order, be  $S_1, \dots, S_t$ . For each  $j = 1, \dots, t$  assign value 1 to every variable of  $S_j \cap \bar{V}_0$  that has not already received one (if  $S_j$  is bad no such variable exists). If some  $x \in \bar{V}_0$  of  $S_j$  takes value 1, then assign value 0 to  $x'$  (recall that  $(x')' = x$ ).
4. Let  $a$  be the resulting assignment to the variables of  $\bar{V}$ . By renaming all variables  $x \in V_0$  such that  $a(x) = 1$ , we obtain a partially Horn formula, whose non-admissible clauses are  $(V_0, V_1)$ -generalized. Thus,  $\phi$  is an lpic.

□

**Remark 7.2.1.** Recall that we have assumed that all the formulas we consider have non-degenerate domains. Note that the above algorithms cannot distinguish such formulas from other formulas of the same form that have degenerate domains. An algorithm that could efficiently decide that, would effectively be (due e.g. to the syntactic form of separable formulas) an algorithm that could decide on input any given formula, which variables are satisfied by exactly one Boolean value and which admit both.

The consequences of the existence of such an algorithm would be critical. For example, consider the known coNP-hard problem of unique satisfiability,

where we are interested in whether a propositional formula  $\phi$  is satisfied by exactly one assignment of values. We could use the aforementioned algorithm for each variable of the formula consecutively. If, at any point, we find a variable that is satisfied by both 0 and 1, then  $\phi$  has more than one satisfying assignment. Else, it doesn't.

## 7.2.2 Syntactic Characterization of (local) possibility domains

In this subsection, we provide syntactic characterizations for (local) possibility domains, by proving they are the models of (local) possibility integrity constraints. Furthermore, we show that given a (local) possibility domain  $D$ , we can produce a (local) possibility integrity constraint, whose set of models is  $D$ , in time polynomial in the size of  $D$ . To obtain the characterization for possibility domains, we proceed as follows. We separately show that each type of a possibility integrity constraint of Definition 5.3.5 corresponds to one of the conditions of Theorem 5.2.1: (i) Domains admitting non-dictatorial binary projection aggregators are the sets of models of separable formulas, those admitting non-projection binary aggregators are the sets of models of renamable partially Horn formulas and (iii) affine domains are the sets of models of affine formulas. For local possibility domains, we directly show they are the models of local possibility integrity constraints.

We will need some additional notation. For a set of indices  $I$ , let  $D_I := \{(a_i)_{i \in I} \mid a \in D\}$  be the projection of  $D$  to the indices of  $I$  and  $D_{-I} := D_{\{1, \dots, n\} \setminus I}$ . Also, for two (partial) vectors  $a = (a_1, \dots, a_k) \in D_{\{1, \dots, k\}}$ ,  $k < n$  and  $b = (b_1, \dots, b_{n-k}) \in D_{\{k+1, \dots, n\}}$ , we define their *concatenation* to be the vector  $ab = (a_1, \dots, a_k, b_1, \dots, b_{n-k})$ . Finally, given two subsets  $D, D' \subseteq \{0, 1\}^n$ , we write that  $D \approx D'$  if we can obtain  $D$  by *permuting* the coordinates of  $D'$ , i.e. if  $D = \{(d_{j_1}, \dots, d_{j_n}) \mid (d_1, \dots, d_n) \in D'\}$ , where  $\{j_1, \dots, j_n\} = \{1, \dots, n\}$ .

We begin with characterizing the domains closed under a non-dictatorial projection aggregator as the models of separable formulas.

**Proposition 7.2.3.**  $D \subseteq \{0, 1\}^n$  admits a binary non-dictatorial projection aggregator  $(f_1, \dots, f_n)$  if and only if there exists a separable formula  $\phi$  whose set of models equals  $D$ .

We will first need the following lemma:

**Lemma 7.2.1.**  *$D$  is closed under a binary non-dictatorial projection aggregator if and only if there exists a partition  $(I, J)$  of  $\{1, \dots, n\}$  such that  $D \approx D_I \times D_J$ .*

*Proof.* ( $\Rightarrow$ ) Let  $F = (f_1, \dots, f_n)$  be a binary non-dictatorial projection aggregator for  $D$ . Assume, without loss of generality, that  $f_i = \text{pr}_1^2$ ,  $i = 1, \dots, k < n$  and  $f_j = \text{pr}_2^2$ ,  $j = k + 1, \dots, n$ . Let also  $I := \{1, \dots, k\}$  and  $J := \{k + 1, \dots, n\}$ . Since  $k < n$ ,  $(I, J)$  is a partition of  $\{1, \dots, n\}$ . To prove that  $D = D_I \times D_J$ , it suffices to prove that  $D_I \times D_J \subseteq D$  (the reverse inclusion is always true).

Let  $a \in D_I$  and  $b \in D_J$ . It holds that there exists an  $a' \in D_I$  and a  $b' \in D_J$  such that both  $ab', a'b \in D$ . Thus:

$$F(ab', a'b) = ab \in D,$$

since  $F = (f_1, \dots, f_n)$  is an aggregator for  $D$ ,  $f_i = \text{pr}_1^2$ ,  $i \in I$  and  $f_j = \text{pr}_2^2$ ,  $j \in J$ .

( $\Leftarrow$ ) Suppose that  $D \approx D_I \times D_J$ , where  $I, J$  is a partition of  $\{1, \dots, n\}$ . Assume, without loss of generality, that  $I = \{1, \dots, k\}$ ,  $k < n$  and  $J = \{k + 1, \dots, n\}$  (thus  $D = D_I \times D_J$ ). Let also  $ab', a'b \in D$ , where  $a, a' \in D_I$  and  $b, b' \in D_J$ .

Obviously, if  $F = (f_1, \dots, f_n)$  is an  $n$ -tuple of projections, such that  $f_i = \text{pr}_1^2$ ,  $i \in I$  and  $f_j = \text{pr}_2^2$ ,  $j \in J$ , then  $F(ab', a'b) = ab \in D$ , since  $a \in D_I$  and  $b \in D_J$ . Thus  $F = (f_1, \dots, f_n)$  is a non-dictatorial projection aggregator for  $D$ .  $\square$

*Proof of Proposition 7.2.3.* ( $\Rightarrow$ ) Since  $D$  admits a binary non-dictatorial projection aggregator  $(f_1, \dots, f_n)$ , by Lemma 7.2.1,  $D \approx D_I \times D_J$ , where  $(I, J)$  is a partition of  $\{1, \dots, n\}$  such that  $I = \{i \mid f_i = \text{pr}_1^2\}$  and  $J = \{j \mid f_j = \text{pr}_2^2\}$ . Let  $\phi_1$  and  $\phi_2$  defined on  $\{x_i \mid i \in I\}$  and  $\{x_j \mid j \in J\}$  respectively, such that  $\text{Mod}(\phi_1) = D_I$  and  $\text{Mod}(\phi_2) = D_J$ . Let also  $\phi = \phi_1 \wedge \phi_2$ . It is straightforward to observe that, since  $\phi_1$  and  $\phi_2$  contain no common variables:

$$\text{Mod}(\phi) \approx \text{Mod}(\phi_1) \times \text{Mod}(\phi_2) = D_I \times D_J \approx D.$$

( $\Leftarrow$ ) Assume that  $\phi$  is separable and that  $\text{Mod}(\phi) = D$ . Since  $\phi$  is separable, we can find a partition  $(I, J)$  of  $\{1, \dots, n\}$ , a formula  $\phi_1$  defined on  $\{x_i \mid i \in I\}$  and a  $\phi_2$  defined on  $\{x_j \mid j \in J\}$ , such that  $\phi = \phi_1 \wedge \phi_2$ . Easily, it holds that:

$$\text{Mod}(\phi) \approx \text{Mod}(\phi_1) \times \text{Mod}(\phi_2) = D_I \times D_J \approx D.$$

The required now follows by Lemma 7.2.1.  $\square$

We now turn our attention to domains closed under binary non projection aggregators.

**Theorem 7.2.4.**  *$D$  admits a binary aggregator  $(f_1, \dots, f_n)$  which is not a projection aggregator if and only if there exists a renamable partially Horn formula  $\phi$  whose set of models equals  $D$ .*

We will first need two lemmas.

**Lemma 7.2.2.** *Suppose  $D$  admits a binary aggregator  $F = (f_1, \dots, f_n)$  such that there exists a partition  $(H, I, J)$  of  $\{1, \dots, n\}$  where  $f_h$  is symmetric for all  $h \in H$ ,  $f_i = \text{pr}_s^2$ , for all  $i \in I$  and  $f_j = \text{pr}_t^2$ , with  $t \neq s$ , for all  $j \in J$ . Then,  $D$  also admits a binary aggregator  $G = (g_1, \dots, g_n)$ , such that  $g_h = f_h$ , for all  $h \in H$  and  $g_i = \text{pr}_s^2$ , for all  $i \in I \cup J$ .*

*Proof.* Without loss of generality, assume that there exist  $1 \leq k < l < n$  such that  $H = \{1, \dots, k\}$ ,  $I = \{k+1, \dots, l\}$  and  $J = \{l+1, \dots, n\}$  and that  $s = 1$  (and thus  $t = 2$ ). It suffices to prove that, for two arbitrary vectors  $a, b \in D$ ,  $G(a, b) \in D$ , where  $(g_1, \dots, g_n)$  is defined as in the statement of the lemma.

Assume that for all  $i \in H$ ,  $f_i(a_i, b_i) = c_i$ . Since  $F$  is an aggregator for  $D$ , it holds that  $F(a, b)$  and  $F(b, a)$  are both vectors in  $D$ . By the same token, so is  $F(F(a, b), F(b, a))$ . The result is now obtained by noticing that:

$$\begin{aligned} F(a, b) &= (c_1, \dots, c_k, a_{k+1}, \dots, a_l, b_{l+1}, \dots, b_n), \\ F(b, a) &= (c_1, \dots, c_k, b_{k+1}, \dots, b_l, a_{l+1}, \dots, a_n), \end{aligned}$$

and thus:  $F(F(a, b), F(b, a)) = (c_1, \dots, c_k, a_{k+1}, \dots, a_n) = G(a, b)$ .  $\square$

**Lemma 7.2.3.** *Suppose  $D$  admits a binary aggregator  $(f_1, \dots, f_n)$  such that, for some  $J \subseteq \{1, \dots, n\}$ ,  $f_j$  is symmetric for all  $j \in J$ . For each  $d = (d_1, \dots, d_n) \in D$ , let  $d^* = (d_1^*, \dots, d_n^*)$  be such that:*

$$d_j^* = \begin{cases} 1 - d_j & \text{if } j \in J, \\ d_j & \text{else,} \end{cases}$$

for  $j = 1, \dots, n$  and set  $D^* = \{d^* \mid d \in D\}$ . Then  $D^*$  admits the binary aggregator  $(g_1, \dots, g_n)$ , where: (i)  $g_j = \wedge$  for all  $j \in J$  such that  $f_j = \vee$ , (ii)  $g_j = \vee$  for all  $j \in J$  such that  $f_j = \wedge$  and (iii)  $g_j = f_j$  for the rest.

Furthermore, if there are two formulas  $\phi$  and  $\phi^*$  such that  $\phi^*$  is obtained from  $\phi$  by renaming all  $x_j$ ,  $j \in J$ , then  $D = \text{Mod}(\phi)$  if and only if  $D^* = \text{Mod}(\phi^*)$ .

Note that we do not assume that the set  $J \subseteq \{1, \dots, n\}$  includes every coordinate  $j$  such that  $f_j$  is symmetric.

*Proof.* The former statement follows from the fact that  $\wedge(1 - d_j, 1 - d'_j) = 1 - \vee(d_j, d'_j)$  (resp.  $\vee(1 - d_j, 1 - d'_j) = 1 - \wedge(d_j, d'_j)$ ), for any  $d, d' \in D$ . For the latter, observe that by renaming  $x_j, j \in J$ , in  $\phi$ , we cause all of its literals to be satisfied by the opposite value. Thus,  $d^*$  satisfies  $\phi^*$  if and only if  $d$  satisfies  $\phi$ .  $\square$

For two vectors  $a, b \in D$ , we define  $a \leq b$  to mean that if  $a_i = 1$  then  $b_i = 1$ , for all  $i \in \{1, \dots, n\}$  and  $a < b$  when  $a \leq b$  and  $a \neq b$ .

*Proof of Theorem 7.2.4.* ( $\Rightarrow$ ) We will work with the corresponding domain  $D^*$  of Lemma 7.2.3 that admits an aggregator  $(g_1, \dots, g_n)$  whose symmetric components, corresponding to the symmetric components of  $(f_1, \dots, f_n)$ , are all equal to  $\wedge$ . Suppose that  $V_0 = \{x_i \mid g_i = \wedge\}$ . For  $D^*$ , we compute a formula  $\phi = \phi_0 \wedge \phi_1$ , where  $\phi_0$  is defined on the variables of  $V_0$  and is Horn and where  $\phi_1$  has only negative appearances of variables of  $V_0$ . The result is then derived by renaming all the variables  $x_j$ , where  $j$  is such that  $f_j = \vee$ .

Let  $I := \{i \mid f_i \text{ is symmetric}\}$  (by the hypothesis,  $I \neq \emptyset$ ). Let also  $J := \{j \mid f_j = \vee\}$  ( $J$  might be empty). Obviously  $J \subseteq I$ . For each  $d = (d_1, \dots, d_n) \in D$ , let  $d^* = (d_1^*, \dots, d_n^*)$ , where  $d_j^* = 1 - d_j$  if  $j \in J$  and  $d_i^* = d_i$  else. Easily, if  $D^* = \{d^* \mid d \in D\}$ , by Lemma 7.2.3 it admits an aggregator  $(g_1, \dots, g_n)$  such that  $g_i = \wedge$ , for all  $i \in I$  and  $g_j = \text{pr}_1^2$ ,  $j \notin I$ . Thus, there is a Horn formula  $\phi_0$  on  $\{x_i \mid i \in I\} := V_0$ , such that  $\text{Mod}(\phi_0) = D_I^*$ .

If  $I = \{1, \dots, n\}$ , we have nothing to prove. Thus, suppose, without loss of generality, that  $I = \{1, \dots, k\}$ ,  $k < n$ . For each  $a = (a_1, \dots, a_k) \in D_I^*$ , let  $B_a := \{b \in D_{-I}^* \mid ab \in D^*\}$  be the set containing all partial vectors that can extend  $a$ . For each  $a \in D_I^*$ , let  $\psi_a$  be a formula on  $\{x_j \mid j \notin I\}$ , such that  $\text{Mod}(\psi_a) = B_a$ . Finally, let  $I_a := \{i \in I \mid a_i = 1\}$  and define:

$$\phi_a := \left( \bigwedge_{i \in I_a} x_i \right) \rightarrow \psi_a,$$

for all  $a \in D_I^*$ .

Consider the formula:

$$\phi = \phi_0 \wedge \left( \bigwedge_{a \in D_I^*} \phi_a \right).$$

We will prove that  $\phi$  is partially Horn and that  $\text{Mod}(\phi) = D^*$ . By Lemma [7.2.3](#), the renamable partially Horn formula for  $D$  can be obtained by renaming in  $\phi$  the variables  $x_i$  such that  $i \in J$ .

We have already argued that  $\phi_0$  is Horn. Also, since  $\phi_a$  is *logically equivalent* to (has exactly the same models as):

$$\left( \bigvee_{i \in I_a} \neg x_i \right) \vee \psi_a,$$

any variable of  $V_0$  that appears in the clauses of some  $\phi_a$ , does so negatively. It follows that  $\phi$  is partially Horn.

Next we show that  $D^* \subseteq \text{Mod}(\phi)$  and that  $\text{Mod}(\phi) \subseteq D^*$ . For the former inclusion, let  $ab \in D^*$ , where  $a \in D_I^*$  and  $b \in B_a$ . Then, it holds that  $a$  satisfies  $\phi_0$  and  $b$  satisfies  $\psi_a$ . Thus  $ab$  satisfies  $\phi_a$ .

Now, let  $a' \in D_I^* : a \not\leq a'$ . Then,  $a$  does not satisfy  $\bigwedge_{i \in I_{a'}} x_i$ , since there exists some coordinate  $i \in I_{a'}$  such that  $a_i = 0$  and  $a'_i = 1$ . Thus,  $ab$  satisfies  $\phi_{a'}$ . Finally, let  $a'' \in D_I^* : a'' < a$ . Then,  $a$  satisfies  $\bigwedge_{i \in I_{a''}} x_i$  and thus we must prove that  $b$  satisfies  $\psi_{a''}$ .

Since  $a'' \in D_I^*$ , there exists a  $c \in D_{-I}^*$  such that  $a''c \in D^*$ . Then, since  $(g_1, \dots, g_n)$  is an aggregator for  $D^*$ :

$$\begin{aligned} (g_1, \dots, g_n)(ab, a''c) = \\ (\wedge(a_1, a''_1), \dots, \wedge(a_k, a''_k), \text{pr}_1^2(b_1, c_1), \dots, \text{pr}_1^2(b_{n-k}, c_{n-k})) = a''b \in D^*, \end{aligned}$$

since  $a'' < a$ . Thus,  $b \in B(a'')$  and, consequently, it satisfies  $\psi_{a''}$ .

We will prove the opposite inclusion by showing that an assignment not in  $D^*$  cannot satisfy  $\phi$ . Let  $ab \notin D^*$ . If  $a \notin D_I^*$ , we have nothing to prove, since  $a$  does not satisfy  $\phi_0$  and thus  $ab \notin \text{Mod}(\phi)$ . So, let  $a \in D_I^*$ . Then,  $b \notin B_a$ , lest  $ab \in D^*$ . But then,  $b$  does not satisfy  $\psi_a$  and thus  $ab$  does not satisfy  $\phi_a$ . Consequently,  $ab \notin \text{Mod}(\phi)$ .

Thus, by renaming the variables  $x_i$ ,  $i \in J$ , we produce a renamable partially Horn formula, call it  $\psi$ , such that  $\text{Mod}(\psi) = D$ .

( $\Leftarrow$ ) Let  $\psi$  be a renamable partially Horn formula with  $\text{Mod}(\psi) = D$ . Let  $J \subseteq \{1, \dots, n\}$  such that, by renaming all the  $x_i$ ,  $i \in J$ , in  $\psi$ , we obtain a partially Horn formula  $\phi$ . Let  $V_0$  be the set of variables such that any clause containing only variables from  $V_0$  is Horn, and that appear only negatively in clauses that contain variables from  $V \setminus V_0$ . By Remark [5.3.2](#), we can assume



that  $\{x_i \mid i \in J\} \subseteq V_0$ . Let also  $\mathcal{C}_0$  be the set of admissible Horn clauses of  $\phi$ .

Let again  $D^* = \{d^* \mid d \in D\}$ , where  $d_j^* = 1 - d_j$  if  $j \in J$  and  $d_i^* = d_i$  else, for all  $d \in D$ . By Lemma 7.2.3,  $\text{Mod}(\phi) = D^*$ . By the same Lemma, and by noticing that whichever the choice of  $J \subseteq \{1, \dots, n\}$ ,  $(D^*)^* = D$ , it suffices to prove that  $D^*$  is closed under a binary aggregator  $(f_1, \dots, f_n)$ , where  $f_i = \wedge$  for all  $i$  such that  $x_i \in V_0$  and  $f_j = \text{pr}_1^2$  for the rest.

Without loss of generality, let  $I = \{1, \dots, k\}$ ,  $k < n$  (lest we have nothing to show) be the set of indices of the variables in  $V_0$ . We need to show that if  $ab, a'b' \in D$ , where  $a, a' \in D_I^*$  and  $b, b' \in D_{-I}^*$ , then  $(a \wedge a')b \in D$ , where  $a \wedge a' = (a_1 \wedge a'_1, \dots, a_k \wedge a'_k)$ .

Let  $\phi = \phi_0 \wedge \phi_1$ , where  $\phi_0$  is the conjunction of the clauses in  $\mathcal{C}_0$  and  $\phi_1$  the conjunction of the rest of the clauses of  $\phi$ . By the hypothesis,  $\phi_0$  is Horn and thus, since  $a, a'$  satisfy  $\phi_0$ , so does  $a \wedge a'$ . Now, let  $C_r$  be a clause of  $\phi_1$ . If any literal of  $C_r$  that corresponds to a variable not in  $\phi_0$  is satisfied by  $b$ , we have nothing to prove. If there is no such literal, since  $ab$  satisfies  $C_r$ , it must hold that a negative literal  $\bar{x}_i$ ,  $i \in I$ , is satisfied by  $a$ . Thus,  $a_i = 0$ , which means that  $a_i \wedge a'_i = 0$  too. Consequently,  $C_r$  is satisfied by  $(a \wedge a')b$ . Since  $C_r$  was arbitrary, the proof is complete.  $\square$

We thus get:

**Theorem 7.2.5.**  *$D$  is a possibility domain if and only if there exists a possibility integrity constraint  $\phi$  whose set of models equals  $D$ .*

*Proof.* ( $\Rightarrow$ ) If  $D$  is a possibility domain, then, by Theorem 5.2.1, it either admits a non-dictatorial binary projection, or a non-projection binary aggregator or a ternary aggregator all components of which are the binary addition mod 2. In the first case, by Proposition 7.2.3,  $D$  is the model set of a separable formula. In the second, by Theorem 7.2.4, it is the model set of a renamable partially Horn formula and in the third, that of an affine formula. Thus, in all cases,  $D$  is the model set of a possibility integrity constraint.

( $\Leftarrow$ ) Let  $\phi$  be a possibility integrity constraint such that  $\text{Mod}(\phi) = D$ . If  $\phi$  is separable, then, by Proposition 7.2.3,  $D$  admits a non-dictatorial binary projection aggregator. If  $\phi$  is renamable partially Horn, then, by Theorem 7.2.4,  $D$  admits a non-projection binary aggregator. Finally, if  $\phi$  is affine, then  $D$  admits a ternary aggregator all components of which are the binary addition mod 2. In every case,  $D$  is a possibility domain.  $\square$

We turn now our attention to local possibility domains. Analogously to the case of possibility domains, we characterize lpd's as the sets of models of lpic's.

**Theorem 7.2.6.** *A domain  $D \subseteq \{0, 1\}^n$  is a local possibility domain if and only if there is a local possibility integrity constraint  $\phi$  such that  $\text{Mod}(\phi) = D$ .*

We will first need two lemmas.

**Lemma 7.2.4.** *Let  $D \subseteq \{0, 1\}^n$  and  $I = \{j_1, \dots, j_t\} \subseteq \{1, \dots, n\}$ . Then, if  $F = (f_1, \dots, f_n)$  is a  $k$ -ary aggregator for  $D$ ,  $(f_{j_1}, \dots, f_{j_t})$  is a  $k$ -ary aggregator for  $D_I$ .*

*Proof.* Without loss of generality, assume  $I = \{1, \dots, s\}$ , where  $s \leq n$  and let  $a^1, \dots, a^k \in D_I$ . It follows that there exist  $b^1, \dots, b^k \in D_{-I}$  such that  $c^1, \dots, c^k \in D$ , where  $c^i = a^i b^i$ ,  $i = 1, \dots, k$ . Since  $F$  is an aggregator for  $D$ :

$$F(c^1, \dots, c^k) := (f_1(c_1^1, \dots, c_1^k), \dots, f_n(c_n^1, \dots, c_n^k)) \in D.$$

Thus,  $(f_1(c_1^1, \dots, c_1^k), \dots, f_s(c_s^1, \dots, c_s^k)) \in D_I$ . □

**Lemma 7.2.5.** *Suppose that  $D$  admits a ternary aggregator  $F = (f_1, \dots, f_n)$ , where  $f_j \in \{\wedge^{(3)}, \oplus, \text{maj}\}$ ,  $j = 1, \dots, n$ . Then  $D$  admits a binary aggregator  $G = (g_1, \dots, g_n)$  such that  $g_i = \wedge$ , for all  $i$  such that  $f_i = \wedge^{(3)}$ ,  $g_j = \text{pr}_2^2$ , for all  $j$  such that  $f_j = \oplus$  and  $g_k = \text{pr}_2^2$ , for all  $k$  such that  $f_k = \text{maj}$ .*

*Proof.* The result is immediate, by defining  $G = (g_1, \dots, g_n)$  such that:

$$g_j(x, y) = f_j(x, x, y),$$

for  $j = 1, \dots, n$ . □

*Proof of Theorem 7.2.6.* ( $\Rightarrow$ ) The proof will closely follow that of Theorem 7.2.4.

Since  $D$  is an lpd, by Theorem 7.1.4, there is a ternary aggregator  $F = (f_1, \dots, f_n)$  such that every component  $f_j \in \{\wedge^{(3)}, \vee^{(3)}, \oplus, \text{maj}\}$ ,  $j = 1, \dots, n$ . Again, let  $D^* = \{d^* \mid d \in D\}$ , where  $d_j^* = 1 - d_j$  if  $j$  is such that  $f_j = \vee^{(3)}$ , and  $d_j^* = d_j$  in any other case. Thus, by Lemma 7.2.3,  $D^*$  admits a ternary aggregator  $G = (g_1, \dots, g_n)$  such that  $g_j \in \{\wedge^{(3)}, \oplus, \text{maj}\}$ , for  $j = 1, \dots, n$ . Thus, by showing that  $D^*$  is described by a lpic  $\phi$ , we will obtain the same result for  $D$  by renaming all the variables  $x_j$ , where  $j$  is such that  $f_j = \vee^{(3)}$ .

Without loss of generality, assume that  $I := \{i \mid g_i = \wedge^{(3)}\} = \{1, \dots, s\}$ ,  $J := \{j \mid g_j = \oplus\} = \{s+1, \dots, t\}$  and  $K := \{k \mid g_k = \text{maj}\} = \{t+1, \dots, n\}$ , where  $0 \leq s \leq t \leq n$ . Since  $D_I^*$  is Horn, there is a Horn formula  $\phi_0$  such that  $\text{Mod}(D_I^*) = \phi_0$ .

If  $s = t = n$ , we have nothing to prove. Thus, suppose  $s < t \leq n$ . For each  $a = (a_1, \dots, a_s) \in D_I^*$ , let  $B_a^1 := \{b \in D_J^* \mid ab \in D_{I \cup J}^*\}$  and  $B_a^2 := \{c \in D_K^* \mid ac \in D_{I \cup K}^*\}$  be the sets of partial vectors extending  $a$  to the indices of  $J$  and  $K$  respectively.

**Claim 7.2.5.** *For each  $a \in D_I^*$ ,  $B_a^1$  and  $B_a^2$  are affine and bijunctive respectively.*

*Proof of Claim:* We will prove the claim for  $B_a^2$ . The proof for  $B_a^1$  is the same.

Let  $b^1, b^2, b^3 \in B_a^2$ . Then  $ab^1, ab^2, ab^3 \in D_{I \cup K}^*$ . Since, by Lemma 7.2.4,  $(g_1, \dots, g_t)$  is an aggregator for  $D_{I \cup K}^*$  and by the definition of  $G$ , it holds that  $ab \in D_{I \cup K}^*$ , where  $b = \text{maj}(b^1, b^2, b^3)$ . Thus,  $b \in B_a^2$  and the result follows.  $\square$

Thus, for each  $a \in D_I^*$ , there is an affine formula  $\psi_a$  and a bijunctive  $\chi_a$ , such that  $\text{Mod}(\psi_a) = B_a^1$  and  $\text{Mod}(\chi_a) = B_a^2$ . Let  $I_a := \{i \in I \mid a_i = 1\}$  and define:

$$\phi_a^1 := \left( \bigwedge_{i \in I_a} x_i \right) \rightarrow \psi_a$$

and

$$\phi_a^2 := \left( \bigwedge_{i \in I_a} x_i \right) \rightarrow \chi_a,$$

for all  $a \in D_I^*$ .

Consider the formula:

$$\phi = \phi_0 \wedge \left( \bigwedge_{a \in D_I^*} \phi_a^1 \right) \wedge \left( \bigwedge_{a \in D_I^*} \phi_a^2 \right).$$

Let  $V_0 = \{x_i \mid i \in I\}$ ,  $V_1 = \{x_j \mid j \in J\}$  and  $V_2 = \{x_k \mid k \in K\}$ . That  $\phi$  is partially Horn with admissible set  $V_0$ , can be seen in the same way as in Theorem 7.2.4. Now, consider  $\psi_a$ , for some  $a \in D_I^*$ . Since it is affine, it is of the form:

$$\psi_a = \bigwedge_{j=1}^r \left( l_{j_1} \oplus \dots \oplus l_{j_t} \right),$$

where  $l_{j_i}$  are literals of variables from  $V_1$ . Thus,  $\phi_a^1$  is equivalent to:

$$\bigwedge_{j=1}^r \left( \left( \bigvee_{i \in I_a} \neg x_i \right) \vee (l_{j_1} \oplus \cdots \oplus l_{j_i}) \right).$$

Thus, the clauses of  $\phi_a^1$  are  $(V_0, V_1)$ -generalized.

Now consider  $\chi_a$ , for some  $a \in D_I^*$ . Since it is bijunctive, it is of the form:

$$\chi_a = \bigwedge_{s=1}^t \left( l_{s_1} \vee l_{s_2} \right),$$

where  $l_{s_1}, l_{s_2}$  are (not necessarily distinct) literals of variables from  $V_2$ . Thus,  $\phi_a^2$  is equivalent to:

$$\bigwedge_{s=1}^t \left( \left( \bigvee_{i \in I_a} \neg x_i \right) \vee (l_{s_1} \vee l_{s_2}) \right).$$

To prove that  $\phi$  is an lpic, we will show that  $V_2$  can be renamed to that  $V_0 \cup V_2$  become a set of admissible variables for the renamed formula. Consider the clauses of

$$\chi := \left( \bigwedge_{a \in D_I^*} \phi_a^2 \right) = \bigwedge_{a \in D_I^*} \left( \bigwedge_{s=1}^t \left( \left( \bigvee_{i \in I_a} \neg x_i \right) \vee (l_{s_1} \vee l_{s_2}) \right) \right).$$

They are comprised of negative appearances of variables from  $V_0$  and of at most two literals from  $V_2$ . Since  $V_0$  and  $V_2$  are disjoint, we can project  $\chi$  to the variables of  $V_2$ , to obtain a satisfiable bijunctive formula. Thus  $\chi_{V_2}$  is renamable Horn. It is obvious that the same renaming, will result in a renaming for  $\chi$  such that all its clauses contain at most one positive literal.

What remains now is to show that  $\text{Mod}(\phi) = D^*$ . By Lemmas [7.2.2](#) and [7.2.5](#), it follows that  $D^*$  admits a binary aggregator  $H = (h_1, \dots, h_n)$  such that  $h_i = \wedge$ , for all  $i \in I$  and  $h_j = \text{pr}_1^2$ , for all  $j \in J \cup K$ . The proof now is exactly like the one of Theorem [7.2.4](#), by letting  $B_a = \{bc \mid b \in B_a^1 \text{ and } c \in B_a^2\}$  and

$$\phi_a = \phi_a^1 \wedge \phi_a^2.$$

( $\Leftarrow$ ) Let  $\psi$  be an lpic, with  $\text{Mod}(\psi) = D$ . Let  $V_0$  and  $V_1$  be subsets of  $V$  as in Definition [5.3.6](#). Let also  $\phi$  be the partially Horn formula obtained by  $\psi$  by

renaming the variables of a subset  $V^* \subseteq V_0$ . Again, assume  $D^* = \{d^* \mid d \in D\}$ , where  $d_j^* = 1 - d_j$  if  $x_j \in V^*$  and  $d_i^* = d_i$  else, for all  $d \in D$ . By Lemma 7.2.3,  $\text{Mod}(\phi) = D^*$ . Thus, by Theorem 7.1.4, it suffices to prove that  $D^*$  is closed under a ternary aggregator  $(f_1, \dots, f_n)$ , where  $f_i \in \{\wedge^{(3)}, \text{maj}, \oplus\}$  for  $i = 1, \dots, n$ . We will in fact show that we do not need any maj components.

Without loss of generality, let  $I = \{1, \dots, s\}$ , be the set of indices of the variables in  $V_0$  and  $J = \{s+1, \dots, n\}$  be that of the indices of variables in  $V_1$ . We will show that if  $ab, a'b', a''b'' \in D^*$ , where  $a, a', a'' \in D_I^*$  and  $b, b', b'' \in D_J^*$ , then

$$d := (\wedge^{(3)}(a, a', a''), \oplus(b, b', b'')) \in D^*.$$

Let  $\phi = \phi_0 \wedge \phi_1$ , where  $\phi_0$  is the conjunction of the clauses containing only variables from  $V_0$  and  $\phi_1$  the conjunction of  $(V_0, V_1)$ -generalized clauses. By the hypothesis,  $\phi_0$  is Horn and thus, since  $a, a', a''$  satisfy  $\phi_0$ , so does  $a \wedge a' \wedge a''$ .

Now, let  $C_r$  be a clause of  $\phi_1$ . Suppose that there is a literal of a variable  $x_i \in V_0$  in  $C_r$  that is satisfied by  $a$ . Since  $\phi$  is partially Horn with respect to  $V_0$ , it must hold that this literal was  $\neg x_i$ . This means that  $a_i = 0$  and thus  $\wedge^{(3)}(a_i, a'_i, a''_i) = 0$ . The same holds if  $\neg x_i$  is satisfied by  $a'$  or  $a''$ . Thus,  $C_r$  is satisfied.

Now, suppose there is no such literal and that the and that the sub-clause of  $C_r$  obtained by deleting the variables of  $V_0$  is:

$$C'_r = (l_1 \oplus \dots \oplus l_z).$$

Since  $ab, a'b', a''b''$  satisfy  $\phi$ , it holds that  $b, b'$  and  $b''$  satisfy  $C'_r$ . Since  $C'_r$  is affine, it holds that  $\oplus(b, b', b'')$  satisfies it.

In all cases, we proved that  $d$  satisfies  $\phi$  and thus the proof is complete.  $\square$

Recall that by Theorem 7.1.4, lpd's are characterized as the domains admitting an aggregator  $(f_1, \dots, f_n)$ , where  $f_i \in \{\wedge^{(3)}, \vee^{(3)}, \oplus, \text{maj}\}$  for  $i = 1, \dots, n$ . Observe though that by the proof of Theorem 7.2.6, we immediately obtain the following result.

**Corollary 7.2.1.**  *$D \subseteq \{0, 1\}^n$  is a local possibility domain if and only if it admits a ternary aggregator  $(f_1, \dots, f_n)$  such that  $f_j \in \{\wedge^{(3)}, \vee^{(3)}, \oplus\}$ , for  $j = 1, \dots, n$ .*

The above result is a strengthening of the corresponding characterization of such domains in the non-Boolean framework, where a domain is taken

to be a subset of  $\prod_{j=1}^m A_j$ , with  $A_j$ s being arbitrary finite sets with at least two elements. It easily follows from [144, Theorem 5.5]<sup>2</sup> that in this more general framework, a domain is a local possibility domain<sup>2</sup>, if and only if it admits a ternary aggregator  $F = (f_1, \dots, f_n)$  such that, for all  $j = 1, \dots, n$  and all two-element subsets  $B_j \subseteq D_j$ , taken as  $\{0, 1\}$ , we have that  $f_j \upharpoonright B_j$  is one of the ternary operations  $\wedge^{(3)}$ ,  $\vee^{(3)}$ ,  $\text{maj}$ ,  $\oplus$  (to which of these four ternary operations the restriction  $f_j \upharpoonright B_j$  is equal to depends on  $j$  and  $B_j$ ). Corollary [7.2.1] is a strengthening in the sense that in the Boolean framework, the  $\text{maj}$  case is missing. Interestingly, we were not able to prove this strengthening without the use of our syntactic characterization given in Theorem [7.2.6] above. This perhaps explains why Corollary [7.2.1] was not previously obtained, despite the fact that for the analogous case of (plain) possibility domains, both a characterization for the non-Boolean framework was given in [144, Theorem 3.1], and a strengthening for the Boolean case, where  $\text{maj}$  is missing, was already known ([76, Theorem 2.1 and Claim 3.6] or Dietrich and List [74, Theorem 2]).

### 7.2.3 Efficient constructions

To finish this section, we will use Zanuttini and Hébrard’s “unified framework” [219]. Recall the definition of a prime formula (Def. [5.3.7]) and consider the following proposition:

**Proposition 7.2.4.** *Let  $\phi_P$  be a prime formula and  $\phi$  be a formula logically equivalent to  $\phi_P$ . Then:*

1. *if  $\phi$  is separable,  $\phi_P$  is also separable and*
2. *if  $\phi$  is renamable partially Horn,  $\phi_P$  is also renamable partially Horn.*

*Proof.* Let  $\phi_P$  be a prime formula. Quine [187] showed that the prime implicates of  $\phi_P$  can be obtained from any formula  $\phi$  logically equivalent to  $\phi_P$ , by repeated (i) resolution and (ii) omission of the clauses that have sub-clauses already created. Thus, using the procedures (i) and (ii) on  $\phi$ , we can obtain every clause of  $\phi_P$ .

If  $\phi$  is separable, where  $(V', V \setminus V')$  is the partition of its vertex set such that no clause contains variables from both  $V'$  and  $V \setminus V'$ , it is obvious that

---

<sup>2</sup>In [144] the terminology “uniform possibility domain” is used.

neither resolution or omission can create a clause that destroys that property. Thus,  $\phi_P$  is separable.

Now, let  $\phi$  be a renamable partially Horn formula where, by renaming the variables of  $V^* \subseteq V$ , we obtain the partially Horn formula  $\phi^*$ , whose admissible set of variables is  $V_0$ . Let also  $\phi_P^*$  be the formula obtained by renaming the variables of  $V^*$  in  $\phi_P$ . Easily,  $\phi_P^*$  is prime.

Observe that the prime implicates of a partially Horn formula, are also partially Horn. Indeed, it is not difficult to observe that neither resolution, nor omission can cause a variable to cease being admissible: suppose  $x \in V_0$ . Then, the only way that it can appear in an inadmissible set due to resolution is if there is an admissible Horn clause  $C$  containing  $\neg x, y$ , where  $y \in V_0$  too and an inadmissible clause  $C'$  containing  $\neg y$ . But then, after using resolution,  $x$  appears negatively to the newly obtained clause. Thus,  $\phi_P^*$  is partially Horn, which means that  $\phi_P$  is renamable partially Horn.  $\square$

We are now ready to prove our first main result:

**Theorem 7.2.7.** *There is an algorithm that, on input  $D \subseteq \{0, 1\}^n$ , halts in time  $O(|D|^2 n^2)$  and either returns that  $D$  is not a possibility domain, or alternatively outputs a possibility integrity constraint  $\phi$ , containing  $O(|D|n)$  clauses, whose set of satisfying truth assignments is  $D$ .*

*Proof.* Given a domain  $D$ , we first use Zanuttini and Hébrard's algorithm to check if it is affine [219, Proposition 8], and if it is, produce, in time  $O(|D|^2 n^2)$  an affine formula  $\phi$  with  $O(|D|n)$  clauses, such that  $\text{Mod}(\phi) = D$ . If it isn't, we use again Zanuttini and Hébrard's algorithm [219] to produce, in time  $O(|D|^2 n^2)$ , a prime formula  $\phi$  with  $O(|D|n)$  clauses, such that  $\text{Mod}(\phi) = D$ . Then, we use the linear algorithms of Proposition 7.2.1 and Theorem 7.2.1 to check if  $\phi$  is separable or renamable partially Horn. If it is either of the two, then  $\phi$  is a possibility integrity constraint and, by Theorem 7.2.5,  $D$  is a possibility domain. Else, by Proposition 7.2.4,  $D$  is not a possibility domain.  $\square$

We end this section by proving our second main result, that given an lpd  $D$ , we can efficiently construct an lpic  $\phi$  such that  $\text{Mod}(\phi) = D$ .

**Theorem 7.2.8.** *There is an algorithm that, on input  $D \subseteq \{0, 1\}^n$ , halts in time  $O(|D|^2 n^2)$  and either returns that  $D$  is not a local possibility domain, or alternatively outputs a local possibility integrity constraint  $\phi$ , containing  $O(|D|n)$  clauses, whose set of satisfying truth assignments is  $D$ .*

We first briefly discuss some results of Zanuttini and Hébrard. For a clause  $C = l_1 \vee \dots \vee l_t$ , where  $l_j$  are literals,  $j = 1, \dots, t$ , let  $E(C) = l_1 \oplus \dots \oplus l_t$ . For a CNF formula  $\phi = \bigwedge_{j=1}^m C_j$ , let  $A(\phi) = \bigwedge_{j=1}^m E(C_j)$ . In [219, Proposition 8], it is proven that if  $\phi$  is prime,  $\text{Mod}(\phi) = D$  and  $D$  is affine, then  $\text{Mod}(A(\phi)) = D$ .

*Proof of Theorem 7.2.8.* Given a domain  $D$ , we first use the algorithm of Zanuttini and Hébrard [219] to produce, in time  $O(|D|^2 n^2)$ , a prime formula  $\phi$  with  $O(|D|n)$  clauses, such that  $\text{Mod}(\phi) = D$ . Note that at this point,  $\phi$  does not contain any generalized clauses (see below). We then use the linear algorithm of Theorem 7.2.1 to produce a set  $V_0$  such that  $\phi$  is renamable partially Horn with admissible set  $V_0$ .

If  $V_0 = V$  we have nothing to prove. Thus, suppose that  $\phi = \phi_0 \wedge \phi_1$ , where  $\phi_0$  contains only variables from  $V_0$ . Let  $\phi'_1$  be the sub-formula of  $\phi_1$ , obtained by deleting all variables of  $V_0$  from  $\phi$ . We then check, with Zanuttini and Hébrard's algorithm, if  $\phi'_1$  is affine. If not, then  $D$  is not an lpd. If it is, we construct the formula  $A^*(\phi_1)$  as follows. For each clause  $C = (l_1 \vee \dots \vee l_s \vee (l_{s+1} \vee \dots \vee l_t))$ , where  $l_1, \dots, l_s$  are literals of variables in  $V_0$ , let:

$$E^*(C) = (l_1 \vee \dots \vee l_s \vee (l_{s+1} \oplus \dots \oplus l_t))$$

and  $A^*(\phi_1) = \bigwedge_{j=1}^m E^*(C_j)$ . Then, the lpic that describes  $D$  is  $\phi_0 \wedge A^*(\phi_1)$ .  $\square$

## 7.2.4 Other Forms of non-Dictatorial Aggregation

In this subsection, we discuss four different notions of non-dictatorial aggregation procedures that have been introduced in the field of judgment aggregation: aggregators that are not generalized dictatorships, and anonymous, monotone and StrongDem aggregators. We prove that pic's, lpic's, a sub-class of pic's, and a subclass of lpic's, respectively, describe domains that admit each of the above four kind of aggregators. Then, we consider the property of systematicity and examine how our results change if the aggregators are required to satisfy it.

**Generalized Dictatorships** Recall the definition of generalized dictatorships.



**Definition 7.2.1.** Let  $F = (f_1, \dots, f_n)$  be an  $n$ -tuple of  $k$ -ary conservative functions.  $F$  is a generalized dictatorship for a domain  $D \subseteq \{0, 1\}^n$ , if, for any  $x^1, \dots, x^k \in D$ , it holds that:

$$F(x^1, \dots, x^k) := (f_1(x_1), \dots, f_n(x_n)) \in \{x^1, \dots, x^k\}. \quad (7.16)$$

Much like dictatorial functions, it is straightforward to observe that if  $F$  is a generalized dictator for  $D$ , then it is also an aggregator for  $D$ .

It should be noted here that in the original definition of Grandi and Endriss [112], generalized dictatorships are defined independently of a specific domain. Specifically, condition (7.16) is required to hold for all  $x^1, \dots, x^k \in \{0, 1\}^n$ . With this stronger definition, they show that the class of generalized dictators coincides with that of functions that are aggregators for every domain  $D \subseteq \{0, 1\}^n$ .

**Remark 7.2.2.** The difference in the definition of generalized dictatorships comes from a difference in the framework we use. Here, we opt to consider the aggregators restricted in the given domain, in the sense that we are not interested in what they do on inputs that are not allowed by it. The implications of this are not very evident in the Boolean framework, especially since we consider aggregators that satisfy IIA, on non-degenerate domains (in fact, this issue will not arise in any other aggregator present in this work, apart from generalized dictatorships). On the other hand, in the non-Boolean framework, using unrestricted aggregators could result in trivial cases of non-dictatorial aggregation, where the aggregator is not a projection only on inputs that are not allowed by the domain.

The following example shows that the result of Grandi and Endriss [112, Theorem 16] does not hold in our setting.

**Example 7.2.1.** Consider the Horn formula:

$$\phi_{11} = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3),$$

whose set of satisfying assignments is:

$$\text{Mod}(\phi_{11}) = \{(0, 0, 0), (0, 0, 1), (0, 1, 0), (1, 0, 0)\}.$$

By definition,  $\text{Mod}(\phi_{11})$  is a Horn domain and it thus admits the binary symmetric aggregator  $\bar{\wedge} = (\wedge, \wedge, \wedge)$ . Furthermore,  $\bar{\wedge}$  is not a generalized dictatorship for  $\text{Mod}(\phi_{11})$ , since:

$$\bar{\wedge}((0, 0, 1), (0, 1, 0)) = (0, 0, 0) \notin \{(0, 0, 1), (0, 1, 0)\}.$$

On the other hand, consider the Horn formula:

$$\phi_{12} = (\neg x_1 \vee x_2) \wedge (x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3).$$

$\bar{\wedge}$  is again an aggregator for the Horn domain:

$$\text{Mod}(\phi_{12}) = \{(0, 0, 0), (0, 1, 0), (0, 1, 1), (1, 1, 1)\},$$

but, contrary to the previous case,  $\bar{\wedge}$  is a generalized dictatorship for the domain  $\text{Mod}(\phi_{12})$ , since it is easy to verify that for any  $x, y \in D'$ ,  $\bar{\wedge}(x, y) \in \{x, y\}$ .

Finally, observe that  $(\wedge, \vee, \vee)$  is an aggregator for  $\text{Mod}(\phi_{12})$  that is not a generalized dictatorship. The latter claim follows from the fact that:

$$(\wedge, \vee, \vee)((0, 1, 0), (1, 1, 1)) = (0, 1, 1) \notin \{(0, 1, 0), (1, 1, 1)\},$$

while the former is left to the reader. Thus, interestingly enough,  $\phi_{12}$  describes a domain admitting an aggregator that is not a generalized dictatorship, although it is not the aggregator that “corresponds” to the formula.  $\diamond$

It is easy to see that  $(\text{pr}_i^k, \dots, \text{pr}_i^k)$  is a generalized dictatorship of any  $D \subseteq \{0, 1\}^n$ , for all  $k \geq 1$  and for all  $i \in \{1, \dots, k\}$ . Thus, trivially, every domain admits aggregators which are generalized dictatorships. On the other hand, every domain  $D \subseteq \{0, 1\}^n$  containing only two elements (a domain cannot contain less than two due to non-degeneracy) admits only generalized dictatorships. Indeed, assume  $D = \{x, y\}$ ,  $x \neq y$  and let  $F$  be a  $k$ -ary aggregator for  $D$ . Obviously,  $F(x, \dots, x) = x$  and  $F(y, \dots, y) = y$ , since  $F$  is unanimous. Also,  $F(x, y) \in \{x, y\} = D$  since  $F$  is an aggregator.

Our aim is again to find a syntactic characterization for domains that admit aggregators which are not generalized dictatorships. The following result shows that these domains are all the possibility domains with at least three elements, and are thus characterized by possibility integrity constraints.

**Theorem 7.2.9.** *A domain  $D \subseteq \{0, 1\}^n$ , with at least three elements, admits an aggregator that is not a generalized dictatorship if and only if it is a possibility domain.*

*Proof.* The forward direction is obtained by the trivial fact that an aggregator that is not a generalized dictatorship is also non-dictatorial.

Now, suppose that  $D$  is a possibility domain. Then it is either affine or it admits a binary non-dictatorial aggregator. We begin with the affine case.

It is a known result that  $D \subseteq \{0, 1\}^n$  is affine if and only if it is closed under  $\oplus$ , or, equivalently, if it admits the minority aggregator:

$$\bar{\oplus} = \underbrace{(\oplus, \dots, \oplus)}_{n\text{-times}}$$

**Claim 7.2.6.** *Let  $D \subseteq \{0, 1\}^n$  be an affine domain. Then, the minority aggregator:*

$$\bar{\oplus} = \underbrace{(\oplus, \dots, \oplus)}_{n\text{-times}}$$

*is not a generalized dictatorship for  $D$ .*

*Proof.* Let  $x, y, z \in D$  be three pairwise distinct vectors. Since  $y \neq z$ , there exists a  $j \in \{1, \dots, n\}$  such that  $y_j \neq z_j$ . It follows that  $y_j + z_j \equiv 1 \pmod{2}$ . This means that  $\oplus(x_j, y_j, z_j) \neq x_j$  and thus that  $\bar{\oplus}(x, y, z) \neq x$ . In the same way we show that  $\bar{\oplus}(x, y, z) \notin \{x, y, z\}$ , which is a contradiction, since  $\bar{\oplus}$  is an aggregator for  $D$ .  $\square$

Recall that the only binary unanimous functions are  $\wedge, \vee, \text{pr}_1^2, \text{pr}_2^2$ .

**Claim 7.2.7.** *Suppose  $D \subseteq \{0, 1\}^n$  admits a binary non-dictatorial non-symmetric aggregator  $F = (f_1, \dots, f_n)$ . Then  $F$  is not a generalized dictatorship.*

*Proof.* Assume, to obtain a contradiction, that  $F$  is a generalized dictatorship for  $D$  and let  $x, y \in D$ . Then,  $F(x, y) := z \in \{x, y\}$ . Assume that  $z = x$ . The case where  $z = y$  is analogous.

Let  $J \subseteq \{1, \dots, n\}$  such that  $f_j$  is symmetric, for all  $j \in J$  and  $f_j$  is a projection otherwise. Note that  $J \neq \{1, \dots, n\}$ . Let also  $I \subseteq \{1, \dots, n\} \setminus J$ , such that  $f_i = \text{pr}_2^2$ , for all  $i \in I$  and  $f_i = \text{pr}_1^2$  otherwise. If  $I \neq \emptyset$ , then, for all  $i \in I$ , it holds that:

$$y_i = \text{pr}_2^2(x_i, y_i) = f_i(x_i, y_i) = z_i = x_i.$$

Since  $x, y$  were arbitrary, it follows that  $D_i = \{x_i\}$ , for all  $i \in I$ . Contradiction, since  $D$  is non-degenerate.

If  $I = \emptyset$ , then  $f_j = \text{pr}_1^2$ , for all  $j \notin J$ . Note that in that case,  $J \neq \emptyset$ , lest  $F$  is dictatorial. Now, consider  $F(y, x) := w \in \{x, y\}$  since  $F$  is a generalized dictatorship. By the definition of  $F$ ,  $w_j = z_j = x_j$ , for all  $j \in J$ , and  $w_i = y_i$ , for all  $i \notin J$ . Thus, if  $w = x$ ,  $D$  is degenerate on  $\{1, \dots, n\} \setminus J$ , whereas if  $w = y$ ,  $D$  is degenerate on  $J$ . In both cases, we obtain a contradiction.  $\square$

The only case left is when  $D \subseteq \{0, 1\}^n$  admits a binary symmetric aggregator. Contrary to the previous case, where we showed that the respective non-dictatorial aggregators could not be generalized dictatorships, here we cannot argue this way, as Example 7.2.1 attests. Interestingly enough, we show that as in Example 7.2.1, we can always find some symmetric aggregator for such a domain that is not a generalized dictatorship.

**Claim 7.2.8.** *Suppose  $D \subseteq \{0, 1\}^n$  admits a binary non-dictatorial symmetric aggregator  $F = (f_1, \dots, f_n)$ . Then, there is a binary symmetric aggregator  $G = (g_1, \dots, g_n)$  for  $D$  ( $G$  can be different from  $F$ ) that is not a generalized dictatorship for  $D$ .*

*Proof.* If  $F$  is not a generalized dictatorship for  $D$ , we have nothing to prove. Suppose it is and let  $J \subseteq \{1, \dots, n\}$ , such that  $f_j = \vee$ , for all  $j \in J$  and  $f_i = \wedge$  for all  $i \notin J$  ( $J$  can be both empty or  $\{1, \dots, n\}$ ).

Let  $D^* = \{d^* = (d_1^*, \dots, d_n^*) \mid d = (d_1, \dots, d_n) \in D\}$ , where:

$$d_j^* = \begin{cases} 1 - d_j & \text{if } j \in J \\ d_j & \text{else.} \end{cases}$$

By Lemma 7.2.3,  $H = (h_1, \dots, h_n)$  is a symmetric aggregator for  $D$  if and only if  $H^* = (h_1^*, \dots, h_n^*)$  is an aggregator for  $D^*$ , where  $h_j^* = h_j$ , for all  $j \notin J$  and, for all  $j \in J$ , if  $h_j = \vee$ , then  $h_j^* = \wedge$  and vice-versa. As expected, the property of being a generalized dictatorship carries on this transformation.

**Claim 7.2.9.**  *$H$  is a generalized dictatorship for  $D$  if and only if  $H^*$  is a generalized dictatorship for  $D^*$ .*

*Proof.* Let  $x = (x_1, \dots, x_n)$ ,  $y = (y_1, \dots, y_n) \in D$  and  $z := H(x, y)$ . Since  $\vee(x_j, y_j) = 1 - \wedge(1 - x_j, 1 - y_j)$  and  $\wedge(x_j, y_j) = 1 - \vee(1 - x_j, 1 - y_j)$ , it holds that  $z_j = h_j^*(x_j^*, y_j^*)$ , for all  $j \notin J$ , and  $1 - z_j = h_j^*(x_j^*, y_j^*)$ , for all  $j \in J$ . Thus,  $z^* = H^*(x^*, y^*)$ . It follows that  $z \in \{x, y\}$  if and only if  $z^* \in \{x^*, y^*\}$ .  $\square$

Now, since  $D$  admits the generalized dictatorship  $F$ , it follows that  $D^*$  admits the binary aggregator  $\bar{\wedge} = (\underbrace{\wedge, \dots, \wedge}_{n\text{-times}})$ , that is also a generalized dic-

tatorship. Our aim is to show that  $D^*$  admits a symmetric aggregator that is not a generalized dictatorship. The result will then follow by Claim 7.2.9.

For two elements  $x^*, y^* \in D^*$ , we write  $x^* \leq y^*$  if, for all  $j \in \{1, \dots, n\}$  such that  $x_j^* = 1$ , it holds that  $y_j^* = 1$ .

**Claim 7.2.10.**  $\leq$  is a total ordering for  $D^*$ .

*Proof.* To obtain a contradiction, let  $x^*, y^* \in D^*$  such that neither  $x^* \leq y^*$  nor  $y^* \leq x^*$ . Thus, there exist  $i, j \in \{1, \dots, n\}$ , such that  $x_i^* = 1$ ,  $y_i^* = 0$ ,  $x_j^* = 0$  and  $y_j^* = 1$ . Thus:

$$\wedge(x_i^*, y_i^*) = \wedge(x_j^*, y_j^*) = 0.$$

Then,  $\bar{\wedge}(x^*, y^*) \notin \{x^*, y^*\}$ . Contradiction, since  $\bar{\wedge}$  is a generalized dictatorship.  $\square$

Thus, we can write  $D^* = \{d^1, \dots, d^N\}$ , where  $d^s \leq d^t$  if and only if  $s \leq t$ . Let  $I \subseteq \{1, \dots, n\}$  be such that, for all  $j \in I$ :  $d_j^s = 0$  for  $s = 1, \dots, N-1$ , and  $d_j^N = 1$ . Observe that  $I$  cannot be empty, lest  $d^N = d^{N-1}$  and that  $I \neq \{1, \dots, n\}$ , since  $|D| \geq 3$ . Let now  $G = (g_1, \dots, g_n)$  such that  $g_j = \wedge$ , for all  $j \in I$  and  $g_j = \vee$ , for all  $j \notin I$ .

$G$  is an aggregator for  $D^*$ . Indeed, let  $d^s, d^t \in D^*$  with  $s \leq t \leq N-1$ . Then, for all  $j \notin I$ :

$$g_j(d_j^s, d_j^t) = \vee(d_j^s, d_j^t) = d_j^t.$$

Also for all  $j \in I$ :

$$g_j(d_j^s, d_j^t) = \wedge(d_j^s, d_j^t) = 0 = d_j^t.$$

Thus,  $G(d^s, d^t) = d^t \in D^*$ . Finally, consider  $G(d^s, d^N)$ . Again,  $g_j(d_j^s, d_j^N) = \wedge(d_j^s, d_j^N) = 0$  for all  $j \in I$  and  $g_j(d_j^s, d_j^N) = \vee(d_j^s, d_j^N) = d_j^N$ , for all  $j \notin I$ . By definition of  $I$ ,  $G(d^s, d^N) = d^{N-1} \in D^*$ . This, last point shows also that  $G$  is not a generalized dictatorship, since, for any  $s \neq N-1$ ,  $d^{N-1} \notin \{d^s, d^N\}$ .  $\square$

This completes the proof of Theorem [7.2.9](#).  $\square$

By Theorems [7.2.5](#) and [7.2.9](#) we obtain the following result.

**Corollary 7.2.2.** *A domain  $D \subseteq \{0, 1\}^n$ , with at least three elements, admits an aggregator that is not a generalized dictatorship if and only if there exists a possibility integrity constraint whose set of models equals  $D$ .*

**Remark 7.2.3.** *What about knowing if a possibility integrity constraint really describes a domain that admits an aggregator that is not a generalized dictatorship? For the requirement of having a non-degenerate domain, the situation is the same as in Remark [7.2.1](#). For the requirement of the domain having at least three elements, given that it is non-degenerate, it is easy to see*

that such domains can only arise as the truth sets of possibility integrity constraints that are Horn, renamable Horn or affine. In all these cases, Creignou and Hébrard [57] have devised polynomial-delay algorithms that generate all the solutions of such formulas, which can easily be implemented to terminate if they find more than two solutions.

**Anonymous, Monotone and StrongDem Aggregators** Our final results concern three kinds of non-dictatorial aggregators, whose properties are based on the majority aggregator. We repeat their definitions here.

**Definition 7.2.2.** Let  $D \subseteq \{0, 1\}^n$ . A  $k$ -ary aggregator  $F = (f_1, \dots, f_n)$  for  $D$  is:

1. *Anonymous*, if it holds that for all  $j \in \{1, \dots, n\}$  and for any permutation  $p : \{1, \dots, k\} \mapsto \{1, \dots, k\}$ :

$$f_j(a_1, \dots, a_k) = f_j(a_{p(1)}, \dots, a_{p(k)}),$$

for all  $a_1, \dots, a_k \in \{0, 1\}$ .

2. *Monotone*, if for all  $j \in \{1, \dots, n\}$  and for all  $i \in \{1, \dots, k\}$ :

$$f_j(a_1, \dots, a_{i-1}, 0, a_{i+1}, \dots, a_k) = 1 \Rightarrow f_j(a_1, \dots, a_{i-1}, 1, a_{i+1}, \dots, a_k) = 1.$$

3. *StrongDem*, if it holds that for all  $j \in \{1, \dots, n\}$  and for all  $i \in \{1, \dots, k\}$ , there exist  $a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_k \in \{0, 1\}$ :

$$f_j(a_1, \dots, a_{i-1}, 0, a_{i+1}, \dots, a_k) = f_j(a_1, \dots, a_{i-1}, 1, a_{i+1}, \dots, a_k).$$

Anonymous aggregators ensure that all the voters are treated equally, while monotone that if more voters agree with the aggregator's result, then the outcome does not change. From a Social Theoretic point of view, Nehring and Puppe [175] have argued that "For Arrowian (i.e. independent) aggregators, monotonicity is extremely natural, and it is hard to see how non-monotone Arrowian aggregators could be of interest in practice." StrongDem aggregators were introduced by Szegedy and Xu [207]. The idea here is that there is a way to fix the votes of any  $k - 1$  voters such that the remaining voter cannot change the outcome of the aggregation. Apart from the interest

these aggregators have for Judgement Aggregation, Szegedy and Xu show that they have strong algebraic properties, as they relate to a property of functions called *strong resilience* (see again [154, 207]).

Notationally, since these properties are defined for each component of an aggregator, we will say that a Boolean function  $f$  is anonymous or monotone if it satisfies property 1 or 2 of Definition 7.2.2 respectively. A Boolean function  $f$  that satisfies property 3 of Definition 7.2.2 has appeared in the bibliography under the name of *1-immune* (see [154]). The first immediate consequence of Definition 7.2.2, is that an anonymous or StrongDem aggregator is non-dictatorial. On the other hand, projection and binary symmetric functions are easily monotone, thus every dictatorial and every binary aggregator is monotone. Furthermore, since projections are neither anonymous nor 1-immune and by Theorem 5.2.1 it is straightforward to observe the following results.

**Corollary 7.2.3.** *Any possibility domain  $D$  either admits a monotone non-dictatorial aggregator or an anonymous one. Furthermore, a domain  $D$  admitting an anonymous or StrongDem aggregator is a local possibility domain.*

Regardless of Corollary 7.2.3, we can find non-dictatorial aggregators that are neither anonymous, nor monotone, nor StrongDem. An easy such example is an aggregator with at least one component being  $\text{pr}_1^3$  and another being  $\oplus$ , since  $\text{pr}_1^3$  is not anonymous,  $\oplus$  is not monotone and neither of the two is 1-immune. Corollary 7.2.3 implies that a domain admitting such an aggregator, admits also another that is monotone or anonymous.

We proceed now with some examples that highlight the various connections between these types of aggregators.

**Example 7.2.2.** *Any renamable Horn or bijunctive formula describes a domain admitting a symmetric or majority aggregator respectively. Such aggregators are anonymous, monotone and StrongDem. For a more complicated example, consider the formula*

$$\phi_{13} = (\neg x_1 \vee x_2) \wedge (x_2 \vee x_3 \vee x_4),$$

whose set of satisfying assignments is the local possibility domain:

$$\text{Mod}(\phi_{13}) = \{0, 1\}^4 \setminus \left( (\{(1, 0)\} \times \{0, 1\}^2) \cup \{(0, 0, 0, 0)\} \right).$$

It is straightforward to check that  $\text{Mod}(\phi_{13})$  admits the anonymous, monotone and StrongDem aggregator  $(\wedge^{(3)}, \vee^{(3)}, \text{maj}, \text{maj})$ .

On the other hand, consider the affine formula

$$\phi_{14} = x_1 \oplus x_2 \oplus x_3,$$

where:

$$\text{Mod}(\phi_{14}) = \{(0, 0, 1), (0, 1, 0), (1, 0, 0), (1, 1, 1)\}.$$

It can be proven (by a combination of results by Dokow and Holzman [76, Example 3] and Kirousis et al. [144, Example 4.5]) that  $\text{Mod}(\phi_{14})$  does not admit any monotone or StrongDem aggregators. On the other hand, it does admit the anonymous aggregator  $\bar{\oplus} = (\oplus, \oplus, \oplus)$ .

Recall that in Example 5.3.6, we argued that the set of satisfying assignments of the formula:  $\phi_7 = (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3)$  is the impossibility domain  $\text{Mod}(\phi_7) = \{0, 1\}^3 \setminus \{(1, 0, 0), (0, 1, 1)\}$ . Consider also, from the same example, the formula  $\phi_9 = (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_4 \vee x_5 \vee x_6) \wedge (x_4 \vee \neg x_5 \vee \neg x_6)$ , whose set of satisfying assignments is:  $\text{Mod}(\phi_9) = \text{Mod}(\phi_7) \times \text{Mod}(\phi_7)$ .  $\text{Mod}(\phi_9)$  is a possibility domain admitting the monotone aggregator  $(\text{pr}_1^2, \text{pr}_1^2, \text{pr}_1^2, \text{pr}_2^2, \text{pr}_2^2, \text{pr}_2^2)$ . On the other hand,  $\text{Mod}(\phi_9)$  admits neither anonymous, nor StrongDem aggregators, as it is not a local possibility domain.  $\diamond$

We now provide examples of StrongDem aggregators that are either not anonymous or not monotone. Domains admitting such aggregators can be proven to also admit aggregators that are anonymous and monotone (see Theorem 7.2.11 below).

**Example 7.2.3.** Let  $F = \bar{f}$ , where  $f$  is a ternary operation defined as follows:

$$\begin{aligned} f(0, 0, 0) &= f(0, 0, 1) = f(0, 1, 1) = f(1, 0, 1) = 0, \\ f(0, 1, 0) &= f(1, 0, 0) = f(1, 1, 0) = f(1, 1, 1) = 1. \end{aligned}$$

Obviously,  $\bar{f}$  is neither anonymous nor monotone, since e.g.  $f(0, 0, 1) \neq f(0, 1, 0)$  and  $f(0, 1, 0) = 1$ , whereas  $f(0, 1, 1) = 0$ . On the other hand,  $\bar{f}$  is StrongDem. Indeed, for each component of  $f$ , it holds that:

$$f(x, 0, 1) = f(0, x, 1) = f(0, 0, x) = 0,$$

for all  $x \in \{0, 1\}$ .



Now, consider  $G = \bar{g}$  where  $g$  is a ternary operation defined as follows:

$$\begin{aligned} g(0,0,0) &= g(0,0,1) = g(0,1,0) = g(1,0,0) = g(1,1,0) = 0, \\ g(0,1,1) &= g(1,0,1) = g(1,1,1) = 1. \end{aligned}$$

Again,  $\bar{g}$  is easily not anonymous, since  $g(1,1,0) \neq g(0,1,1)$ . On the other hand,  $\bar{g}$  is monotone and StrongDem. For the latter, observe that:

$$g(x,0,0) = g(0,x,0) = g(0,0,x) = 0,$$

for all  $x \in \{0,1\}$ . The former is very easy to check and is left to the reader.

Finally, let  $H = \bar{h}$ , where  $h$  is a 4-ary operation defined as follows:

$h(x,y,z,w) = 1$  if and only if exactly two or all of  $x,y,z,w$  are equal to 1.

Since the output of  $h$  does not depend on the positions of the input bits,  $h$  is anonymous. Also,  $h$  is 1-immune, since:

$$h(x,0,0,0) = h(0,y,0,0) = h(0,0,z,0) = h(0,0,0,w),$$

for all  $x,y,z,w \in \{0,1\}$ . On the other hand,  $h$  is not monotone, since  $h(0,0,1,1) = 1$  and  $h(0,1,1,1) = 0$ .  $\diamond$

The only combination of properties from Definition [7.2.2](#) we have not seen, is an anonymous and monotone aggregator that is not Strong Dem. We end this subsection by proving that such aggregators do not exist.

**Lemma 7.2.6.** *Let  $f$  be a  $k$ -ary anonymous and monotone Boolean function. Then,  $f$  is also 1-immune.*

*Proof.* For  $k = 2$ , the only anonymous functions are  $\wedge$  and  $\vee$ , which are also 1-immune.

Let  $k \geq 3$ . Since  $f$  is anonymous and monotone, it is not difficult to observe that there is some  $l \in \{0, \dots, k\}$ , such that the output of  $f$  is 0 if and only if there are at most  $l$  1's in the input bits. If  $l > 0$ , then:

$$f(x,0,0 \dots, 0,0) = f(0,x,0, \dots, 0,0) = \dots = f(0,0,0, \dots, 0,x) = 0,$$

for all  $x \in \{0,1\}$ . If  $l = 0$ , then:

$$f(x,1,1 \dots, 1,1) = f(1,x,1, \dots, 1,1) = \dots = f(1,1,1, \dots, 1,x) = 1,$$

for all  $x \in \{0,1\}$ . In both cases,  $f$  is 1-immune.  $\square$

We proceed with the syntactic characterization of domains admitting anonymous aggregators. Nehring and Puppe [175, Theorem 2] showed that a domain admits a monotone locally non-dictatorial aggregator if and only if it admits a monotone anonymous one. Kirousis et al. [144] strengthened this result by dropping the monotonicity requirement and fixing the arity of the anonymous aggregator, as a direct consequence of Theorem 7.1.4.

**Corollary 7.2.4** (Kirousis et al. [144], Corollary 5.11).  *$D$  is a local possibility domain if and only if it admits a ternary anonymous aggregator.*

*Proof.* Immediate from the fact that any aggregator of the type described in Theorem 7.1.4 is anonymous.  $\square$

Thus, we obtain the following result.

**Corollary 7.2.5.**  *$D$  admits a  $k$ -ary anonymous aggregator if and only if there exists a local possibility integrity constraint whose set of models equals  $D$ .*

We now turn to monotone aggregators. Recall that a  $k$ -ary Boolean operation  $f$  is *linear*, if there exist constants  $c_0, \dots, c_k \in \{0, 1\}$  such that:

$$f(x_1, \dots, x_k) = c_0 \oplus c_1x_1 \oplus \dots \oplus c_kx_k,$$

where  $\oplus$  again denotes binary addition mod 2. We need two facts concerning linear functions.

**Lemma 7.2.7.** *Let  $f : \{0, 1\}^k \mapsto \{0, 1\}$  be a linear function and suppose that  $c_0, c_1, \dots, c_k \in \{0, 1\}$  such that:*

$$f(x_1, \dots, x_k) = c_0 \oplus c_1x_1 \oplus \dots \oplus c_kx_k.$$

*Then,  $f$  is unanimous if and only if  $c_0 = 0$  and there is an odd number pairwise distinct indices  $i \in \{1, \dots, k\}$  such that  $c_i = 1$ .*

*Proof.* The inverse direction is straightforward. For the forward direction, set  $x_1 = \dots = x_k = 0$ . Then,  $f(0, \dots, 0) = c_0$  and thus  $c_0 = 0$  since  $f$  is unanimous. Finally, assume, to obtain a contradiction, that there is an even number of  $c_1, \dots, c_k$  that are equal to 1. Set  $x_1 = \dots = x_k = 1$ . Then, it holds that  $f(1, \dots, 1) = 0$  and  $f$  is not unanimous. Contradiction.  $\square$

Since we work only with unanimous functions, from now on we will assume that a linear function satisfies the conditions of Lemma [7.2.7](#). This implies also that any linear function has odd arity. Note also that any  $k$ -ary linear functions  $f$ , with exactly one  $i \in \{1, \dots, k\}$  such that  $c_i = 1$  is an essentially unary function.

**Lemma 7.2.8.** *Let  $f : \{0, 1\}^k \mapsto \{0, 1\}$  be a linear function,  $k \geq 3$ . Then, either  $f$  is an essentially unary function, or it is neither monotone nor 1-immune.*

*Proof.* Let  $c_1, \dots, c_k \in \{0, 1\}$  such that:

$$f(x_1, \dots, x_k) = c_1x_1 \oplus \dots \oplus c_kx_k$$

and assume it is not an essentially unary function. Then, there exist at least three pairwise distinct indices  $i \in \{1, \dots, k\}$  such that  $c_i = 1$ . If there are exactly three,  $f = \oplus$ , which is easily neither monotone, nor 1-immune.

Assume now that there are at least five pairwise distinct  $i \in \{1, \dots, k\}$  such that  $c_i = 1$ . We will need only four of these indices.

Now, let  $j_1, j_2, j_3, j_4 \in \{1, \dots, k\}$  such that  $c_{j_1}, c_{j_2}, c_{j_3}, c_{j_4} = 1$ . Set  $x_{j_1} = x_{j_2} = x_{j_3} = 1$  and  $x_i = 0$ , for all  $i \in \{1, \dots, k\} \setminus \{j_1, j_2, j_3\}$ . Then,  $f(x_1, \dots, x_k) = 1$ . By letting  $x_{j_4} = 1$  too, we obtain  $f(x_1, \dots, x_k) = 0$ , which shows that  $f$  is not monotone.

Finally, to obtain a contradiction, suppose  $f$  is 1-immune. Then, there exist  $d_2, \dots, d_k \in \{0, 1\}$  such that:

$$\begin{aligned} f(0, d_2, \dots, d_k) = (1, d_2, \dots, d_k) &\Leftrightarrow \\ c_2d_2 \oplus \dots \oplus c_kd_k = c_1 \oplus c_2d_2 \oplus \dots \oplus c_kd_k &\Leftrightarrow \\ c_1 = 0. & \end{aligned}$$

Continuing in the same way, we can prove that  $c_j = 0$ , for  $j = 1, \dots, k$ , which is a contradiction.  $\square$

We are now ready to prove our results concerning monotone and Strong-Dem aggregators.

**Theorem 7.2.10.** *A domain  $D \subseteq \{0, 1\}^n$  admits a monotone non-dictatorial aggregator of some arity if and only if it admits a binary non-dictatorial one.*

*Proof.* That a domain admitting a binary non-dictatorial aggregator, admits also a non-dictatorial monotone one is obvious, since all binary unanimous functions are monotone.

For the forward direction, since  $D$  admits a monotone non-dictatorial aggregator, it is a possibility domain. Now, to obtain a contradiction, suppose  $D$  does not admit a binary non-dictatorial aggregator. Kirousis et al. [144, Lemma 3.4] showed that in this case, every  $k$ -ary aggregator,  $k \geq 2$  for  $D$  is systematic (the notion of *local monomorphicity* they use corresponds to systematicity in the Boolean framework).

Now, since  $D$  contains no binary non-dictatorial aggregators,  $\wedge, \vee \notin \mathcal{C}_j$ , for all  $j \in \{1, \dots, n\}$ . Thus, by Post's lattice, either  $\text{maj}$  or  $\oplus$  are contained in  $\mathcal{C}_j$ , for all  $j \in \{1, \dots, n\}$  (since the aggregators must be systematic), lest each  $\mathcal{C}_j$  contains only projections.

Assume that  $\text{maj}$  is an aggregator for  $D$ . Then, by Kirousis et al. [144, Theorem 3.7],  $D$  admits also a binary non-dictatorial aggregator. Contradiction.

Thus, we also have that  $\text{maj} \notin \mathcal{C}_j$ ,  $j = 1, \dots, n$ . It follows that only  $\oplus \in \mathcal{C}_j$ ,  $j = 1, \dots, n$ . By Post [186], it follows that for all  $j \in \{1, \dots, n\}$ ,  $\mathcal{C}_j$  contains only linear functions (see also [30]). By Lemma [7.2.8], we obtain a contradiction.  $\square$

Thus, by Proposition [7.2.3] and Theorem [7.2.4], we obtain the following syntactic characterization.

**Corollary 7.2.6.**  *$D$  admits a  $k$ -ary non-dictatorial monotone aggregator if and only if there exists a separable or renamable partially Horn integrity constraint whose set of models equals  $D$ .*

To end this subsection, we now consider StrongDem aggregators. We employ the “diamond” operator  $\diamond$ , in order to combine ternary aggregators to obtain new ones whose components are commutative functions (recall Subsec. [7.1.3]). Unfortunately, this operator will not suffice for our purposes, so we will also use a new operator we call the *star* operator  $\star$ .

Let  $F = (f_1, \dots, f_n)$  and  $G = (g_1, \dots, g_n)$  be two  $n$ -tuples of ternary functions. Define  $H := F \star G$  to be the  $n$ -tuple of ternary functions  $H = (h_1, \dots, h_n)$  where:

$$h_j(x, y, z) = f_j(f_j(x, y, z), f_j(x, y, z), g_j(x, y, z)),$$

for all  $x, y, z \in \{0, 1\}$ . Easily, if  $F$  and  $G$  are aggregators for a domain  $D$ , then so is  $H$ , since it is produced by a superposition of  $F$  and  $G$ . Under some assumptions for  $F$  and  $G$ ,  $H$  has no components equal to  $\oplus$ .

**Lemma 7.2.9.** *Let  $F = (f_1, \dots, f_n)$  be an  $n$ -tuple of ternary functions, such that  $f_j \in \{\wedge^{(3)}, \vee^{(3)}, \text{maj}, \oplus\}$ ,  $j = 1, \dots, n$ , and let  $J = \{j \mid f_j = \oplus\}$ . Let also  $G = (g_1, \dots, g_n)$  be an  $n$ -tuple of ternary functions, such that  $g_j \in \{\wedge^{(3)}, \vee^{(3)}, \text{maj}\}$ , for all  $j \in J$ . Then, for the  $n$ -tuple of ternary functions  $F \star G := H = (h_1, \dots, h_n)$ , it holds that:*

$$h_j \in \{\wedge^{(3)}, \vee^{(3)}, \text{maj}\},$$

for  $j = 1, \dots, n$ .

*Proof.* First, let  $j \in \{1, \dots, n\} \setminus J$ . Then,  $f_j \in \{\wedge^{(3)}, \vee^{(3)}, \text{maj}\}$  and let  $x, y, z \in \{0, 1\}$  that are not all equal (lest we have nothing to show since all  $f_j, g_j$  are unanimous). If  $f_j = \wedge^{(3)}$ , then easily:

$$\begin{aligned} h_j(x, y, z) &= \wedge^{(3)}(\wedge^{(3)}(x, y, z), \wedge^{(3)}(x, y, z), g_j(x, y, z)) = \\ &= \wedge^{(3)}(0, 0, g_j(x, y, z)) = 0, \end{aligned}$$

which shows that  $h_j = \wedge^{(3)}$ . Analogously, we show that  $f_j = \vee^{(3)}$  implies that  $h_j = \vee^{(3)}$ . Finally, let  $f_j = \text{maj}$  and let  $\text{maj}(x, y, z) := z \in \{0, 1\}$ . Then:

$$\begin{aligned} h_j(x, y, z) &= \text{maj}(\text{maj}(x, y, z), \text{maj}(x, y, z), g_j(x, y, z)) = \\ &= \text{maj}(z, z, g_j(x, y, z)) = z, \end{aligned}$$

which shows that  $h_j = \text{maj}$ .

Thus, we can now assume that  $J \neq \emptyset$ . Let  $j \in J$ . Then, we have that  $f_j = \oplus$  and  $g_j \in \{\wedge^{(3)}, \vee^{(3)}, \text{maj}\}$ . Thus, we have that:

$$h_j(x, y, z) = \oplus(\oplus(x, y, z), \oplus(x, y, z), g_j(x, y, z)) = g_j(x, y, z),$$

from which it follows that  $h_j \in \{\wedge^{(3)}, \vee^{(3)}, \text{maj}\}$ . □

At last, we are ready to prove our final results.

**Theorem 7.2.11.** *A Boolean domain  $D \subseteq \{0, 1\}^n$  admits a  $k$ -ary Strong-Dem aggregator if and only if it admits a ternary aggregator  $F = (f_1, \dots, f_n)$  such that  $f_j \in \{\wedge^{(3)}, \vee^{(3)}, \text{maj}\}$ , for  $j = 1, \dots, n$ .*

*Proof.* It is very easy to see that all the functions in  $\{\wedge^{(3)}, \vee, \text{maj}\}$  are 1-immune. Thus, we only need to prove the forward direction of the theorem.

To that end, let  $F = (f_1, \dots, f_n)$  be a  $k$ -ary StrongDem aggregator for  $D$ . Then, by Theorem 7.1.4, there exists a ternary aggregator  $G = (g_1, \dots, g_n)$  such that  $g_j \in \{\wedge^{(3)}, \vee^{(3)}, \text{maj}, \oplus\}$  for  $j = 1, \dots, n$ . Let  $J = \{j \mid f_j = \oplus\}$ . If  $J = \emptyset$ , then we have nothing to prove. Otherwise, consider the clones  $\mathcal{C}_j$ , for each  $j \in J$ .

Suppose now that there exists a  $j \in J$ , such that  $\mathcal{C}_j$  contains neither  $\wedge$ , nor  $\vee$ , nor  $\text{maj}$ . By Post's classification of clones of Boolean functions (see [30, 186]) and since  $\mathcal{C}_j$  contains  $\oplus$  and only unanimous functions,  $\mathcal{C}_j$  contains only linear unanimous functions. Again, by Lemma 7.2.8,  $\mathcal{C}_j$  does not contain any 1-immune function. Contradiction.

Thus, for each  $j \in J$ , it holds that  $\mathcal{C}_j$  contains either  $\wedge$ , or  $\vee$  or  $\text{maj}$ . In the first two cases,  $\mathcal{C}_j$  obviously contains  $\wedge^{(3)}$  or  $\vee^{(3)}$  too respectively. Then, it holds that for each  $j \in J$  there exists an aggregator  $H^j = (h_1^j, \dots, h_n^j)$ , such that  $h_j^j \in \{\wedge^{(3)}, \vee^{(3)}, \text{maj}\}$ . Let  $J := \{j_1, \dots, j_t\}$ .

We will now perform a series of iterative combinations between  $G$  and the various  $H^j$ 's, using the  $\diamond$  and  $\star$  operators, in order to obtain the required aggregator.

First, let  $G^j = G \diamond H^j$ , for all  $j \in J$ . By Kirousis et al. [144, Lemma 5.10], we have that

$$G_i^j \in \{\wedge^{(3)}, \vee^{(3)}, \text{maj}, \oplus\},$$

for all  $i \in \{1, \dots, n\}$  and  $j \in J$ . Furthermore,

$$G_{j_s}^{j_s} \in \{\wedge^{(3)}, \vee^{(3)}, \text{maj}\},$$

for  $s = 1, \dots, t$ . Thus for the aggregator:

$$G^* := (\dots ((G \star G^{j_1}) \star G^{j_2}) \star \dots \star G^{j_t}),$$

we have, by Lemma 7.2.9:

$$G_j^* \in \{\wedge^{(3)}, \vee^{(3)}, \text{maj}\},$$

for  $j = 1, \dots, n$ , which concludes the proof.  $\square$

Recall Definition 5.3.6. We say that a local possibility integrity constraint is  $\oplus$ -free, if  $V_2 = \emptyset$ . Thus, we obtain the following syntactic characterization.

**Corollary 7.2.7.** *A Boolean domain  $D \subseteq \{0, 1\}^n$  admits a  $k$ -ary StrongDem aggregator if and only if there exists an  $\oplus$ -free local possibility integrity constraint whose set of satisfying assignments equals  $D$ .*

**Systematic Aggregators** We end this work with a discussion concerning systematic aggregators. This is a natural requirement for aggregators from a Social Choice point of view, given that the issues that need to be decided are of the same nature. Recall that  $F = (f_1, \dots, f_n)$  is systematic if  $f_1 = f_2 = \dots = f_n$ . The following is obvious by considering the definitions of an aggregator and a polymorphism.

**Lemma 7.2.10.** *Let  $D \subseteq \{0, 1\}^n$  be a Boolean domain and  $F = \bar{f}$  a systematic  $n$ -tuple of  $k$ -ary Boolean functions. Then  $F$  is an aggregator for  $D$  if and only if  $f$  is a polymorphism for  $D$ .*

This directly implies that domains admitting non-dictatorial systematic aggregators are either Horn, dual-Horn, bijunctive or affine. We thus immediately obtain the following characterization.

**Corollary 7.2.8.** *A Boolean domain  $D \subseteq \{0, 1\}^n$  admits a  $k$ -ary non-dictatorial systematic aggregator if and only if there exists an integrity constraint which is either Horn, dual Horn, bijunctive or affine, whose set of satisfying assignments equals  $D$ .*

**Remark 7.2.4.** *Why does  $\overline{\text{maj}}$  appears here, although it did not in the characterization of possibility domains (Theorem 5.2.1)? In the Boolean case, a domain admitting  $\overline{\text{maj}}$ , also admits a binary aggregator  $F = (f_1, \dots, f_n)$ , such that  $f_j \in \{\wedge, \vee\}$ ,  $j = 1, \dots, n$  (see Kirousis et al. [144, Theorem 3.7]). The problem is that this aggregator need not be systematic. In fact, the proof of the aforementioned theorem would produce a systematic aggregator only if  $(0, \dots, 0)$  or  $(1, \dots, 1) \in D$ .*

Now, what if want to characterize domains admitting some of the various non-dictatorial aggregators we discussed, but requiring also that these aggregators satisfy systematicity? By Theorem 7.2.9, we know that Corollary 7.2.8 works for domains admitting systematic aggregators that are not generalized dictatorships too. Furthermore, all the aggregators (resp. integrity constraints) of Corollary 7.3.12 (resp. 7.2.8) are locally non-dictatorial and anonymous aggregators (resp. lpic's), thus we also have characterizations for domains admitting systematic locally non-dictatorial or anonymous aggregators.

For domains admitting monotone or StrongDem systematic aggregators, we will obtain the result by Lemma 7.2.8 and Post's Lattice. We will again use the terminology of polymorphisms.

**Corollary 7.2.9.** *A domain  $D \subseteq \{0, 1\}^n$  admits a  $k$ -ary systematic non-dictatorial monotone or StrongDem aggregator if and only if it is closed under  $\wedge$ ,  $\vee$  or maj.*

*Proof.* It is known (and straightforward to see) that the set of polymorphisms of a domain is a clone. Let  $\mathcal{C}$  be the Boolean clone of polymorphisms of  $D$ . Since it admits a non-dictatorial aggregator, at least one operator from  $\wedge, \vee, \text{maj}, \oplus$  is in  $\mathcal{C}$ . By Lemma 7.2.8, this cannot be only  $\oplus$ .  $\square$

Thus, finally, we have the following result.

**Corollary 7.2.10.** *A Boolean domain  $D \subseteq \{0, 1\}^n$  admits a  $k$ -ary systematic non-dictatorial monotone or StrongDem aggregator if and only if there exists an integrity constraint which is either Horn, dual Horn or biconjunctive, whose set of satisfying assignments equals  $D$ .*

## 7.3 The Computational Complexity of Aggregation

In this section, we turn our attention in the computational complexity of deciding if a domain admits an aggregator from a specific class of non-dictatorial aggregators. We begin with the case where the domain is given explicitly as a set of vectors over  $\mathcal{D}$ . Note that in case  $\mathcal{D}$  is Boolean, we have already provided such algorithms in Subsec. 7.2.3. We begin by showing that there is a polynomial-time algorithm that decides whether a set of feasible voting patterns is a possibility domain (Subsec. 7.3.1). We then proceed to show that this problem is expressible in Transitive Closure Logic (Subsec. 7.3.2). Then, in subsection 7.3.3, we provide a polynomial-time algorithm for checking if a domain is a uniform possibility domain. After that, we provide complexity bounds for the same problem, in case the domain is provided implicitly, via an agenda or an integrity constraint (Subsec. 7.3.4). Finally, we extend these results to other types of non-dictatorial aggregation that have been used in the bibliography (Subsec. 7.3.5).

### 7.3.1 Tractability of Possibility Domains

Theorems 5.2.2 and 7.1.1 provide necessary and sufficient conditions for a set  $X$  to be a possibility domain in the Boolean framework and in the non-



Boolean framework, respectively. Admitting a binary non-dictatorial aggregator is a condition that appears in both of these characterizations. Our first result asserts that this condition can be checked in polynomial time.

**Theorem 7.3.1.** *There is a polynomial-time algorithm for solving the following problem: given a set  $X$  of feasible evaluations, determine whether or not  $X$  admits a binary non-dictatorial aggregator and, if it does, produce one.*

We first show that the existence of a binary non-dictatorial aggregator on  $X$  is tightly related to connectivity properties of a certain directed graph  $H_X$  defined next. If  $X \subseteq \mathcal{D}^n$  is a set of feasible evaluations, then  $H_X$  is the following directed graph:

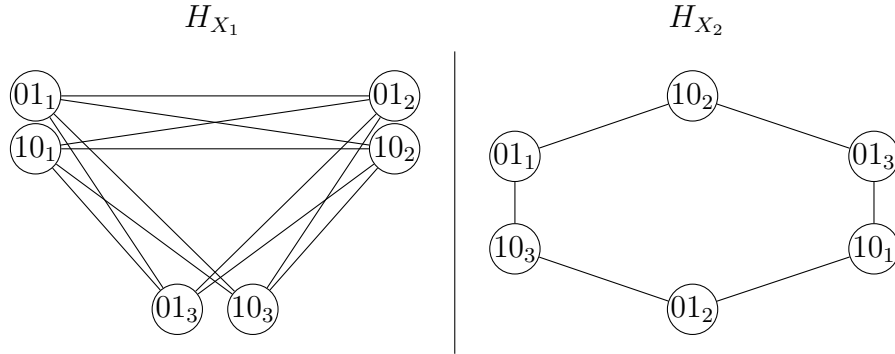
- The vertices of  $H_X$  are the pairs of *distinct* elements  $u, u' \in X_j$ , for  $j \in \{1, \dots, n\}$ . Each such vertex will usually be denoted by  $uu'_j$ . When the coordinate  $j$  is understood from the context, we will often be dropping the subscript  $j$ , thus denoting such a vertex by  $uu'$ .

Also, if  $u \in X_j$ , for some  $j \in \{1, \dots, n\}$ , we will often use the notation  $u_j$  to indicate that  $u$  is an element of  $X_j$ .

- Two vertices  $uu'_s$  and  $vv'_t$ , where  $s \neq t$ , are connected by a directed edge from  $uu'_s$  to  $vv'_t$ , denoted by  $uu'_s \rightarrow vv'_t$ , if there are a total evaluation  $\mathbf{z} \in X$  that extends the partial evaluation  $(u_s, v_t)$  and a total evaluation  $\mathbf{z}' \in X$  that extends the partial evaluation  $(u'_s, v'_t)$ , such that there is no total evaluation  $\mathbf{y} \in X$  that extends  $(u_s, v'_t)$ , and has the property that  $y_i = z_i$  or  $y_i = z'_i$ , for every  $i \in \{1, \dots, n\}$ .

For vertices  $uu'_s, vv'_t$ , corresponding to issues  $s, t$  (that need not be distinct), we write  $uu'_s \rightarrow\rightarrow vv'_t$  to denote the existence of a directed path from  $uu'_s$  to  $vv'_t$ . In the next example, we describe explicitly the graph  $H_X$  for several different sets  $X$  of feasible voting patterns. Recall that a *directed* graph  $G$  is *strongly connected* if for every pair of vertices  $(u, v)$  of  $G$ , there is a (directed) path from  $u$  to  $v$ .

**Example 7.3.1.** *Let  $X_1 = \{(0, 0, 1), (0, 1, 0), (1, 0, 0), (1, 1, 1)\}$  and  $X_2 = \{(0, 0, 1), (0, 1, 0), (1, 0, 0)\}$ . Both  $H_{X_1}$  and  $H_{X_2}$  have six vertices, namely  $01_j$  and  $10_j$ , for  $j = 1, 2, 3$ . In the figures below, we use undirected edges between two vertices  $uu'_s$  and  $vv'_t$  to denote the existence of both  $uu'_s \rightarrow vv'_t$  and  $vv'_t \rightarrow uu'_s$ .*

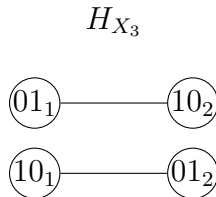


Consider  $01_1, 01_2$  of  $H_{X_1}$ . Since the partial vectors  $(0, 0)$  and  $(1, 1)$  extend to  $(0, 0, 1)$  and  $(1, 1, 1)$ , respectively, we need to check if there is a vector in  $X_1$  extending  $(0, 1)$ , but whose third coordinate is 1. Since  $(0, 1, 1) \notin X_2$ , we have that  $H_{X_1}$  contains both edges  $01_1 \rightarrow 01_2$  and  $01_2 \rightarrow 01_1$ . Now, since the partial vectors  $(0, 1)$  and  $(1, 0)$  extend to  $(0, 1, 0)$  and  $(1, 0, 0)$ , respectively, and since neither  $(0, 0, 0)$  nor  $(1, 1, 0)$  are in  $X_1$ , we have that  $01_1 \leftrightarrow 10_2$ . By the above and because of the symmetric structure of  $X_1$ , it is easy to see that every two vertices  $uv'_i$  and  $vv'_j$  of  $H_{X_1}$  are connected if and only if  $i \neq j$ .

For  $X_2$ , observe that, since no partial vector containing two “1”s, in any two positions, extends to an element of  $X_2$ , there are no edges between the vertices  $01_i, 01_j$  and  $10_i, 10_j$ , for any  $i, j \in \{1, 2, 3\}$ ,  $i \neq j$ . In the same way as with  $H_{X_1}$ , we get that  $H_{X_2}$  is a cycle.

There are two observations to be made, concerning  $H_{X_1}$  and  $H_{X_2}$ . First, they are both strongly connected graphs. Also, neither  $X_1$  nor  $X_2$  admit binary non-dictatorial aggregators ( $X_1$  admits only a minority aggregator and  $X_2$  is an impossibility domain).

Finally, consider  $X_3 := \{(0, 1), (1, 0)\}$ . The graph  $H_{X_3}$  has four vertices,  $01_1, 10_1, 01_2$  and  $10_2$ , and it is easy to see that  $H_{X_3}$  has only the following edges:



Observe that  $X_3$  is not strongly connected (it is not even connected) and that, in contrast to the sets  $X_1$  and  $X_2$ , the set  $X_3$  admits two binary non-dictatorial aggregators, namely,  $(\wedge, \vee)$  and  $(\vee, \wedge)$ . In Lemma [7.3.2](#), we establish a tight connection between strong connectedness and the existence of binary non-dictatorial aggregators.

We now state and prove two lemmas about the graph  $H_X$ .

**Lemma 7.3.1.** *Assume that  $F = (f_1, \dots, f_n)$  is a binary aggregator on  $X$ .*

1. *If  $uu'_s \rightarrow vv'_t$  and  $f_s(u, u') = u$ , then  $f_t(v, v') = v$ .*
2. *If  $uu'_s \rightarrow\rightarrow vv'_t$  and  $f_s(u, u') = u$ , then  $f_t(v, v') = v$ .*

*Proof.* The first part of the lemma follows from the definitions and the fact that  $F$  is conservative. Indeed, if  $uu'_s \rightarrow vv'_t$ , then there are a total evaluation  $z = (z_1, \dots, z_n) \in X$  that extends  $(u_s, v_t)$  (i.e.,  $z_s = u$  and  $z_t = v$ ) and a total evaluation  $z' = (z'_1, \dots, z'_n) \in X$  that extends  $(u'_s, v'_t)$  (i.e.,  $z'_s = u'$  and  $z'_t = v'$ ), such that there is no total evaluation in  $X$  that extends  $(u_s, v'_t)$  and agrees with  $z$  or with  $z'$  on every coordinate. Consider the total evaluation  $(f_1(z_1, z'_1), \dots, f_n(z_n, z'_n))$ , which is in  $X$  because  $F$  is an aggregator on  $X$ . Since each  $f_j$  is conservative, we must have that  $f_j(z_j, z'_j) \in \{z_j, z'_j\}$ , for every  $j$ , hence  $f_t(z_t, z'_t) = f_t(v, v') \in \{v, v'\}$ . Consequently, if  $f_s(u, u') = u$ , then we must have  $f_t(v, v') = v$ , else  $(f_1(z_1, z'_1), \dots, f_n(z_n, z'_n))$  extends  $(u_s, v'_t)$  and agrees with  $z$  or with  $z'$  on every coordinate. The second part of the lemma easily follows from the first part by induction.  $\square$

**Lemma 7.3.2.**  *$X$  admits a binary non-dictatorial aggregator if and only if the directed graph  $H_X$  is not strongly connected.*

Before delving into the proof, consider the graphs of Example [7.3.1](#). Using the fact that the graphs  $H_{X_1}$  and  $H_{X_2}$  are strongly connected and also using the second item of Lemma [7.3.1](#), it is easy to see that  $X_1$  and  $X_2$  admit no binary non-dictatorial aggregator; indeed, let  $F = (f_1, f_2, f_3)$  be a binary aggregator of either of these two sets and suppose that  $f_1(0, 1) = 0$ . Since in both graphs  $H_{X_1}$  and  $H_{X_2}$ , there are paths from  $01_1$  to every other vertex, it follows that  $f_j = pr_1^2$ ,  $j = 1, 2, 3$ . If  $f_1(0, 1) = 1$ , we get that  $f_j = pr_2^2$ ,  $j = 1, 2, 3$ , in the same way.

In contrast, consider  $H_{X_3}$  and let  $G = (g_1, g_2)$  be a pair of binary functions with  $g_1(0, 1) = 0$ . For  $G$  to be an aggregator, Lemma [7.3.1](#) forces us to set

$g_2(1, 0) = 1$ . But now, by setting  $g_1(1, 0) = 0$ , and thus  $g_2(0, 1) = 1$ , we get that  $(g_1, g_2) = (\wedge, \vee)$  is a non-dictatorial binary aggregator for  $X_3$ .

*Proof of Lemma 7.3.2* We first show that if  $X$  admits a binary non-dictatorial aggregator, then  $H_X$  is not strongly connected. In the contrapositive form, we show that if  $H_X$  is strongly connected, then  $X$  admits no binary non-dictatorial aggregator. This is an easy consequence of the preceding Lemma 7.3.1. Indeed, assume that  $H_X$  is strongly connected and let  $F = (f_1, \dots, f_n)$  be a binary aggregator on  $X$ . Take two distinct elements  $x$  and  $x'$  of  $X_1$ . Since  $F$  is conservative, we have that  $f_1(x, x') \in \{x, x'\}$ . Assume first that  $f_1(x, x') = x$ . We claim that  $f_j = \text{pr}_1^2$ , for every  $j \in \{1, \dots, n\}$ . To see this, let  $y$  and  $y'$  be two distinct elements of  $X_j$ , for some  $j \in \{1, \dots, n\}$ . Since  $H_X$  is strongly connected, we have that  $xx'_1 \rightarrow\rightarrow yy'_j$ . Since also  $f_1(x, x') = x$ , Lemma 7.3.1 implies that  $f_j(y, y') = y = \text{pr}_1^2(y, y')$  and so  $f_j = \text{pr}_1^2$ . Next, assume that  $f_1(x, x') = x'$ . We claim that  $f_j = \text{pr}_2^2$ , for every  $j \in \{1, \dots, n\}$ . To see this, let  $y$  and  $y'$  be two distinct elements of  $X_j$ , for some  $j \in \{1, \dots, n\}$ . Since  $H_X$  is strongly connected, we have that  $yy'_j \rightarrow\rightarrow xx'_1$ , hence, if  $f_j(y, y') = y$ , then, Lemma 7.3.1, implies that  $f_1(x, x') = x$ , which is a contradiction because  $x \neq x'$ . Thus,  $f_j(y, y') = y'$  and so  $f_j = \text{pr}_2^2$ .

For the converse, assume that  $H_X$  is not strongly connected and let  $uu'_s, vv'_t$  be two vertices of  $H_X$  such that there is no path from  $uu'_s$  to  $vv'_t$  in  $H_X$ , i.e., it is not true that  $uu'_s \rightarrow\rightarrow vv'_t$ . Let  $V_1, V_2$  be a partition of the vertex set such that  $uu'_s \in V_1, vv'_t \in V_2$ , and there is no edge from a vertex in  $V_1$  to a vertex in  $V_2$ . We will now define a binary aggregator  $F = (f_1, \dots, f_n)$  and prove that it is non-dictatorial.

Given  $\mathbf{z}, \mathbf{z}' \in X$ , we set  $f_j(z_j, z'_j) = z_j$  if  $zz'_j \in V_1$ , and we set  $f_j(z_j, z'_j) = z'_j$  if  $zz'_j \in V_2$ , for  $j \in \{1, \dots, n\}$ . Since  $uu'_s \in V_1$ , we have that  $f_s \neq \text{pr}_2^2$ ; similarly, since  $vv'_t \in V_2$ , we have that  $f_t \neq \text{pr}_1^2$ . Consequently,  $F$  is not a dictatorial function on  $X$ . Thus, what remains to be proved is that if  $\mathbf{z}, \mathbf{z}' \in X$ , then  $F(\mathbf{z}, \mathbf{z}') \in X$ . For this, we will show that if  $F(\mathbf{z}, \mathbf{z}') \notin X$ , then there is an edge from an element of  $V_1$  to an element of  $V_2$ , which is a contradiction.

Assume that  $\mathbf{q} = F(\mathbf{z}, \mathbf{z}') \notin X$ . Let  $K$  be a minimal subset of  $\{1, \dots, n\}$  such that  $\mathbf{q} \upharpoonright K$  cannot be extended to a total evaluation  $\mathbf{w}$  in  $X$  that agrees with  $\mathbf{z}$  or with  $\mathbf{z}'$  on  $\{1, \dots, n\} \setminus K$  (i.e., if  $j \in \{1, \dots, n\} \setminus K$ , then  $w_j = z_j$  or  $w_j = z'_j$ ). Since  $\mathbf{z}'$  is in  $X$ , it does not extend  $\mathbf{q} \upharpoonright K$ , hence there is a number  $p \in K$  such that  $q_p = f_p(z_p, z'_p) = z_p \neq z'_p$ . It follows that the vertex  $zz'_p$  is in  $V_1$ . Similarly, since  $\mathbf{z}$  is in  $X$ , it does not extend  $\mathbf{q} \upharpoonright K$ , hence there is

a number  $r \in K$  such that  $q_r = f_r(z_r, z'_r) = z'_r \neq z_r$ . It follows that the vertex  $zz'_r$  is in  $V_2$ . Consequently, there is no edge from  $zz'_p$  to  $zz'_r$  in  $H_X$ . We will arrive at a contradiction by showing that  $zz'_p \rightarrow zz'_r$ . Consider the set  $K \setminus \{r\}$ . By the minimality of  $K$ , there is a total evaluation  $\mathbf{w}$  in  $X$  that extends  $\mathbf{q} \upharpoonright K \setminus \{r\}$  and agrees with  $\mathbf{z}$  or with  $\mathbf{z}'$  outside  $K \setminus \{r\}$ . In particular, we have that  $w_p = q_p = z_p$  and  $w_r = z_r$ . Similarly, by considering the set  $K \setminus \{p\}$ , we find that there is a total evaluation  $\mathbf{w}'$  in  $X$  that extends  $\mathbf{q} \upharpoonright K \setminus \{p\}$  and agrees with  $\mathbf{z}$  or with  $\mathbf{z}'$  outside  $K \setminus \{p\}$ . In particular, we have that  $w'_p = z'_p$  and  $w'_r = q_r = z'_r$ . Note that  $\mathbf{w}$  and  $\mathbf{w}'$  agree on  $K \setminus \{p, r\}$ . Since  $\mathbf{q} \upharpoonright K$  does not extend to a total evaluation that agrees with  $\mathbf{z}$  or with  $\mathbf{z}'$  outside  $K$ , we conclude that there is no total evaluation  $\mathbf{y}$  in  $X$  that extends  $(z_p, z'_r)$  and agrees with  $\mathbf{w}$  or with  $\mathbf{w}'$  on every coordinate. Consequently,  $zz'_p \rightarrow zz'_r$ , thus we have arrived at a contradiction.  $\square$

**Proof of Theorem 7.3.1:** Given a set  $X$  of feasible evaluations, the graph  $H_X$  can be constructed in time bounded by a polynomial in the size  $|X|$  of  $X$  (in fact, in time  $O(|X|^5)$ ). There are well-known polynomial-time algorithms for testing if a graph is strongly connected and, in case it is not, producing the *strongly connected components (scc)* of the graph; e.g., Kosaraju's algorithm presented in [195] and Tarjan's algorithm in [211]. Consequently, by Lemma 7.3.2, there is a polynomial-time algorithm for determining whether or not a given set  $X$  admits a binary non-dictatorial aggregator. Moreover, if  $X$  admits such an aggregator, then one can be constructed in polynomial-time from the strongly connected components of  $H_X$  via the construction in the proof of Lemma 7.3.2.  $\square$

The next corollary follows from Theorem 7.3.1 and Theorem 7.1.3.

**Corollary 7.3.1.** *There is a polynomial-time algorithm for the following decision problem: given a set  $X$  of feasible evaluations, is  $X$  totally blocked?*

We now turn to the problem of detecting possibility domains in the non-Boolean framework.

**Theorem 7.3.2.** *There is a polynomial-time algorithm for solving the following problem: given a set  $X$  of feasible evaluations, determine whether or not  $X$  is a possibility domain and, if it is, produce a binary non-dictatorial aggregator, or a ternary majority aggregator or a ternary minority aggregator for  $X$ .*

*Proof.* It is straightforward to check that, by Theorem [7.1.1](#) and Theorem [7.3.1](#), it suffices to show that there is a polynomial-time algorithm that, given  $X$ , detects whether or not  $X$  admits a majority aggregator or a minority aggregator, and, if it does, produces such an aggregator.

Let  $X$  be a set of feasible evaluations, where  $I = \{1, \dots, n\}$  is the set of issues and  $\mathcal{D}$  is the set of the position values. We define the *disjoint union*  $\mathbb{D}$  of the set of position values as:

$$\mathbb{D} = \bigsqcup_{j=1}^n \mathcal{D} = \bigcup_{j=1}^n \{(x, j) \mid x \in \mathcal{D}\}.$$

We also set

$$\tilde{X} = \{((x_1, 1), \dots, (x_n, n)) \mid (x_1, \dots, x_n) \in X\} \subseteq \mathbb{D}^n.$$

We will show that we can go back-and-forth between conservative majority or minority polymorphisms for  $\tilde{X}$  and majority or minority aggregators for  $X$ .

Let  $f : \mathbb{D}^k \rightarrow \mathbb{D}$  be a conservative polymorphism for  $\tilde{X}$ . We define the  $n$ -tuple  $F = (f_1, \dots, f_n)$  of  $k$ -ary functions  $f_1, \dots, f_n$  as follows: if  $x_j^1, \dots, x_j^k \in X_j$ , for  $j \in \{1, \dots, n\}$ , then we set  $f_j(x_j^1, \dots, x_j^k) = y_j$ , where  $y_j$  is such that  $f((x_j^1, j), \dots, (x_j^k, j)) = (y_j, j)$ . Such a  $y_j$  exists and is one of the  $x_j^i$ 's because  $f$  is conservative, and hence  $f((x_j^1, j), \dots, (x_j^k, j)) \in \{(x_j^1, j), \dots, (x_j^k, j)\}$ . It is easy to see that  $F$  is an aggregator for  $X$ . Moreover, if  $f$  is a majority or a minority operation on  $\tilde{X}$ , then  $F$  is a majority or a minority aggregator on  $X$ .

Next, let  $F = (f_1, \dots, f_n)$  be a majority or a minority aggregator for  $X$ . We define a ternary function  $f : \mathbb{D}^3 \rightarrow \mathbb{D}$  as follows. Let  $(x, j), (y, s), (z, t)$  be three elements of  $\mathbb{D}$ .

- If  $j = s = t$ , then we set  $f((x, j), (y, s), (z, t)) = (f_j(x, y, z), j)$ .
- If  $j, s, t$  are *not* all equal, then if at least two of  $(x, j), (y, s), (z, t)$  are equal to each other, we set
 
$$f((x, j), (y, s), (z, t)) = \text{maj}((x, j), (y, s), (z, t)),$$
 if  $F$  is a majority aggregator on  $X$ , and we set
 
$$f((x, j), (y, s), (z, t)) = \oplus((x, j), (y, s), (z, t)),$$
 if  $F$  is a minority aggregator on  $X$ ;
- otherwise, we set  $f((x, j), (y, s), (z, t)) = (x, j)$ .

It is easy to see that if  $F$  is a majority or a minority aggregator for  $X$ , then  $f$  is a conservative majority or a conservative minority polymorphism on  $\tilde{X}$ . It follows that  $X$  admits a majority or a minority aggregator if and only if  $\tilde{X}$  is closed under a conservative majority or minority polymorphism. We directly apply the algorithm of Bessiere et al. and Carbonnel that efficiently detects conservative majority or conservative minority polymorphisms, respectively, and, when it has, computes them [22, 48], to  $\Gamma = \{\tilde{X}\}$ .  $\square$

We end this subsection by showing that using the graph  $H_X$ , we can compute a binary aggregator for  $X$  that has as many symmetric components as possible. This will allow us to obtain better complexity bounds in the sequel.

Given a domain  $X \subseteq \mathcal{D}^n$  and a binary aggregator  $F = (f_1, \dots, f_n)$  for  $X$ , we say that  $F$  is a *maximum symmetric* aggregator for  $X$  if, for every binary aggregator  $G = (g_1, \dots, g_n)$  for  $X$ , for every  $j \in \{1, \dots, n\}$  and for all binary  $B_j \subseteq X_j$ , if  $g_j \upharpoonright_{B_j}$  is symmetric, then so is  $f_j \upharpoonright_{B_j}$ . Note that a maximum symmetric aggregator does not necessarily have any symmetric components, for example in case  $X$  is an impossibility domain. Furthermore, if  $F$  and  $G$  are both maximum symmetric aggregators for  $X$ , then they differ only on inputs on which they are not symmetric.

**Lemma 7.3.3.** *Every domain  $X$  admits a maximum symmetric aggregator.*

*Proof.* Assume that there is no maximum symmetric aggregator for  $X$ . Then, there exist two indices  $i, j \in \{1, \dots, n\}$  and two binary subsets  $B_i \subseteq X_i$ ,  $B_j \subseteq X_j$ , such that:

- there are two binary aggregators  $F = (f_1, \dots, f_n)$  and  $G = (g_1, \dots, g_n)$  such that  $f_i \upharpoonright_{B_i}$  and  $g_j \upharpoonright_{B_j}$  are symmetric and
- there is no binary aggregator  $H = (h_1, \dots, h_n)$  such that both  $h_i \upharpoonright_{B_i}$  and  $h_j \upharpoonright_{B_j}$  are symmetric.

Let  $H = (h_1, \dots, h_n)$  be the  $n$ -tuple of binary functions such that, for all  $l \in \{1, \dots, n\}$  and for all  $x, y \in X_l$ :  $h_l(x, y) := g_l(f_l(x, y), f_l(y, x))$ . That  $H$  is an aggregator for  $X$  follows easily from the fact that  $F$  and  $G$  are. Let also  $B_i = \{a, b\}$  and  $B_j = \{c, d\}$ . Since  $f_i \upharpoonright_{B_i}$  is symmetric,  $f_i(a, b) = f_i(b, a)$ , thus:

$$h_i(a, b) = g_i(f_i(a, b), f_i(b, a)) = g_i(f_i(a, b), f_i(a, b)) = f_i(a, b).$$

It follows that  $h_i \upharpoonright_{B_i}$  is symmetric. Now, by the hypothesis,  $f_j \upharpoonright_{B_j}$  is not symmetric. Assume w.l.o.g. that  $f_j \upharpoonright_{B_j} = \text{pr}_1^2$ . Thus, it holds that:

$$h_j(c, d) = g_i(f_i(c, d), f_i(d, c)) = g_i(c, d),$$

which means that  $h_j \upharpoonright_{B_j}$  is symmetric. Contradiction.  $\square$

To proceed, we discuss a result concerning the structure of the graph  $H_X$ . We say that two scc's  $S_p$  and  $S_q$  of  $H_X$  are *related* if there exists a  $j \in \{1, \dots, n\}$  and two distinct elements  $u, u' \in X_j$ , such that  $uu'_j \in S_p$  and  $u'u_j \in S_q$ .

**Lemma 7.3.4.** *Let  $X$  be a set of feasible voting pattern and assume that  $S_p, S_q$  and  $S_r$  are three pairwise distinct scc's of  $H_X$ . Then,  $S_p, S_q$  and  $S_r$  cannot be pairwise related.*

*Proof.* To obtain a contradiction, assume they are. Then, there exist (not necessarily distinct) indices  $i, j, l \in \{1, \dots, n\}$  and pairwise distinct elements  $u, u' \in X_i, v, v' \in X_j$  and  $w, w' \in X_l$  such that  $uu'_i, vv'_j \in S_p, ww'_l, u'u_i \in S_q$  and  $v'v_j, w'w_l \in S_r$ . Since  $uu'_i \rightarrow\rightarrow vv'_j$  and  $vv'_j \rightarrow\rightarrow uu'_i$ , it follows that  $v'v_j \rightarrow\rightarrow u'u_i$  and  $u'u_i \rightarrow\rightarrow v'v_j$ . Thus,  $S_q$  and  $S_r$  form together an scc of  $H_X$ . Contradiction.  $\square$

We are now ready to show that we can find a maximum symmetric aggregator for a domain  $X$ , in polynomial time to its size.

**Corollary 7.3.2.** *There is a polynomial-time algorithm for solving the following problem: given a set  $X$  of feasible evaluations, produce a maximum symmetric aggregator for  $X$ .*

*Proof.* Construct the graph  $H_X$ . For a set of vertices  $S$ , let  $N^+(S)$  and  $N^-(S)$  be its *extended outwards and inwards neighborhood* respectively in  $H_X$ . That is,  $N^+(S) = S \cup \{uu'_i \mid \exists vv'_j \in S : vv'_j \rightarrow\rightarrow uu'_i\}$  and  $N^-(S) = S \cup \{uu'_i \mid \exists vv'_j \in S : uu'_i \rightarrow\rightarrow vv'_j\}$ .

We define the  $n$ -tuple of binary functions  $F = (f_1, \dots, f_n)$  as follows. If  $H_X$  is strongly connected, set  $f_j = \text{pr}_1^2$  for all  $j \in \{1, \dots, n\}$ . Else, assume w.l.o.g. that  $H_X$  is *connected*. If it is not, we can deal with each connected component independently in the same way. Assume that  $S_1, \dots, S_t, t \geq 2$ , are the scc's of  $H_X$ , in their topological order.

1. For each  $uu'_i \in S_1$ , set  $f_i(u, u') = u'$ .



2. Let  $S$  be the set of vertices of every scc of  $H_X$  that is related with  $S_1$ . For each  $vv'_j \in N^+(S)$ , set  $f_j(v, v') = v$ .
3. Let  $S'$  be the set of vertices of every scc of  $H_X$  that is related with an scc of  $S$ . Note that due to Lemma 7.3.4, such an scc cannot be related with  $S_1$ . For each  $ww'_j \in N^-(S)$ , set  $f_j(w, w') = w'$ .

Note that any remaining scc must be in another connected component. If this is the case, we proceed as above for each such connected component. Finally, to be formally correct, let  $f_j(a, b) = a$  for every  $j \in \{1, \dots, n\}$  and  $a, b \in \mathcal{D}$  such that either  $a$  or  $b \notin X_j$ .

We have already shown that, given  $X$ ,  $H_X$  can be constructed in polynomial time to its size and its scc's can also be computed in linear time to the size of  $H_X$ . Steps 1 – 3 can easily be implemented by checking once every scc of  $H_X$ . Thus, the overall process is clearly polynomial.

It remains to show that  $F = (f_1, \dots, f_n)$  is indeed a maximum symmetric aggregator for  $X$ . To obtain a contradiction, suppose it is not. Then, there exist a  $i \in \{1, \dots, n\}$ , a binary subset  $B_i \subseteq X_i$  and a binary aggregator  $G = (g_1, \dots, g_n)$  for  $X$  such that  $f_i|_{B_i}$  is not symmetric, whereas  $g_i|_{B_i}$  is.

Assume that  $B_i = \{u, u'\}$ . Since  $G$  is a binary aggregator for  $X$  and  $g_i|_{B_i}$  is symmetric, by Lemma 7.3.1 there are no paths  $uu'_i \rightarrow\rightarrow u'u_i$  or  $u'u_i \rightarrow\rightarrow uu'_i$  in  $H_X$ . Consequently, there are two distinct scc's of  $H_X$ , say  $S_p$  and  $S_q$ , such that  $uu'_i \in S_p$  and  $u'u_i \in S_q$  and there are no paths connecting a vertex in  $S_p$  with a vertex in  $S_q$ . Given the way we constructed  $F = (f_1, \dots, f_n)$ , the vertices of both these scc's are either all in  $S$  or they are all in  $S'$ . We show that that in both cases, there exist three pairwise distinct scc's of  $H_X$  that are pairwise related. This is a contradiction, by Lemma 7.3.4.

First, assume that the vertices of  $S_p$  and  $S_q$  are all in  $S$ . The case where the vertices of  $S_p$  and  $S_q$  are all in  $S'$  is analogous. If both of them are related with  $S_1$ , then  $S_1, S_p$  and  $S_q$  are pairwise related. Contradiction. Else, without loss of generality, assume that  $S_p$  is not related to  $S_1$ . Then, there exists some scc  $S_r$  of  $H_X$  that is related with  $S_1$ , such that  $S_p \subseteq N^+(S_r)$ . Since  $S_r$  is related with  $S_1$ , there is some vertex  $vv'_j \in S_1$  such that  $v'v_j \in S_r$ . Then  $v'v_j \rightarrow\rightarrow uu'_i$ , which implies that  $u'u_i \rightarrow\rightarrow vv'_j$ . Contradiction, since we took the scc's of  $H_X$  in their topological order.  $\square$

### 7.3.2 Expressibility in Transitive Closure Logic

We now show that Theorem [7.3.1](#) can be refined to show that the test of whether  $X$  admits a binary non-dictatorial aggregator can be expressed in Transitive Closure Logic. For this, we need to first encode a set  $X$  of feasible evaluations by a suitable finite structure.

We consider a relational schema  $\mathbf{R}$  consisting of three unary relations  $X'$ ,  $I'$ ,  $V'$ , and one ternary relation  $R'$ . Intuitively,  $X'$  will be interpreted by a set of feasible evaluations,  $I'$  will be interpreted by the set of issues at hand, and  $V'$  will be interpreted by the set of positions over all issues.

Given a set  $X \subseteq A^m$  of feasible evaluations, we let  $\mathbf{A}(X) = (A, X, I, V)$  be the following finite  $\mathbf{R}$ -structure:

- $A = X \cup I \cup V$ , where  $I = \{1, \dots, n\}$  is the set of issues, and  $V$  is the union of all positions over all issues.
- $R$  is the ternary relation consisting of all triples  $(x, j, v)$  such that  $x \in X$  and  $v$  is the  $j$ -th coordinate of  $x$ , i.e., the position for issue  $j$  in  $x$ .

It is clear that  $X$  can be identified with the finite structure  $\mathbf{A}(X)$ . Conversely, if we are given a finite  $\mathbf{R}$ -structure  $\mathbf{A}$  in which  $R \subseteq X \times I \times V$ , then  $X$  can be thought of as a set of feasible evaluations over the issues  $I$ .

**Lemma 7.3.5.** *There is a first-order formula  $\varphi(u, u', k, v, v', l)$  such that, for every set  $X$  of feasible evaluations, we have that  $\varphi(u, u', k, v, v', l)$  defines the edge relation of the directed graph  $H_X$ , when interpreted on the finite structure  $\mathbf{A}(X)$ .*

*Proof.* Consider the following first-order formula  $\varphi(u, u', k, v, v', l)$ :

$$\begin{aligned} & \exists z \exists z' ((X'(z) \wedge X'(z') \wedge R'(z, k, u) \wedge R'(z, l, v) \wedge R(z', k, u') \wedge R(z', l, v')) \\ & \wedge \neg \exists y (X'(y) \wedge R'(y, k, u) \wedge R'(y, l, v')) \\ & \wedge \forall j \forall w (R'(y, j, w) \rightarrow (R'(z, j, w) \vee R'(z', j, w))))). \end{aligned}$$

It is immediate from the definition of  $H_X$  that the formula  $\phi(u, u', k, v, v', l)$  defines indeed the edge relation of  $H_X$ .  $\square$

Transitive Closure Logic (TCL) extends first-order logic with the ability to form the transitive closure of first-order definable relations; see e.g. [\[161\]](#). As such, it is a fragment of Least Fixed-Point Logic LFP. Both TCL and LFP

have been extensively studied in the context of *descriptive complexity* [129, 161], where the aim is to express and study algorithmic problems using logical formalisms. It is known that TCL is contained in NLOGSPACE and that LFP is contained in PTIME on the class of all finite graphs. Furthermore, it is also known these containments are strict on the class of all finite graphs, while  $\text{TCL} = \text{NLOGSPACE}$  and  $\text{LFP} = \text{PTIME}$  on the class of all finite *ordered* graphs (that is, graphs with an additional total order on their vertices that can be used in TCL and LFP formulas).

**Theorem 7.3.3.** *The following problem is expressible in Transitive Closure Logic: given a set  $X$  of feasible evaluations (encoded as the finite structure  $\mathbf{A}(X)$ ), does  $X$  admit a binary, non-dictatorial aggregator? Hence, this problem is also expressible in LFP.*

*Proof.* The result follows immediately from Lemma 7.3.2, Lemma 7.3.5, and the definition of Transitive Closure Logic.  $\square$

It is known that every property that can be expressed in Transitive Closure Logic is in NLOGSPACE. Thus, the problem of detecting if  $X$  admits a binary non-dictatorial aggregator is in NLOGSPACE. Note that membership of this problem in NLOGSPACE could also be inferred from Lemma 7.3.2 and the observation that the graph  $H_X$  can be constructed in LOGSPACE. Lemma 7.3.5 strengthens this observation by showing that  $H_X$  is actually definable in first-order logic, which is a small fragment of LOGSPACE.

Now, let  $X \subseteq \{0, 1\}^n$ . We prove the following result.

**Lemma 7.3.6.** *Checking whether a domain  $X \subseteq \{0, 1\}^n$  is affine can be done in LOGSPACE.*

*Proof.* Suppose we have a Turing Machine with a read-only tape containing the tuples of  $X$ . In the work tape, we store triples  $(i_1, i_2, i_3)$  of integers in  $\{1, \dots, k\}$  in binary. This takes  $O(\log k)$  space. The integer  $i_j$  points to the  $i_j$ -th element of  $X$ . We want to examine if the sum modulo 2 of these three elements is also in  $X$ .

To do that, for each such triple, we examine all integers  $i_4 \leq k$  one at a time. This adds another  $\log k$  number of cells in the work tape. We then store all integers  $j \leq n$  binary, one at a time, using another  $\log n$  bits to the work tape.

Once we have  $i_1, i_2, i_3, i_4$ , and  $j$  on the work tape, we check whether the entry  $a_j^{i_4} = \oplus(a_j^{i_1}, a_j^{i_2}, a_j^{i_3})$ . If it is, we go to the next  $j$ . If it is not, we go

to the next  $i_4$ , and when we are done with the triple  $(i_1, i_2, i_3)$ , we go to the next such triple. If for every triple  $(i_1, i_2, i_3)$ , we find a suitable integer  $i_4$ ,  $X$  is affine. Else, it is not.  $\square$

By Corollary [7.1.2](#), Lemma [7.3.6](#) and the discussion following Theorem [7.3.3](#), we obtain the following result.

**Theorem 7.3.4.** *The following problem is in NLOGSPACE: given a set  $X \subseteq \{0, 1\}^n$  of feasible evaluations in the Boolean framework, decide whether or not  $X$  is a possibility domain.*

### 7.3.3 Tractability of Uniform Possibility Domains

The final results of this section is about the complexity of detecting uniform possibility domains.

**Theorem 7.3.5.** *There is a polynomial-time algorithm for solving the following problem: given a set  $X$  of feasible evaluations, determine whether or not  $X$  is a uniform possibility domain and, if it is, produce a ternary weak near-unanimity aggregator for  $X$ .*

*Proof.* By Theorem [7.1.4](#), a set  $X$  of feasible evaluations is a uniform possibility domain if and only if there is a ternary aggregator  $F = (f_1, \dots, f_n)$  such that each  $f_j$  is a weak near-unanimity operation, i.e., for all  $j \in \{1, \dots, n\}$  and for all  $x, y \in X_j$ , we have that  $f_j(x, y, y) = f_j(y, x, y) = f_j(y, y, x)$ . As in the proof of Theorem [7.3.2](#), we can go back-and-forth between  $X$  and the set  $\tilde{X}$  and verify that  $X$  is a uniform possibility domain if and only if  $\tilde{X}$  has a ternary, conservative, weak near-unanimity polymorphism. Theorem [4.2.10](#) then implies that the existence of such a polymorphism can be tested in polynomial time, and that such a polymorphism can be produced in polynomial time, if one exists.  $\square$

In the Boolean case, we can prove the tractability of detecting locally non-dictatorial aggregators without using Theorem [4.2.10](#). This will allow us to obtain better complexity bounds in Subsec. [7.3.4](#). In the Boolean case, Theorem [7.1.4](#) has been strengthened by Diaz et al.:

**Corollary 7.3.3.** *Diaz et al. [[72](#), Corollary 4.1] A Boolean domain  $X \subseteq \{0, 1\}^n$  is a local possibility domain if and only if it admits a ternary aggregator  $F = (f_1, \dots, f_n)$  such that  $f_j \in \{\wedge^{(3)}, \vee^{(3)}, \oplus\}$ , for  $j = 1, \dots, n$ .*

Thus, we can obtain the following algorithm in the Boolean case.

**Corollary 7.3.4.** *There is a polynomial-time algorithm for solving the following problem: given a Boolean set  $X \subseteq \{0, 1\}^n$  of feasible evaluations, determine whether or not  $X$  is a local possibility domain and, if it is, produce a ternary aggregator  $F = (f_1, \dots, f_n)$  for  $X$  such that  $f_j \in \{\wedge^{(3)}, \vee^{(3)}\}, \oplus$ ,  $j = 1, \dots, n$ .*

*Proof.* Let  $F = (f_1, \dots, f_n)$  be the binary maximal symmetric aggregator obtained by Corollary 7.3.2 and let  $I, J \subseteq \{1, \dots, m\}$  such that  $f_i = \wedge$  for all  $i \in I$  and  $f_j = \vee$  for all  $j \in J$  (both  $I$  and  $J$  can be empty). We prove that  $X$  is a local possibility domain if and only if it admits the ternary aggregator  $G = (g_1, \dots, g_n)$ , where  $g_i = \wedge^{(3)}$  for all  $i \in I$ ,  $g_j = \vee^{(3)}$  for all  $j \in J$  and  $g_k = \oplus$ , for all  $k \in \{1, \dots, n\} \setminus (I \cup J)$ . Since we can obtain  $F$  in polynomial-time to the size of  $X$  and since checking whether  $G$  is an aggregator for  $X$  can be done also in polynomial time, the procedure is clearly polynomial to the size of  $X$ .

That  $X$  is a local possibility domain if it admits  $G$  is self-evident. Assume now that  $X$  is a local possibility domain. Let also  $F^{(3)} = (f_1^{(3)}, \dots, f_m^{(3)})$  be the  $m$ -tuple of ternary functions, where

$$f_j^{(3)}(x, y, z) = f_j(f_j(x, y), z),$$

$j = 1, \dots, m$ . It is straightforward to check that (i)  $F^{(3)}$  is an aggregator for  $X$ , (ii)  $f_i^{(3)} = \wedge^{(3)}$  for all  $i \in I$ , (iii)  $f_j^{(3)} = \vee^{(3)}$  for all  $j \in J$  and (iv)  $f_k^{(3)} \in \{\text{pr}_1^3, \text{pr}_3^3\}$  for all  $k \in \{1, \dots, m\} \setminus (I \cup J)$ .

Now, since  $X$  is a local possibility domain, by Corollary 7.3.3,  $X$  admits a ternary aggregator  $H = (h_1, \dots, h_n)$ , such that  $h_j \in \{\wedge^{(3)}, \vee^{(3)}, \oplus\}$ ,  $j = 1, \dots, m$ . If  $H = G$ , there is nothing to prove.

First, assume that there is some  $k \notin I \cup J$ , such that  $h_k \in \{\wedge^{(3)}, \vee^{(3)}\}$ . Now, let  $G' = (g'_1, \dots, g'_n) := H \diamond F^{(3)}$  and  $F' = (f'_1, \dots, f'_n)$  be a binary  $m$ -tuple of functions such that:

$$f'_j(x, y) := g'(x, x, y), \text{ for all } x, y \in A.$$

It holds that  $G'$  is an aggregator for  $X$  such that  $g'_j \in \{\wedge^{(3)}, \vee^{(3)}\}$ , for all  $j \in I \cup J \cup \{k\}$ . Thus,  $F'$  is a non-dictatorial aggregator and  $f'_j$  is symmetric for all  $j \in I \cup J \cup \{k\}$ . Contradiction, since  $F$  is a maximal symmetric aggregator and  $f_k$  is not symmetric.

Finally, suppose that there is some  $j \in I \cup J$  such that  $h_k = \oplus$ , for all  $j \in K \cup \{j\}$ . Then,  $G = H \diamond F^{(3)}$  and thus is an aggregator for  $X$ .  $\square$

### 7.3.4 Implicitly given Domains

We now proceed to establish complexity bounds for checking whether a domain is a possibility domain or a local possibility domain in the two variants of the logic-based approach (recall that a local possibility domain is a uniform possibility domain in the Boolean framework), where  $X$  is provided either via an agenda, or as the truth set of an integrity constraint.

Suppose that we have a *propositional formula*  $\phi$  on  $n$  variables  $x_1, \dots, x_n$ . Each variable  $x_j$  corresponds to the  $j$ -th issue,  $j = 1, \dots, n$ , where the possible positions are 0 and 1. Let  $X_\phi := \text{Mod}(\phi)$  be the set consisting of all  $n$ -ary vectors of *satisfying truth assignments* of  $\phi$ . In this setting, we say that  $\phi$  is an *integrity constraint*.

For the second variant, suppose that we have an *agenda*  $\bar{\phi} = (\phi_1, \dots, \phi_n)$  of  $n$  propositional formulas. For a formula  $\psi$  and an  $x \in \{0, 1\}$ , let

$$\psi^x := \begin{cases} \psi & \text{if } x = 1, \\ \neg\psi & \text{if } x = 0. \end{cases}$$

Finally, let:

$$X_{\bar{\phi}} := \left\{ \mathbf{x} = (x_1, \dots, x_n) \in \{0, 1\}^n \mid \bigwedge_{j=1}^n \phi_j^{x_j} \text{ is satisfiable} \right\}.$$

Recall that, as usual in aggregation theory, we have assumed that domains  $X$  are non-degenerate, i.e.  $|X_j| \geq 2$  (thus  $X_j = \{0, 1\}$  in the Boolean framework), for  $j = 1, \dots, n$ . Thus, we assume that both integrity constraints and agendas are such that their domains are non-degenerate. On the other hand, when we consider (propositional) formulas, we do not assume anything regarding their domain (it can even be empty).

It is well known that given a domain  $X \subseteq \{0, 1\}^n$ , there is a formula  $\phi$  such that its set of models  $\text{Mod}(\phi)$  is equal to  $X$ ; see e.g. [80]. Dokow and Holzman prove that there is also an agenda  $\bar{\phi}$  such that  $X_{\bar{\phi}} = X$  [77]. Thus the three variants (explicit representation, implicit representation via an integrity constraint, and implicit representation via an agenda) are in some sense equivalent, as regards the existence of (local) non-dictatorial aggregators. However, neither the integrity constraint nor the agenda describing a given domain need be unique. Thus, there is a possible loss of information when passing from one variant to another; as List and Puppe argue, this can be significant for certain aspects of the aggregation problem [163].

In terms of the computational complexity of passing from one framework to another, Zanuttini and Hébrard show that given a domain  $X$ , one can construct a formula  $\phi$  such that  $\text{Mod}(\phi) = X$  in polynomial time in the size of the domain [219]. Also, the construction in [75], where given a domain  $X$ , we obtain an agenda  $\bar{\phi}$  such that  $X_{\bar{\phi}} = X$  can obviously be carried out in polynomial time to the size of the domain. It is very easy to find integrity constraints and agendas whose domains are exponentially large on their respective sizes: consider for example the integrity constraint  $(x_1 \vee \neg x_1) \wedge \cdots \wedge (x_n \vee \neg x_n)$  and the agenda  $(x_1, \dots, x_n)$ , where  $x_1, \dots, x_n$  are pairwise distinct variables. Both have domains equal to the full Boolean domain  $\{0, 1\}^n$ . Finally, Endriss et al. show that, unless the *polynomial hierarchy* collapses, we cannot describe an agenda by an integrity constraint of polynomial size to that of the agenda and that, given an integrity constraint  $\phi$ , the problem of finding an agenda  $\bar{\phi}$  such that  $X_{\bar{\phi}} = X_\phi$  is FNP-complete [83].

Here, we examine the computational complexity of checking if a domain  $X$  is a (local) possibility domain in both the integrity constraint variant and the agenda variant. In all that follows, we assume that the integrity constraints are defined on at least three variables and agendas contain at least three propositional formulas, since domains  $X \subseteq \{0, 1\}^n$  where  $n = 1$  or  $2$  are all possibility domains.

**Integrity Constraints** Let  $\phi$  be an integrity constraint on  $n$  variables and let  $X_\phi = \text{Mod}(\phi) \subseteq \{0, 1\}^n$ . The following Theorem provides an upper bound to the complexity of checking if  $X_\phi$  is a possibility domain.

**Theorem 7.3.6.** *Deciding, on input  $\phi$ , whether  $X_\phi$  admits a non-dictatorial aggregator is in  $\Sigma_2^P \cap \Pi_2^P$ .*

*Proof.* By Corollary [7.1.2],  $X_\phi$  is a possibility domain if and only if it admits a binary non-dictatorial aggregator or it is affine. The problem of whether  $X_\phi$  is affine is in  $\Pi_1^P = \text{coNP}$  (and thus in  $\Sigma_2^P \cap \Pi_2^P$  too), since it can be cast as follows.

For all  $n$ -tuples  $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \{0, 1\}$ , if all three satisfy  $\phi$ , then so does the  $n$ -tuple  $\oplus(\mathbf{x}, \mathbf{y}, \mathbf{z})$ .

For the above problem concerning the existence of binary non-dictatorial aggregators for  $X_\phi$ , we will show separately that it is both in  $\Sigma_2^P$  and in  $\Pi_2^P$ . For the former, note that there are only four conservative (equivalently Paretian)

functions from  $\{0, 1\}^2 \mapsto \{0, 1\}$ , namely  $\text{pr}_1^2$ ,  $\text{pr}_2^2$ ,  $\wedge$  and  $\vee$ . Therefore, there are  $4^n - 2$  tuples  $F = (f_1, \dots, f_n)$ , with  $f_j : \{0, 1\}^2 \mapsto \{0, 1\}$ ,  $j = 1, \dots, n$  of  $n$  conservative functions, where not all  $f_j$  are the same projection function. Such an  $n$ -tuple can be thought of as a binary  $(2 \times 3)n$ -sequence, where each  $f_j$  is encoded by a sequence  $(01a10b)$ ,  $a, b \in \{0, 1\}$ , meaning that  $f_j(0, 1) = a$  and  $f_j(1, 0) = b$ .

Let us call such  $F$  *candidates for non-dictatorial aggregators*. The question of deciding whether a given  $F$  is one of the  $4^n - 2$  candidates for non-dictatorial aggregators is easily in P. Also, the question of whether a given binary  $F$  is an aggregator for  $X_\phi$  can be cast as:

For all  $n$ -tuples  $\mathbf{x}, \mathbf{y} \in \{0, 1\}^m$ , if both satisfy  $\phi$  then so does the  $n$ -tuple  $F(\mathbf{x}, \mathbf{y})$

and is thus in  $\Pi_1^P$ . Therefore the problem of whether  $X_\phi$  admits a binary non-dictatorial aggregator is in  $\Sigma_2^P$  because it can be cast as:

There exists a  $F = (f_1, \dots, f_n)$  such that  $F$  is a candidate for non-dictatorial aggregator and for all  $m$ -tuples  $\mathbf{x}, \mathbf{y} \in \{0, 1\}^n$ , if both satisfy  $\phi$ , then so does the  $n$ -tuple  $F(\mathbf{x}, \mathbf{y})$ .

To show that it is also in  $\Pi_2^P$ , recall that, by Lemma [7.3.2](#), the set  $X_\phi$  admits a binary non-dictatorial aggregator if and only if the graph  $H_X$  is not strongly connected. We will show that checking if  $H_{X_\phi}$  is strongly connected is in  $\Sigma_2^P$ , which means that checking if  $H_{X_\phi}$  is not strongly connected is in  $\Pi_2^P$ .

First note that the size of  $H_X$  is polynomial in the size of  $\phi$ , since it has  $2m$  nodes, where  $n$  is the number of variables of  $\phi$ . Thus, it suffices to prove that testing whether two nodes of  $H_{X_\phi}$  are connected is in NP with an oracle in coNP.

To test if two nodes are connected, we first obtain a path witnessing this. To verify it is indeed a path, we need to check if any two of its consecutive nodes, say  $uu'_s$  and  $vv'_t$ , are connected by the edge  $uu'_s \rightarrow vv'_t$  in  $H_X$ . To do that, we can again take the satisfying assignments  $z$  and  $z'$  of  $\phi$  that witness that  $uu'_s \rightarrow vv'_t$ . Then, using the coNP oracle, we need to check that there is no  $z^*$  that: (i) satisfies  $\phi$ , (ii) extends  $uv'$  and (iii) agrees on every coordinate either  $z$  or  $z'$ . □

We now show the analogous result for local possibility domains.



**Theorem 7.3.7.** *Deciding, on input  $\phi$ , whether or not  $X_\phi$  admits a locally non-dictatorial aggregator is in  $\Sigma_2^P \cap \Pi_2^P$ .*

*Proof.* We follow the proof of Theorem 7.3.6. By Corollary 7.3.3 we have only three functions, namely  $\wedge^{(3)}$ ,  $\vee^{(3)}$ ,  $\oplus$ , which, when combined to an  $n$ -ary tuple  $F = (f_1, \dots, f_n)$ , form a local non-dictatorial aggregator for  $X$ . Thus, the proof is exactly the same, with the difference that we now have  $3^n$  tuples that can be encoded as  $(6 \times 3)n$ -binary sequences and we conclude that the problem is in  $\Sigma_2^P$ .

For the containment in  $\Pi_2^P$ , we argue as follows. Given access to  $H_{X_\phi}$  we can, by Corollary 7.3.4, obtain in polynomial time a ternary WNU aggregator  $G = (g_1, \dots, g_n)$  such that  $X_\phi$  is a local possibility domain if and only if it admits  $G$ . Whether  $X_\phi$  admits  $G$  or not is in  $\text{coNP}$  (in the same way we check if  $X_\phi$  is affine) and thus in  $\Pi_2^P$  too. Since the size of  $H_{X_\phi}$  is polynomial to that of  $\phi$ , it suffices to show that testing whether there is no edge  $uu'_i \rightarrow\rightarrow vv'_j$  in  $H_{X_\phi}$  is in  $\Pi_2^P$ . This problem can be expressed as:

For all assignments  $\mathbf{a} = (a_1, \dots, a_n)$ ,  $\mathbf{b} = (b_1, \dots, b_n)$ , there exists an assignment  $\mathbf{c} = (c_1, \dots, c_n)$  such that, if  $\mathbf{a}$ ,  $\mathbf{b}$  satisfy  $\phi$  and  $a_i = u$ ,  $b_i = u'$ ,  $a_j = v$ ,  $b_j = v'$ , then  $\mathbf{c}$  satisfies  $\phi$ ,  $c_i = u$ ,  $c_j = v'$  and  $c_l \in \{a_l, b_l\}$  for all  $l \in \{1, \dots, n\} \setminus \{i, j\}$ .

Thus, the proof is complete.  $\square$

In terms of lower bounds, we provide polynomial-time reductions from two  $\text{coNP}$ -complete problems: the *semantical independence* problem and the *unsatisfiability* problem for propositional formulas. The latter is the well known problem of whether a formula has no satisfying assignments. In the former, we are asking whether a given propositional formula is (*semantically*) *dependent* to all its variables. That is, there is no variable (or set of variables) such that whether an assignment of values satisfies the given formula or not, does not depend on the values of the variable(s). For a systematic overview of this problem and its variations, see 156 and 157. In the setting of agendas, this notion has been studied under the name *agenda separability*, in 158.

For domain  $X \subseteq \{0, 1\}^n$  and a nonempty subset  $I \subseteq \{1, \dots, n\}$ , we denote by  $X_I$  the projection of  $X$  to  $I$ , that is the set of all partial vectors with indices in  $I$  that can be extended to elements of  $X$ .

**Definition 7.3.1.** *Let  $\phi(x_1, \dots, x_m)$  be a propositional formula, where  $X := \text{Mod}(\phi)$  and let  $V \subseteq \{x_1, \dots, x_m\}$  be a subset of its variables. Let also  $i \in \{1, \dots, m\}$ . We say that  $\phi$  is:*

i. (semantically) independent from variable  $x_i$  if:

$$X \approx X_{\{i\}} \times X_{-\{i\}}$$

ii. (semantically) independent from the set of variables  $V$  if it is independent from every  $x_j \in V$ .

In our setting, an integrity constraint being independent from a variable  $x_j$  means that issue  $j$  does not contribute anything in the logical consistency restrictions imposed by the constraint. Lang et al. showed that the problem of checking if a propositional formula depends on all its variables (is *simplified variable-dependent* in their terminology), is **coNP**-complete [157].

To make our reductions easier to follow, we work with the specific domain:

$$\text{Imp} := \{0, 1\}^3 \setminus \{(0, 0, 0), (1, 1, 1)\}.$$

Observe that  $\text{Imp}$  corresponds to a natural and well studied problem in both preference and judgment aggregation. Suppose that we have three alternatives  $A$ ,  $B$  and  $C$ , where issue 1 corresponds to deciding between  $A$  and  $B$ , issue 2 corresponds to deciding between  $B$  and  $C$ , and issue 3 corresponds to deciding between  $C$  and  $A$ . In that setting, we can assume that, in each issue, 1 denotes preferring the former option and 0 the latter. One can easily see now that  $\text{Imp}$  corresponds to the natural requirement of transitivity to our preferences.

**Lemma 7.3.7.** *Imp is an impossibility domain.*

*Proof.* By Corollary 7.1.2, we only need to check if  $\text{Imp}$  is affine, or if it admits a binary non-dictatorial aggregator. Easily.

$$\bar{\oplus}((1, 0, 0), (0, 1, 0), (0, 0, 1)) = (0, 0, 0) \notin \text{Imp},$$

thus  $\text{Imp}$  is not affine.

On the other hand, let  $F = (f_1, f_2, f_3)$  be a binary non-dictatorial thuple of functions. There are  $4^3 - 2 = 62$  cases for  $F$ . We arbitrarily choose to show three of them. The rest are left to the interested reader.

- If  $f_1 = f_2 = \wedge$  and  $f_3 = \vee$ , then  $F((1, 0, 0), (0, 1, 0)) = (0, 0, 0) \notin \text{Imp}$ .
- If  $f_1 = \wedge$ ,  $f_2 = \vee$  and  $f_3 = \text{pr}_1^2$ , then  $F((1, 0, 0), (0, 0, 1)) = (0, 0, 0) \notin \text{Imp}$ .

- If  $f_1 = \vee$ ,  $f_2 = \text{pr}_1^2$  and  $f_3 = \text{pr}_2^2$ , then  $F((0, 0, 1), (0, 1, 0)) = (0, 0, 0) \notin \text{Imp}$ .

Thus, for  $F$  to be an aggregator for  $\text{Imp}$ , it must hold that  $f_1 = f_2 = f_3 = \text{pr}_d^2$ ,  $d = 1, 2$  and, consequently  $\text{Imp}$  admits only dictatorial binary aggregators.  $\square$

Let  $\psi$  be the propositional formula:

$$\psi(y_1, y_2, y_3) = (y_1 \vee y_2 \vee y_3) \wedge (\neg y_1 \vee \neg y_2 \vee \neg y_3). \quad (7.17)$$

Easily,  $\text{Mod}(\psi) = \text{Imp}$ . We are now ready to obtain our reductions.

**Theorem 7.3.8.** *Deciding, on input  $\phi$ , whether or not  $X_\phi$  admits a non-dictatorial aggregator is  $\text{coNP}$ -hard.*

*Proof.* Let  $\chi(x_1, \dots, x_l)$  be a propositional formula on  $l$  variables. We construct, in polynomial time, a formula  $\phi$  such that  $\chi$  is independent from at least one of its variables if and only if  $X_\phi$  is a possibility domain. In all that follows,  $n = l + 3$ .

Let:

$$\phi(x_1, \dots, x_l, y_1, y_2, y_3) = \chi(x_1, \dots, x_l) \oplus \psi(y_1, y_2, y_3),$$

where:  $\{x_1, \dots, x_l\}$  and  $\{y_1, y_2, y_3\}$  are disjoint sets of variables.

First, note that the length  $|\phi|$  of  $\phi$  is linear to that of  $\chi$ , since  $|\phi| = |\chi| + 6$ . Thus the construction is polynomial.

By (7.17), it holds that:

$$X_\phi = \left( \text{Mod}(\chi) \times \{(0, 0, 0), (1, 1, 1)\} \right) \cup \left( \text{Mod}(\neg\chi) \times \text{Imp} \right). \quad (7.18)$$

We first consider the two extreme cases. If  $\chi$  is unsatisfiable or a tautology, then by (7.18) we have that:

$$X_\phi = \{0, 1\}^l \times \text{Imp},$$

or that

$$X_\phi = \{0, 1\}^l \times \{(0, 0, 0), (1, 1, 1)\}$$

respectively. In both cases, we have that  $\chi$  is independent from all its variables and  $X_\phi$  is a possibility domain, since it is a Cartesian product.

Thus, we can assume that both  $\text{Mod}(\chi)$  and  $\text{Mod}(\neg\chi)$  are not empty. We proceed with a series of claims.

**Claim 7.3.1.**  $X_\phi$  is not affine.

*Proof.* Let  $\mathbf{a} := (a_1, \dots, a_l) \in \text{Mod}(\neg\chi)$ . Then,  $(\mathbf{a}, 0, 1, 1)$ ,  $(\mathbf{a}, 1, 0, 1)$  and  $(\mathbf{a}, 1, 1, 0) \in X_\phi$ . Furthermore:

$$\bar{\oplus}((\mathbf{a}, 0, 1, 1), (\mathbf{a}, 1, 0, 1), (\mathbf{a}, 1, 1, 0)) = (\mathbf{a}, 0, 0, 0) \notin X_\phi.$$

Thus,  $X_\phi$  is not affine.  $\square$

By Corollary 7.1.2 and Claim 7.3.1,  $X_\phi$  is a possibility domain if and only if it admits a binary non-dictatorial aggregator. The following claims show that such an aggregator must be in a restricted class.

**Claim 7.3.2.** Assume  $F = (f_1, \dots, f_n)$  is a binary aggregator for  $X_\phi$ . Then,  $f_{l+1} = f_{l+2} = f_{l+3} = \text{pr}_d^2$ ,  $d \in \{1, 2\}$ .

*Proof of Claim:* To obtain a contradiction, assume  $(f_{l+1}, f_{l+2}, f_{l+3}) \neq (\text{pr}_d^2, \text{pr}_d^2, \text{pr}_d^2)$ ,  $d = 1, 2$ . By Lemma 7.3.7,  $\text{Imp}$  is an impossibility domain. Thus, there exist  $\mathbf{x}, \mathbf{y} \in \text{Imp}$  such that

$$\mathbf{z} := (f_{l+1}, f_{l+2}, f_{l+3})(\mathbf{x}, \mathbf{y}) \in \{(0, 0, 0), (1, 1, 1)\}.$$

Let  $\mathbf{a} := (a_1, \dots, a_l) \in \text{Mod}(\neg\chi)$ . Then  $(\mathbf{a}, \mathbf{x}), (\mathbf{a}, \mathbf{y}) \in X_\phi$ , but:

$$F((\mathbf{a}, \mathbf{x}), (\mathbf{a}, \mathbf{y})) = (\mathbf{a}, \mathbf{z}) \notin X_\phi.$$

Thus,  $F$  is not an aggregator for  $X_\phi$ . Contradiction.  $\square$

The following claim states that  $F = (f_1, \dots, f_l, f_{l+1}, f_{l+2}, f_{l+3})$  cannot have its first  $l$  coordinates be projections to the same coordinate  $d \in \{1, 2\}$  and the last three be projections to the other. This can also be derived by (7.18), since  $X_\phi$  is not a Cartesian product. For a proof of this general and straightforward characterization, the interested reader is referred to [72]. Here we opted to showcase the technique we follow throughout the rest of the proof.

**Claim 7.3.3.** Assume  $F = (f_1, \dots, f_n)$  is a binary aggregator for  $X_\phi$ . If  $f_{l+1} = f_{l+2} = f_{l+3} = \text{pr}_d^2$ , then there is at least one  $j \in \{1, \dots, l\}$  such that  $f_j \neq \text{pr}_{d'}^2$ ,  $d, d' \in \{1, 2\}$ ,  $d \neq d'$ .

*Proof.* We show the claim for  $d = 2$  and  $d' = 1$ . The analogous arguments hold for the case where  $d = 1$  and  $d' = 2$ .

To obtain a contradiction, assume that  $f_1 = \dots = f_l = \text{pr}_1^2$ . Let  $\mathbf{a} := (a_1, \dots, a_l) \in \text{Mod}(\chi)$  and  $\mathbf{b} := (b_1, \dots, b_l) \in \text{Mod}(\neg\chi)$ . Then,  $(\mathbf{a}, 0, 0, 0)$  and  $(\mathbf{b}, 0, 0, 1) \in X_\phi$ , but:

$$F((\mathbf{a}, 0, 0, 0), (\mathbf{b}, 0, 0, 1)) = (\mathbf{a}, 0, 0, 1) \notin X_\phi.$$

Thus,  $F$  is not an aggregator for  $X_\phi$ . Contradiction.  $\square$

**Claim 7.3.4.** *Assume  $F = (f_1, \dots, f_n)$  is a binary aggregator for  $X_\phi$ . Then, there is at least one  $j \in \{1, \dots, l\}$  such that  $f_j$  is not symmetric.*

*Proof.* By Claim [7.3.2](#), for some  $d \in \{1, 2\}$ ,  $f_{l+1} = f_{l+2} = f_{l+3} = \text{pr}_d^2$ . To obtain a contradiction, assume that  $f_j$  is symmetric for all  $j \in \{1, \dots, l\}$ . Assume also w.l.o.g. that  $d = 2$ . The analogous arguments work for  $d = 1$ .

Let  $\mathbf{a} := (a_1, \dots, a_l) \in \text{Mod}(\chi)$ ,  $\mathbf{b} := (b_1, \dots, b_l) \in \text{Mod}(\neg\chi)$  and

$$(f_1, \dots, f_l)(\mathbf{a}, \mathbf{b}) := \mathbf{c}.$$

Then,  $(\mathbf{a}, 0, 0, 0)$  and  $(\mathbf{b}, 0, 0, 1) \in X$ . Since  $F$  is an aggregator for  $X_\phi$ :

$$\begin{aligned} f((\mathbf{a}, 0, 0, 0), (\mathbf{b}, 0, 0, 1)) &= (\mathbf{c}, 0, 0, 0) \in X_\phi, \\ f((\mathbf{b}, 0, 0, 1), (\mathbf{a}, 0, 0, 0)) &= (\mathbf{c}, 0, 0, 1) \in X_\phi, \end{aligned}$$

which, by [\(7.18\)](#), implies that  $\mathbf{c} \in \text{Mod}(\chi) \cap \text{Mod}(\neg\chi)$ . Contradiction.  $\square$

The last claim deals with the case where we have both symmetric and non-symmetric components in  $(f_1, \dots, f_l)$ . It completely outlines the class of binary aggregators available for  $X_\phi$ . Notationally, if  $\mathbf{a} \in \{0, 1\}^n$  and  $I \subseteq \{1, \dots, n\}$ ,  $\mathbf{a}_I$  denotes the projection of  $\mathbf{a}$  to the coordinates in  $I$ .

**Claim 7.3.5.** *Assume  $F = (f_1, \dots, f_n)$  is a binary aggregator for  $X_\phi$ . Then, there exists a non-empty subset  $J \subseteq \{1, \dots, l\}$  such that  $f_j = \text{pr}_d^2$  for all  $j \in J \cup \{l+1, l+2, l+3\}$ ,  $d = 1, 2$ .*

*Proof.* By Claim [7.3.2](#), for some  $d \in \{1, 2\}$ ,  $f_{l+1} = f_{l+2} = f_{l+3} = \text{pr}_d^2$ . To obtain a contradiction, assume that  $f_j \neq \text{pr}_d^2$ , for all  $j \in \{1, \dots, l\}$ . Assume also w.l.o.g. that  $d = 2$ . The analogous arguments work for  $d = 1$ .

By Claims [7.3.3](#) and [7.3.4](#), there exists a partition  $(I, J)$  of  $\{1, \dots, l\}$ , such that  $f_i$  is symmetric for all  $i \in I$  and  $f_j = \text{pr}_1^2$  for all  $j \in J$ . To make things

easier to follow, assume w.l.o.g. that there exists an  $s \in \{1, \dots, l-1\}$  such that  $I = \{1, \dots, s\}$  and  $J = \{s+1, \dots, l\}$ . Let  $\mathbf{a} := (a_1, \dots, a_l) \in \text{Mod}(\chi)$ ,  $\mathbf{b} := (b_1, \dots, b_l) \in \text{Mod}(\neg\chi)$  and assume that:

$$(f_1, \dots, f_s)(\mathbf{a}_I, \mathbf{b}_I) := \mathbf{c}.$$

Then,  $(\mathbf{a}, 0, 0, 0)$  and  $(\mathbf{b}, 0, 0, 1) \in X_\phi$ . Since  $F$  is an aggregator for  $X_\phi$ , it must hold that:

$$F((\mathbf{a}, 0, 0, 0), (\mathbf{b}, 0, 0, 1)) = (\mathbf{c}, \mathbf{a}_J, 0, 0, 1) \in X_\phi,$$

which, by (7.18), implies that  $(\mathbf{c}, \mathbf{a}_J) \in \text{Mod}(\neg\chi)$ . Furthermore, again since  $F$  is an aggregator for  $X_\phi$ , it must be the case that:

$$F((\mathbf{c}, \mathbf{a}_J, 0, 0, 1), (\mathbf{a}, 0, 0, 0)) = (\mathbf{c}, \mathbf{a}_J, 0, 0, 0) \in X_\phi,$$

which, by (7.18), implies that  $(\mathbf{c}, \mathbf{a}_J) \in \text{Mod}(\chi)$ . Thus,  $(\mathbf{c}, \mathbf{a}_J) \in \text{Mod}(\chi) \cap \text{Mod}(\neg\chi)$ . Contradiction.  $\square$

By the above claims and Corollary 7.1.2,  $X_\phi$  is a possibility domain if and only if it admits a binary non-dictatorial aggregator such that there exists a  $d \in \{1, 2\}$  and a non-empty  $J \subseteq \{1, \dots, l\}$ , where, for all  $j \in J \cup \{l+1, l+2, l+3\}$ ,  $f_j = \text{pr}_d^2$ . It is not difficult to see that such an aggregator exists for  $d = 1$  if and only if it does for  $d = 2$ . Thus, we can safely assume that  $d = 1$ .

Before proving that our reduction works, we need some notation. For a domain  $Y \subseteq \{0, 1\}^n$  and a non-empty set of indices  $I \subseteq \{1, \dots, n\}$ , let  $Y_I$  be the projection of  $Y$  to the indices of  $I$ , that is, the set of partial vectors with coordinates in  $I$  that can be extended to vectors of  $Y$ .

First, assume that there exist  $p$  variables  $x_{i_1}, \dots, x_{i_p} \in \{x_1, \dots, x_l\}$ , where  $1 < p < l$ , such that  $\chi$  is independent from all of them. Let also  $J = \{1, \dots, l\} \setminus \{i_1, \dots, i_p\}$ . Then, (7.18) can be written as:

$$X_\phi \approx \{0, 1\}^p \times \left( \left( \text{Mod}(\chi)_J \times \{(0, 0, 0), (1, 1, 1)\} \right) \cup \left( \text{Mod}(\neg\chi)_J \times \text{Imp} \right) \right).$$

It is straightforward to observe that any  $n$ -tuple  $F = (f_1, \dots, f_n)$  of binary functions, such that  $f_{p+1} = \dots = f_n = \text{pr}_d^2$ ,  $d = 1, 2$  is an aggregator for  $X_\phi$ . Thus  $X_\phi$  is a possibility domain.

Now, assume that  $X_\phi$  is a possibility domain and  $F = (f_1, \dots, f_n)$  a binary non-dictatorial aggregator for  $X_\phi$ . Let also  $(I, J)$  be a partition of  $\{1, \dots, l\}$ , such that, for all  $j \in J \cup \{l+1, l+2, l+3\}$ ,  $f_j = \text{pr}_1^2$  and for all  $i \in I$ ,  $f_i \neq \text{pr}_1^2$ . Again, to simplify things, assume that there exists an  $s \in \{1, \dots, l-1\}$  such that  $I = \{1, \dots, s\}$ ,  $J = \{s+1, \dots, l\}$ . We consider the following three cases:

- If  $f_i := \text{pr}_2^2$ , for all  $i \in I$ , we show that  $\chi$  is independent from  $x_1, \dots, x_s$ . Suppose there exist vectors  $\mathbf{a}, \mathbf{b} \in \{0, 1\}^s$  and  $\mathbf{c} \in \{0, 1\}^{l-s}$ , such that  $(\mathbf{a}, \mathbf{c}) \in \text{Mod}(\chi)$  and  $(\mathbf{b}, \mathbf{c}) \in \text{Mod}(\neg\chi)$ . Then,  $(\mathbf{a}, \mathbf{c}, 0, 0, 0) \in X_\phi$  and  $(\mathbf{a}, \mathbf{c}, 0, 0, 1) \in X_\phi$ . Since  $F$  is an aggregator for  $X_\phi$ , it must hold that:

$$F((\mathbf{a}, \mathbf{c}, 0, 0, 0), (\mathbf{b}, \mathbf{c}, 0, 0, 1)) = (\mathbf{b}, \mathbf{c}, 0, 0, 0) \in X_\phi,$$

which, by (7.18), implies that  $(\mathbf{b}, \mathbf{c}) \in \text{Mod}(\chi) \cap \text{Mod}(\neg\chi)$ . Contradiction. Since  $\mathbf{a}, \mathbf{b}$  and  $\mathbf{c}$  where chosen arbitrarily, it follows that  $\chi$  is independent from  $x_1, \dots, x_s$ .

- If  $f_i$  is symmetric, for all  $i \in I$ , we show that  $\chi$  is independent from  $x_1, \dots, x_s$ . Suppose there exist vectors  $\mathbf{a}, \mathbf{b} \in \{0, 1\}^s$  and  $\mathbf{c} \in \{0, 1\}^{l-s}$ , such that  $(\mathbf{a}, \mathbf{c}) \in \text{Mod}(\chi)$  and  $(\mathbf{b}, \mathbf{c}) \in \text{Mod}(\neg\chi)$ . Also, assume that  $(f_1, \dots, f_s)(\mathbf{a}, \mathbf{b}) := \mathbf{z}$ . Then,  $(\mathbf{a}, \mathbf{c}, 0, 0, 0) \in X_\phi$  and  $(\mathbf{b}, \mathbf{c}, 0, 0, 1) \in X_\phi$ . Since  $F$  is an aggregator for  $X_\phi$ , it must hold that:

$$\begin{aligned} F((\mathbf{a}, \mathbf{c}, 0, 0, 0), (\mathbf{b}, \mathbf{c}, 0, 0, 1)) &= (\mathbf{z}, \mathbf{c}, 0, 0, 0) \in X_\phi, \\ F((\mathbf{b}, \mathbf{c}, 0, 0, 1), (\mathbf{a}, \mathbf{c}, 0, 0, 0)) &= (\mathbf{z}, \mathbf{c}, 0, 0, 1) \in X_\phi, \end{aligned}$$

which, by (7.18), implies that  $(\mathbf{z}, \mathbf{c}) \in \text{Mod}(\chi) \cap \text{Mod}(\neg\chi)$ . Contradiction. Since  $\mathbf{a}, \mathbf{b}$  and  $\mathbf{c}$  where chosen arbitrarily, it follows that  $\chi$  is independent from  $x_1, \dots, x_s$ .

- If there is a partition  $(I_1, I_2)$  of  $I$ , such that  $f_i = \text{pr}_2^2$  for all  $i \in I_1$  and  $f_i$  is symmetric for all  $i \in I_2$ , we show that that  $\chi$  is independent from all  $x_i$  such that  $i \in I_1$ . Assume again w.l.o.g. that there is a  $t \in \{1, \dots, s-1\}$  such that  $I_1 = \{1, \dots, t\}$  and  $I_2 = \{t+1, \dots, s\}$ . Suppose there exist vectors  $\mathbf{a}, \mathbf{b} \in \{0, 1\}^t$  and  $\mathbf{c} \in \{0, 1\}^{l-t}$ , such that  $(\mathbf{a}, \mathbf{c}) \in \text{Mod}(\chi)$  and  $(\mathbf{b}, \mathbf{c}) \in \text{Mod}(\neg\chi)$ . Then,  $(\mathbf{a}, \mathbf{c}, 0, 0, 0) \in X_\phi$  and  $(\mathbf{a}, \mathbf{c}, 0, 0, 1) \in X_\phi$ . Since  $F$  is an aggregator for  $X_\phi$ , it must hold that:

$$F((\mathbf{a}, \mathbf{c}, 0, 0, 0), (\mathbf{b}, \mathbf{c}, 0, 0, 1)) = (\mathbf{b}, \mathbf{c}, 0, 0, 0) \in X_\phi,$$

which, by (7.18), implies that  $(\mathbf{b}, \mathbf{c}) \in \text{Mod}(\chi) \cap \text{Mod}(\neg\chi)$ . Contradiction. Since  $\mathbf{a}, \mathbf{b}$  and  $\mathbf{c}$  were chosen arbitrarily, it follows that  $\phi$  is independent from  $x_1, \dots, x_t$ .

This concludes the proof of both the reduction and Theorem 7.3.8.  $\square$

Fortunately, the equivalent lower bound in the case of local possibility domains can be obtained quicker. Here the reduction is from the unsatisfiability problem.

**Theorem 7.3.9.** *Deciding, on input  $\phi$ , whether or not  $X_\phi$  admits a locally non-dictatorial aggregator is coNP-hard.*

*Proof.* We show that the problem of whether a logical formula  $\chi$ , defined on  $k$  variables  $x_1, \dots, x_l$ , is unsatisfiable, reduces to that of deciding if the truth set of a formula is a local possibility domain.

Let  $\psi(y_1, y_2, y_3)$  be the propositional formula with  $\text{Mod}(\psi) = \text{Imp}$ , where  $\{y_1, y_2, y_3\} \cap \{x_1, \dots, x_l\} = \emptyset$ . Consider the formula:

$$\phi = (\chi(x_1, \dots, x_l) \wedge \psi(y_1, y_2, y_3)) \vee (z \rightarrow w),$$

where  $z$  and  $w$  are variables not among those of  $\chi$  or  $\psi$ . First note that the length of  $\phi$  is again linear to that of  $\chi$ , since  $|\phi| = |\chi| + 8$  and thus the construction is polynomial.

Let  $n := l + 5$ . We will show that  $\chi$  is unsatisfiable if and only if

$$X_\phi = \left( \text{Mod}(\chi) \times \text{Imp} \times \{0, 1\}^2 \right) \cup \left( \{0, 1\}^{k+3} \times \{(0, 0), (0, 1), (1, 1)\} \right), \quad (7.19)$$

is a local possibility domain.

First, assume  $\chi$  is unsatisfiable. Then, (7.19):

$$X_\phi = \{0, 1\}^{l+3} \times \{(0, 0), (0, 1), (1, 1)\},$$

which is a local possibility domain, since it admits for example the binary aggregator  $F = (f_1, \dots, f_m)$ , where  $f_j = \wedge$ ,  $j = 1, \dots, m$ .

On the other hand, let  $\text{Mod}(\phi)$  be a local possibility domain, and assume  $\chi$  is satisfiable by some assignment  $\mathbf{a} = (a_1, \dots, a_l)$ . Since  $\text{Mod}(\phi)$  is a local possibility domain, by Theorem 7.1.4, it admits a ternary locally non-dictatorial aggregator  $F = (f_1, \dots, f_n)$ .



By Lemma 7.3.7,  $\text{Imp}$  is an impossibility domain. Thus, there exist  $\mathbf{b}^i = (b_1^i, b_2^i, b_3^i) \in \text{Imp}$ ,  $i = 1, 2, 3$ , such that:

$$\mathbf{c} := (f_{l+1}, f_{l+2}, f_{l+3})(\mathbf{b}^1, \mathbf{b}^2, \mathbf{b}^3) \notin \text{Imp}.$$

Since  $\mathbf{b}^1, \mathbf{b}^2, \mathbf{b}^3 \in \text{Imp}$ , it holds that  $(\mathbf{a}, \mathbf{b}^i, 1, 0) \in X_\phi$ , for  $i = 1, 2, 3$ . On the other hand:

$$F((\mathbf{a}, \mathbf{b}^1, 1, 0), (\mathbf{a}, \mathbf{b}^2, 1, 0), (\mathbf{a}, \mathbf{b}^3, 1, 0)) = (\mathbf{a}, \mathbf{c}, 1, 0) \notin X_\phi.$$

Thus  $F$  is not an aggregator for  $X_\phi$ . Contradiction.  $\square$

**Agendas** Suppose now that we have an agenda  $\bar{\phi} = (\phi_1, \dots, \phi_n)$ . We prove the following upper bounds to the complexity of deciding whether  $X_{\bar{\phi}}$  is a (local) possibility domain.

**Theorem 7.3.10.** *Given the agenda  $\bar{\phi} = (\phi_1, \dots, \phi_n)$ , the question whether  $X_{\bar{\phi}}$  admits a non-dictatorial aggregator is in  $\Delta_3^P$ .*

*Proof.* We will show that the problem can be decided in  $P$  with an oracle in  $\Sigma_2^P$ . By Corollary 7.1.2,  $X_{\bar{\phi}}$  is a possibility domain if and only if it is affine or it admits a binary non-dictatorial aggregator. For the former, note that it can be cast as:

For all  $m$ -tuples  $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \{0, 1\}^n$ , if it holds that:

$$\bigwedge_{j=1}^n \phi_j^{x_j}, \bigwedge_{j=1}^m \phi_j^{y_j} \text{ and } \bigwedge_{j=1}^n \phi_j^{z_j}$$

are all satisfiable, then so is  $\bigwedge_{j=1}^n \phi_j^{w_j}$ , where  $w_j = \oplus(x_j, y_j, z_j)$ ,  $j = 1, \dots, m$ .

Thus, it is in  $\Pi_2^P \subseteq \Delta_3^P$ .

It remains to show that the latter problem, or equivalently the problem of checking if  $H_{X_{\bar{\phi}}}$  is strongly connected, is in  $\Delta_3^P$ . Since  $H_{X_{\bar{\phi}}}$  has  $2n$  vertices, its size is polynomial to that of the agenda. Also, checking if a graph is strongly connected is in  $P$ . Thus it suffices to show that  $H_{X_{\bar{\phi}}}$  can be constructed within polynomial time with an oracle in  $\Sigma_2^P$ .

Consider two vertices  $uu'_s$  and  $vv'_t$  of  $H_{X_{\bar{\phi}}}$ , with  $s \neq t$ . To decide if there is an edge from  $uu'_s$  to  $vv'_t$ , it suffices to check the following:

There exist binary  $m$ -sequences  $\mathbf{x}, \mathbf{y}$  such that:

- both  $\bigwedge_{j=1}^n \phi_j^{x_j}$  and  $\bigwedge_{j=1}^n \phi_j^{y_j}$  are satisfiable,
- $x_s = u, x_t = v, y_s = u'$  and  $y_t = v'$  and
- for all  $n$ -sequences  $\mathbf{z}$ , either  $\bigwedge_{j=1}^n \phi_j^{z_j}$  is not satisfiable or at least one of the following is not true: (i)  $z_k = u$ , (ii)  $z_l = v'$ , (iii)  $z_j \in \{x_j, y_j\}$  for  $j = 1, \dots, n$ .

Notice that this can be done with an oracle in  $\Sigma_2^P$ . □

In what concerns local possibility domains, we have the following upper bound.

**Theorem 7.3.11.** *Given the agenda  $\bar{\phi} = (\phi_1, \dots, \phi_n)$ , deciding whether  $X_{\bar{\phi}}$  admits a locally non-dictatorial aggregator is in  $\Delta_3^P$ .*

*Proof.* We have already argued in Theorem 7.3.10 that  $H_{X_{\bar{\phi}}}$  can be constructed in polynomial time with an oracle in  $\Sigma_2^P$ . Then, we can obtain in polynomial time the aggregator  $G = (g_1, \dots, g_n)$  of Corollary 7.3.4 and testing whether  $X_{\bar{\phi}}$  admits it is in  $\Pi_2^P$  (in the same way we check if  $X_{\bar{\phi}}$  is affine). □

In terms of lower bounds, a straightforward idea would be to construct, given an integrity constraint, an agenda with the same domain, since that would immediately imply that the lower bounds for the integrity constraints carry on to the agendas. Unfortunately, as discussed above, this is an FNP-complete problem. However, in [83], given an integrity constraint  $\phi$ , Endriss et al. provide an agenda  $\bar{\phi}$ , of polynomial size to the length of  $\phi$ , such that  $X_{\bar{\phi}} = X_{\phi}$ . The reason this result does not imply the existence of a polynomial reduction, is that to construct  $\bar{\phi}$ , one needs a satisfying assignment of  $\phi$ . And of course, finding such an assignment is intractable. Fortunately, we can get past that in the problems we consider.

**Theorem 7.3.12.** *Given the agenda  $\bar{\phi} = (\phi_1, \dots, \phi_n)$ , the question whether  $X_{\bar{\phi}}$  admits a non-dictatorial aggregator is coNP-hard.*

*Proof.* Let  $\chi(x_1, \dots, x_k)$  be a propositional formula on  $k$  variables. We construct, in polynomial time, an agenda  $\bar{\phi}$  such that  $\chi$  is independent from at least one of its variables if and only if  $X_{\bar{\phi}}$  is a possibility domain. In all that follows,  $n = k + 3$ .

Let:

$$\phi(x_1, \dots, x_k, y_1, y_2, y_3) = \chi(x_1, \dots, x_k) \oplus \psi(y_1, y_2, y_3),$$

where:  $\{x_1, \dots, x_k\}$  and  $\{y_1, y_2, y_3\}$  are disjoint sets of variables and where  $\text{Mod}(\psi) = \text{Imp}$ . As in Theorem [7.3.8](#), we have that  $\phi$ 's length is linear to that of  $\chi$  and that:

$$X_\phi = \left( \text{Mod}(\chi) \times \{(0, 0, 0), (1, 1, 1)\} \right) \cup \left( \text{Mod}(\neg\chi) \times \text{Imp} \right). \quad (7.20)$$

Pick an arbitrary vector  $\mathbf{a} = (a_1, \dots, a_k) \in \{0, 1\}^k$  and set:

$$\mathbf{b} = (b_1, \dots, b_m) := \begin{cases} (a_1, \dots, a_k, 0, 0, 0) & \text{if } \chi(a_1, \dots, a_k) = 1, \\ (a_1, \dots, a_k, 0, 0, 1) & \text{else.} \end{cases}$$

In both cases,  $\mathbf{b}$  satisfies  $\phi$ . Thus, we can use Proposition 3 of [\[83\]](#), to construct an agenda  $\bar{\phi}$  such that  $X_{\bar{\phi}} = X_\phi$ , whose size is polynomial in the length of  $\phi$  and thus in that of  $\chi$  too. Since deciding whether  $\mathbf{a}$  satisfies  $\chi$  or not can be done in polynomial time, our construction is polynomial. Also, by Theorem [7.3.8](#),  $\chi$  is independent from at least one of its variables if and only if  $X_\phi$ , and thus  $X_{\bar{\phi}}$  too, is a possibility domain.  $\square$

The analogous arguments give us coNP-hardness in the case of local possibility domains.

**Theorem 7.3.13.** *Given the agenda  $\bar{\phi} = (\phi_1, \dots, \phi_n)$ , the question whether  $X_{\bar{\phi}}$  admits a locally non-dictatorial aggregator is coNP-hard.*

*Proof.* Let  $\chi(x_1, \dots, x_k)$  be a propositional formula. We construct an agenda  $\bar{\phi}$  such that  $X_{\bar{\phi}}$  is a local possibility domain if and only if  $\chi$  is unsatisfiable. Let  $\psi(y_1, y_2, y_3)$  be again the propositional formula with  $\text{Mod}(\psi) = \text{Imp}$ , where  $\{y_1, y_2, y_3\} \cap \{x_1, \dots, x_k\} = \emptyset$ . Consider the formula:

$$\phi = (\chi(x_1, \dots, x_k) \wedge \psi(y_1, y_2, y_3)) \vee (z \rightarrow w),$$

where  $z$  and  $w$  are variables not among those of  $\chi$  or  $\psi$ .

Let  $n := k + 5$ . By [\(7.19\)](#) we have that:

$$X_\phi = \left( \text{Mod}(\chi) \times \text{Imp} \times \{0, 1\}^2 \right) \cup \left( \{0, 1\}^{k+3} \times \{(0, 0), (0, 1), (1, 1)\} \right). \quad (7.21)$$

Again, pick an arbitrary vector  $\mathbf{a} = (a_1, \dots, a_k) \in \{0, 1\}^k$  and set:

$$\mathbf{b} = (b_1, \dots, b_n) := (a_1, \dots, a_k, 0, 0, 0, 0, 0).$$

$\mathbf{b}$  satisfies  $\phi$ . Thus, we can use Proposition 3 of [83], to construct an agenda  $\bar{\phi}$  such that  $X_{\bar{\phi}} = X_{\phi}$ , whose size is polynomial in the length of  $\phi$  and thus in that of  $\chi$  too. Also, by Theorem 7.3.9, we have that  $\chi$  is unsatisfiable if and only if  $X_{\phi}$ , and thus  $X_{\bar{\phi}}$  too, is a local possibility domain.  $\square$

A related result, that has been answered in [84], is whether the domain of an agenda admits the majority aggregator. In [174] and [84], such agendas are characterized as those satisfying the *median property*, that is, agendas whose every inconsistent subset, contains an inconsistent subset of size 2. Endriss et al. [84] show that checking if an agenda satisfies the median property is  $\Pi_2^P$ -complete. Unfortunately, this does not extend to the problem of determining if the domain of an agenda is a possibility domain, since, even though checking for the minority aggregator is in  $\Pi_2^P$ , the existence of binary non-dictatorial aggregators seems to be a harder problem.

### 7.3.5 Other types of non-dictatorial aggregation

In this subsection, we quickly extend the results of the previous subsections in four cases of non-dictatorial aggregation that have been used in the bibliography. Namely, we discuss *generalized dictatorships*, *anonymous*, *monotone* and *systematic* aggregators. We only consider the case where we search if an implicitly given Boolean domain admits such aggregators. The case where the domain is given explicitly has been shown to be tractable in [72], since the results there directly provide the required aggregators.

Recall that a domain  $X \subseteq \{0, 1\}^m$  admits an aggregator that is not a generalized dictatorship if and only if it is a possibility domain with at least three elements (Th. 7.2.9).

Using that, we can easily prove the same complexity bounds we had for deciding if the domain of an integrity constraint admits a non-dictatorial aggregator.

**Corollary 7.3.5.** *Deciding, on input  $\phi$ , whether  $X_{\phi}$  admits an aggregator that is not a generalized dictatorship is in  $\Sigma_2^P \cap \Pi_2^P$ .*

*Proof.* By Theorem 7.2.9, it suffices to show that deciding if  $X_{\phi}$  has at least three elements is in  $\Sigma_2^P \cap \Pi_2^P$ . Indeed, this can be written as:

There exist tuples  $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \{0, 1\}^n$  that are pairwise distinct, such that all three satisfy  $\phi$ .

Thus, deciding if  $|X_\phi| \geq 3$  is in  $\Sigma_1^P \subseteq \Sigma_2^P \cap \Pi_2^P$ . The rest of the proof is identical with that of Theorem [7.3.6](#).  $\square$

**Corollary 7.3.6.** *Deciding, on input  $\phi$ , whether  $X_\phi$  admits an aggregator that is not a generalized dictatorship is coNP-hard.*

*Proof.* Immediate by Theorems [7.2.9](#) and [7.3.8](#), since the domain of [\(7.18\)](#) has more than two elements.  $\square$

In case the domain is provided via an agenda, we can again easily obtain the same bounds.

**Corollary 7.3.7.** *Given the agenda  $\bar{\phi} = (\phi_1, \dots, \phi_n)$ , the question whether  $X_{\bar{\phi}}$  admits an aggregator that is not a generalized dictatorship is in  $\Delta_3^P$ .*

*Proof.* Again by Theorem [7.2.9](#), it suffices to show that deciding whether  $X_{\bar{\phi}}$  has at least three elements is in  $\Delta_3^P$ . Indeed, the problem can be written as:

There exist  $n$ -tuples  $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \{0, 1\}^n$  which are pairwise distinct and such that  $\bigwedge_{j=1}^n \phi_j^{x_j}$ ,  $\bigwedge_{j=1}^n \phi_j^{y_j}$  and  $\bigwedge_{j=1}^n \phi_j^{z_j}$  are all satisfiable.

Thus, it is in  $\Sigma_2^P \subseteq \Delta_3^P$ . The rest of the proof is identical that of Theorem [7.3.10](#).  $\square$

**Corollary 7.3.8.** *Given the agenda  $\bar{\phi} = (\phi_1, \dots, \phi_n)$ , the question whether  $X_{\bar{\phi}}$  admits an aggregator that is not a generalized dictatorship is coNP-hard.*

*Proof.* Immediate by Theorems [7.2.9](#) and [7.3.12](#), since the domain in [\(7.20\)](#) has more than two elements.  $\square$

It is immediate to observe that a ternary anonymous aggregator is always WNU. Thus, using Theorem [7.1.4](#), Kirousis et al. proved the following result.

**Corollary 7.3.9.** [\[144\]](#), *Corollary 5.11]  $X \subseteq \{0, 1\}^n$  is a local possibility domain if and only if it admits an anonymous aggregator.*

In fact, a local possibility domain always admits a ternary such aggregator. We can now obtain all the complexity bounds we have for local non-dictatorial aggregators, in the case where we search for anonymous ones.

**Corollary 7.3.10.** *A. Deciding, on input  $\phi$ , whether  $X_\phi$  admits an anonymous aggregator is i. in  $\Sigma_2^P \cap \Pi_2^P$  and ii. coNP-hard.*

*B. Given the agenda  $\bar{\phi} = (\phi_1, \dots, \phi_m)$ , the question whether  $X_{\bar{\phi}}$  admits an anonymous aggregator is i. in  $\Delta_3^P$  and ii. coNP-hard.*

*Proof.* Ai. Immediate by Corollary 7.3.9 and Theorem 7.3.7.

Aii. Immediate by Corollary 7.3.9 and Theorem 7.3.9.

Bi. Immediate by Corollary 7.3.9 and Theorem 7.3.11.

Bii. Immediate by Corollary 7.3.9 and Theorem 7.3.13. □

It is not difficult to see that all binary aggregators (both dictatorial and non-dictatorial) have that property. Also, in [75] and [144], it has been proven that if a domain admits a majority aggregator, it also admits a binary non-dictatorial one. Combining this with Theorem 7.1.2, we obtain the following result.

Recall that a domain  $X \subseteq \{0, 1\}^n$  admits a monotone non-dictatorial aggregator of some arity if and only if it admits a binary non-dictatorial one (Th. 7.2.10). Thus we can again easily obtain the required complexity bounds.

**Corollary 7.3.11.** *A. Deciding, on input  $\phi$ , whether  $X_\phi$  admits a monotone non-dictatorial aggregator is i. in  $\Sigma_2^P \cap \Pi_2^P$  and ii. coNP-hard.*

*B. Given the agenda  $\bar{\phi} = (\phi_1, \dots, \phi_n)$ , the question whether  $X_{\bar{\phi}}$  admits a monotone non-dictatorial aggregator is i. in  $\Delta_3^P$  and ii. coNP-hard.*

*Proof.* Ai. Immediate by Theorem 7.2.10 and Theorem 7.3.6.

Aii. Immediate by Theorem 7.2.10 and by noticing that the only available aggregators for the construction of (7.18) in Theorem 7.3.8 are binary non-dictatorial ones.

Bi. Immediate by Theorem 7.2.10 and Theorem 7.3.10.

Bii. Immediate by Theorem 7.2.10 and by noticing that the only available aggregators for the construction of (7.20) in Theorem 7.3.12 are binary non-dictatorial ones. □

**Corollary 7.3.12.** *Let  $X \subseteq \{0, 1\}^n$  be a Boolean domain. Then, either  $X$  admits only essentially unary functions, or it is closed under  $\wedge$ ,  $\vee$ ,  $\text{maj}$  or  $\oplus$ .*

This result can be obtained directly by Post's Lattice, without considering complexity theoretic notions. For a direct algebraic approach, see also [208, Proposition 1.12] (by noting that the only Boolean *semi-projections* of arity at least 3 are projections).

Corollary 7.3.12 translates in our framework as follows.

**Corollary 7.3.13.** *Let  $X \subseteq \{0, 1\}^n$  be a Boolean domain. Then  $X$  admits a systematic non-dictatorial aggregator if and only if it admits the aggregators  $\bar{\wedge}$ ,  $\bar{\vee}$ ,  $\bar{\text{maj}}$  or  $\bar{\oplus}$ .*

In case of integrity constraints, the problem of detecting if their domains admit systematic non-dictatorial aggregators is **coNP**-complete.

**Proposition 7.3.1.** *Deciding, on input  $\phi$ , whether  $X_\phi$  admits a systematic non-dictatorial aggregator is **coNP**-complete.*

*Proof.* In Theorem 7.3.6, we have already shown membership in **coNP** for detecting closure under  $\oplus$ . The proof for checking closure under  $\wedge$ ,  $\vee$ ,  $\text{maj}$  is essentially the same.

Thus we only need to show **coNP**-hardness. We reduce from the known **coNP**-complete problem of *tautology*, where we check if a propositional formula is satisfied by all assignments of values.

Let  $\chi(x_1, \dots, x_k)$  be the input propositional formula on  $k$  variables, and  $\psi(y_1, y_2, y_3)$  be the formula such that  $\text{Mod}(\psi) = \text{Imp}$ , where  $\{x_1, \dots, x_k\} \cap \{y_1, y_2, y_3\} = \emptyset$ . Let also  $m = k + 3$ .

Consider the formula:

$$\phi(x_1, \dots, x_k, y_1, y_2, y_3) = \chi \vee \psi.$$

If  $\chi$  is a tautology,  $X_\phi = \{0, 1\}^n$ , which is closed under  $\wedge$ ,  $\vee$ ,  $\text{maj}$  and  $\oplus$ . Otherwise:

$$X_\phi = \left( \text{Mod}(\chi) \times \{0, 1\}^3 \right) \cup \left( \text{Mod}(\neg\chi) \times \text{Imp} \right).$$

Easily now, if  $\mathbf{a} = (a_1, \dots, a_k)$  does not satisfy  $\phi$ , and  $f \in \{\wedge, \vee, \text{maj}, \oplus\}$ , it holds that  $(\mathbf{a}, 0, 0, 1)$ ,  $(\mathbf{a}, 0, 1, 0)$ ,  $(\mathbf{a}, 1, 0, 0) \in X_\phi$ , but:

$$\mathbf{b} := \bar{f}((\mathbf{a}, 1, 0, 0), (\mathbf{a}, 0, 1, 0), (\mathbf{a}, 1, 0, 0)) = \{(\mathbf{a}, 0, 0, 0), (\mathbf{a}, 1, 1, 1)\}.$$

Thus, since  $\mathbf{b} \notin X_\phi$ ,  $X_\phi$  does not admit any systematic non-dictatorial aggregator.  $\square$

Finally, we can obtain the corresponding results in the case the domain is provided via an agenda.

**Proposition 7.3.2.** *Deciding, on input  $\bar{\phi} = (\phi_1, \dots, \phi_n)$ , whether  $X_{\bar{\phi}}$  admits a systematic non-dictatorial aggregator is in  $\Pi_2^P$  and coNP-hard.*

*Proof.* In Theorem [7.3.11](#), we have already shown membership in  $\Pi_2^P$  for detecting closure under  $\oplus$ . The proof for checking closure under  $\wedge$ ,  $\vee$ , maj is essentially the same.

For coNP-hardness, observe that given a formula  $\chi$ , we can again set  $\phi = \chi \vee \psi$  and construct an agenda whose domain is the same with  $X_\phi$  of Proposition [7.3.1](#) in polynomial time, by [[83](#), Proposition 3]. Thus, the same reduction as in Proposition [7.3.1](#) works.  $\square$

**Semantic equivalence** We end this subsection with a sort discussion concerning formulas that are not (local) possibility integrity constraints, but are semantically equivalent to such formulas, in the sense that they have the same set of models.

In a, yet unpublished, extended version of [[143](#)], Kirousis et al. show that the problem of deciding whether the domain of a given formula is a possibility domain, is in  $\Sigma_2^P \cap \Pi_2^P$  and that of whether it is a local possibility domain, is in  $\Sigma_2^P$  and coNP-hard. Thus, we immediately obtain the following results.

**Corollary 7.3.14.** *Deciding, on input  $\phi$ , whether there exists a possibility integrity constraint  $\psi$  such that  $\text{Mod}(\phi) = \text{Mod}(\psi)$  is in  $\Sigma_2^P \cap \Pi_2^P$ .*

*Furthermore, deciding, on input  $\phi$ , whether there exists an lpic  $\psi$  such that  $\text{Mod}(\phi) = \text{Mod}(\psi)$  is in  $\Sigma_2^P$  and coNP-hard.*



# Conclusions

In this thesis, we studied CSP's, mainly from an algorithmic point of view. On one hand, we used the probabilistic method to devise randomized algorithms that guarantee the existence of solution to such problems, and find them in polynomial time. On the other, we used the algebraic toolkit to obtain results in the field of judgment aggregation.

Firstly, by translating the CSP framework as a set of events to be avoided over a common probability space, we devised efficient randomized algorithms that, given some assumptions on the dependencies and the probabilities of the events, prove the existence of and find a point in the probability space such that no event occurs. More specifically, we algorithmically proved lopsided versions of the symmetric and asymmetric Lovász Local Lemma and Shearer's Lemma. We worked in the variable framework of Moser and Tardos [171,172] which, although it is more restrictive than using general probability spaces, is broad enough to include a great variety of interesting applications and, furthermore, is easily amenable for algorithmic purposes.

Our inspiration came mainly from Moser's [171] RESAMPLE procedure that he used to prove the Lovász Local Lemma, along with the direct probabilistic approach that Giotis et al. [104,107] employed to analyze it. Specifically, we show how to express the probability that at least  $n$  such RESAMPLE procedures will be needed by a recurrence relation, which we subsequently solve by analytic means. In our view, this approach is useful both for theoretical purposes and applications. To demonstrate that, we first showed how to implement "moser-like" algorithms for sparser dependency graphs that appear in the bibliography. Then, we applied it to show that  $2\Delta - 1$  colors suffice to acyclically color the edges of a graph with maximum degree  $\Delta$ , thus obtaining the best known bound for the acyclic chromatic index of a graph  $G$  and, lastly, to construct  $c$ -separating codes, a useful type of codes whose explicit constructions are scarce in the bibliography.

In the future, this line of work could be extended in various ways. Firstly, in addition to acyclic edge coloring and separating codes, there are many combinatorial problems, like the satisfiability problem and vertex coloring problems, to which the LLL has already been fruitfully applied. Our direct probabilistic approach could possibly allow us to obtain better bounds for the parameters of these problems. Secondly, we have already argued that in the variable framework, Shearer's Lemma is not a necessary condition for avoiding all the undesirable events. It would certainly be interesting to use our approach in order to algorithmically prove versions of the LLL for even sparser (lopsi)dependency graphs, or even for a version of Shearer's lemma that provides an also necessary condition in the variable framework. Finally, apart from the LLL, there are various other probabilistic conditions that could take its place, like Chebyshev's Inequality [23,212], which could result in interesting algorithmic results.

Apart from the probabilistic approach, our work was also situated in aggregation theory and, specifically, in judgment aggregation. Here, we characterized various domains, both in terms of the aggregators they admit and syntactically, by the formulas (integrity constraints) that describe them. We also showed a dichotomy theorem for the complexity of the multi-sorted constraint satisfaction problem defined over a special class of possibility domains we named uniform possibility domains. Furthermore, we provided efficient algorithms that recognize if a domain admits various types of non-dictatorial aggregators. In case the domain is Boolean, we provided algorithms that construct an integrity constraint describing it and we also showed how to efficiently recognize integrity constraints of specific syntactic forms that give rise to interesting possibility domains. Finally, we showed upper and lower complexity bounds for the problems of deciding if an integrity constraint or an agenda give rise to possibility domains with various properties.

One possible aim is of course to attempt to make the aforementioned complexity bounds tight. Also, almost all our work is under the assumption of conservative aggregators. The equivalent results for aggregators that are simply Paretian would certainly be interesting. Finally, there are many voting rules across the bibliography that have not yet been expressed in the abstract framework of Dokow and Holzman [77]. It would be interesting to obtain characterizations for the domains that admit such voting rules.

# Bibliography

- [1] Dimitris Achlioptas and Fotis Iliopoulos. Untitled notes. *Unpublished private communication*, 2013.
- [2] Dimitris Achlioptas and Fotis Iliopoulos. Random walks that find perfect objects and the Lovász local lemma. *Journal of the ACM (JACM)*, 63(3):22, 2016.
- [3] Fotis Iliopoulos Achlioptas, Dimitris and Vladimir Kolmogorov. A local lemma for focused stochastic algorithms. *SIAM Journal on Computing*, 48(5):1583–1602, 2019.
- [4] A. V. Aho and J. D. Ullman. Universality of data retrieval languages. In *Proceedings of the 6th ACM SIGACT-SIGPLAN Symposium on Principles of programming languages – POPL ’79*, 1979.
- [5] Michael Albert, Alan Frieze, and Bruce Reed. Multicoloured Hamilton cycles. *The Electronic Journal of Combinatorics*, 2(1):R10, 1995.
- [6] Reginald BJT Allenby and Alan Slomson. *How to count: An introduction to combinatorics*. CRC Press, 2011.
- [7] Noga Alon. A parallel algorithmic version of the local lemma. *Random Structures & Algorithms*, 2(4):367–378, 1991.
- [8] Noga Alon and Joel H Spencer. *The probabilistic method*. John Wiley & Sons, 2016.
- [9] Noga Alon, Benny Sudakov, and Ayal Zaks. Acyclic edge colorings of graphs. *Journal of Graph Theory*, 37(3):157–167, 2001.
- [10] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

- [11] Kenneth J Arrow. *Social choice and individual values*. Wiley, New York, 1951.
- [12] Albert Atserias, Phokion G Kolaitis, and Simone Severini. Generalized satisfiability problems via operator assignments. *Journal of Computer and System Sciences*, 105:171–198, 2019.
- [13] Alexander Barg, G Robert Blakley, and Gregory A Kabatiansky. Digital fingerprinting codes: Problem statements, constructions, identification of traitors. *IEEE Transactions on Information Theory*, 49(4):852–865, 2003.
- [14] Libor Barto. The dichotomy for conservative constraint satisfaction problems revisited. In *Proc. 26th Annual IEEE Symp. on Logic in Computer Science*, pages 301–310, 2011.
- [15] Libor Barto. The collapse of the bounded width hierarchy. *Journal of Logic and Computation*, 26(3):923–943, 2014.
- [16] Libor Barto and Marcin Kozik. Constraint satisfaction problems of bounded width. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 595–603. IEEE, 2009.
- [17] Libor Barto and Marcin Kozik. New conditions for Taylor varieties and CSP. In *2010 25th Annual IEEE Symposium on Logic in Computer Science*, pages 100–109. IEEE, 2010.
- [18] József Beck. An algorithmic approach to the Lovász local lemma. I. *Random Structures & Algorithms*, 2(4):343–365, 1991.
- [19] Edward A Bender and L Bruce Richmond. A multivariate Lagrange inversion formula for asymptotic calculations. *The Electronic Journal of Combinatorics*, 5(1):33, 1998.
- [20] François Bergeron, Gilbert Labelle, and Pierre Leroux. *Combinatorial species and tree-like structures*, volume 67. Cambridge University Press, 1998.
- [21] Paul R. Berman, AD Scott, and M Karpinski. Approximation hardness and satisfiability of bounded occurrence instances of SAT. *ECCC*, 10(022), 2003.

- [22] Christian Bessiere, Clément Carbonnel, Emmanuel Hebrard, George Katsirelos, and Toby Walsh. Detecting and exploiting subproblem tractability. In *IJCAI*, pages 468–474, 2013.
- [23] Irénée-Jules Bienaymé. *Considérations à l'appui de la découverte de Laplace sur la loi de probabilité dans la méthode des moindres carrés*. Imprimerie de Mallet-Bachelier, 1853.
- [24] Rodrigo Bissacot, Roberto Fernández, Aldo Procacci, and Benedetto Scoppola. An improvement of the Lovász local lemma via cluster expansion. *Combinatorics, Probability and Computing*, 20(5):709–719, 2011.
- [25] Manuel Bodirsky. Complexity classification in infinite-domain constraint satisfaction. *arXiv preprint arXiv:1201.0856*, 2012.
- [26] Manuel Bodirsky and Jan Kára. The complexity of temporal constraint satisfaction problems. *Journal of the ACM (JACM)*, 57(2):1–41, 2010.
- [27] Manuel Bodirsky and Marcello Mamino. Constraint satisfaction problems over numeric domains. In *Dagstuhl Follow-Ups*, volume 7. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.
- [28] Manuel Bodirsky and Michael Pinsker. Schaefer’s theorem for graphs. *Journal of the ACM (JACM)*, 62(3):1–52, 2015.
- [29] VG Bodnarchuk, L Ao Kaluzhnin, Viktor N Kotov, and Boris A Romov. Galois theory for post algebras. I. *Cybernetics*, 5(3):243–252, 1969.
- [30] Elmar Böhler, Nadia Creignou, Steffen Reith, and Heribert Vollmer. Playing with boolean blocks, part I: Post’s lattice with applications to complexity theory. In *SIGACT News*. Citeseer, 2003.
- [31] Elmar Böhler, Nadia Creignou, Steffen Reith, and Heribert Vollmer. Playing with boolean blocks, part II: Constraint satisfaction problems. In *ACM SIGACT-Newsletter*. Citeseer, 2004.
- [32] Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D. Procaccia, editors. *Handbook of Computational Social Choice*. Cambridge University Press, 2016.

- [33] Andrei Bulatov and Víctor Dalmau. A simple algorithm for Mal'tsev constraints. *SIAM Journal on Computing*, 36(1):16–27, 2006.
- [34] Andrei Bulatov, Peter Jeavons, and Andrei Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM journal on computing*, 34(3):720–742, 2005.
- [35] Andrei A Bulatov. Tractable conservative constraint satisfaction problems. In *18th Annual IEEE Symposium of Logic in Computer Science, 2003. Proceedings.*, pages 321–330. IEEE, 2003.
- [36] Andrei A Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *Journal of the ACM (JACM)*, 53(1):66–120, 2006.
- [37] Andrei A Bulatov. Complexity of conservative constraint satisfaction problems. *ACM Transactions on Computational Logic (TOCL)*, 12(4):1–66, 2011.
- [38] Andrei A Bulatov. Conservative constraint satisfaction re-visited. *Journal of Computer and System Sciences*, 82(2):347–356, 2016.
- [39] Andrei A Bulatov and Peter Jeavons. Tractable constraints closed under a binary operation. In *Oxford University*. Citeseer, 2000.
- [40] Andrei A Bulatov and Peter Jeavons. An algebraic approach to multi-sorted constraints. In *Principles and Practice of Constraint Programming–CP 2003*, pages 183–198. Springer, 2003.
- [41] Andrei A Bulatov, Andrei A Krokhin, and Peter Jeavons. Constraint satisfaction problems and finite algebras. In *International Colloquium on Automata, Languages, and Programming*, pages 272–282. Springer, 2000.
- [42] Andrei A. Bulatov and Matthew Valeriote. Recent results on the algebraic approach to the CSP. In *Complexity of Constraints*, pages 68–92. Springer, 2008.
- [43] Hubie Chen Bulatov, Andrei and Victor Dalmau. Learnability of relatively quantified generalized formulas. In *International Conference on Algorithmic Learning Theory*. Springer, Berlin, Heidelberg, 2004.

- [44] Jin-Yi Cai and Aaron Gorenstein. Geiger’s theorem. *Complexity of Counting Problems: Lecture 9*, 2012.
- [45] Jin-Yi Cai and Chetan Rao. Universal algebra. *Complexity of Counting Problems: Lecture 8*, 2012.
- [46] Xing Shi Cai, Guillem Perarnau, Bruce Reed, and Adam Bene Watts. Acyclic edge colourings of graphs with large girth. *Random Structures & Algorithms*, 50(4):511–533, 2017.
- [47] Clément Carbonnel. The dichotomy for conservative constraint satisfaction is polynomially decidable. In *International Conference on Principles and Practice of Constraint Programming*, pages 130–146. Springer, 2016.
- [48] Clément Carbonnel. The meta-problem for conservative mal’tsev constraints. In *Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*, 2016.
- [49] Fabrizio Cariani, Marc Pauly, and Josh Snyder. Decision framing in judgment aggregation. *Synthese*, 163(1):1–24, 2008.
- [50] Hubie Chen and Victor Dalmau. (smart) look-ahead arc consistency and the pursuit of csp tractability. In *International Conference on Principles and Practice of Constraint Programming*, pages 182–196. Springer, 2004.
- [51] Hubie Chen and Benoit Larose. Asking the metaquestions in constraint tractability. *ACM Transactions on Computation Theory (TOCT)*, 9(3):1–27, 2017.
- [52] PM Cohn. *Universal Algebra*. Harper & Row, 1965.
- [53] Louis Comtet. *Advanced combinatorics, enlarged ed.*, d, 1974.
- [54] Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, 1971.
- [55] Martin C Cooper. An optimal k-consistency algorithm. *Artificial Intelligence*, 41(1):89–95, 1989.

- [56] Daniel W Cranston. Acyclic edge-coloring of planar graphs:  $\delta$  colors suffice when  $\delta$  is large. *SIAM Journal on Discrete Mathematics*, 33(2):614–628, 2019.
- [57] Nadia Creignou and J-J Hébrard. On generating all solutions of generalized satisfiability problems. *RAIRO-Theoretical Informatics and Applications*, 31(6):499–511, 1997.
- [58] Nadia Creignou and Miki Hermann. Complexity of generalized satisfiability counting problems. *Information and computation*, 125(1):1–12, 1996.
- [59] Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity classifications of boolean constraint satisfaction problems*. SIAM, 2001.
- [60] Nadia Creignou, Phokion G Kolaitis, and Heribert Vollmer. *Complexity of constraints: an overview of current research themes*. Springer Science & Business Media, 2008.
- [61] Nadia Creignou, Phokion G Kolaitis, and Bruno Zanuttini. Structure identification of boolean relations and plain bases for co-clones. *Journal of Computer and System Sciences*, 74(7):1103–1115, 2008.
- [62] Béla Csákány. All minimal clones on the three-element set. *Acta cybernetica*, 6(3):227–238, 1983.
- [63] Víctor Dalmau and Justin Pearson. Closure functions and width 1 problems. In *International Conference on Principles and Practice of Constraint Programming*, pages 159–173. Springer, 1999.
- [64] Brian A Davey and Hilary A Priestley. *Introduction to lattices and order*. Cambridge university press, 2002.
- [65] Nicolas De Condorcet et al. *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. Cambridge University Press, 2014.
- [66] Abraham De Moivre. *The doctrine of chances: or, a method of calculating the probability of events in play*. W. Pearson, 1718.
- [67] Rina Dechter. From local to global consistency. *Artificial intelligence*, 55(1):87–107, 1992.



- [68] Rina Dechter and Judea Pearl. Structure identification in relational data. *Artificial Intelligence*, 58(1-3):237–270, 1992.
- [69] Alvaro del Val. On 2-sat and renamable Horn. In *Proceedings of the National Conference on Artificial Intelligence*, pages 279–284. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2000.
- [70] D Deng, Douglas R Stinson, and Ruizhong Wei. The Lovász local lemma and its applications to some combinatorial arrays. *Designs, Codes and Cryptography*, 32(1-3):121–134, 2004.
- [71] Josep Díaz, Lefteris M. Kirousis, Sofia Kokonezi, and John Livieratos. Algorithmically efficient syntactic characterization of possibility domains. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece.*, pages 50:1–50:13, 2019.
- [72] Josep Díaz, Lefteris M. Kirousis, Sofia Kokonezi, and John Livieratos. Algorithmically efficient syntactic characterization of possibility domains. *Bulletin of the Hellenic Mathematical Society*, 63:97–135, 2019.
- [73] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.
- [74] Franz Dietrich and Christian List. Arrow’s theorem in judgment aggregation. *Social Choice and Welfare*, 29(1):19–33, 2007.
- [75] Elad Dokow and Ron Holzman. Aggregation of binary evaluations for truth-functional agendas. *Social Choice and Welfare*, 32(2):221–241, 2009.
- [76] Elad Dokow and Ron Holzman. Aggregation of binary evaluations. *Journal of Economic Theory*, 145(2):495–511, 2010.
- [77] Elad Dokow and Ron Holzman. Aggregation of non-binary evaluations. *Advances in Applied Mathematics*, 45(4):487–504, 2010.
- [78] William F Dowling and Jean H Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *The Journal of Logic Programming*, 1(3):267–284, 1984.

- [79] Ramez Elmasri and Sham Navathe. *Fundamentals of database systems*. Pearson London, 2016.
- [80] Herbert B. Enderton. *A mathematical introduction to logic*. Elsevier, 2001.
- [81] Ulle Endriss. Judgment aggregation. In Brandt et al. [32], pages 399–426.
- [82] Ulle Endriss and Ronald de Haan. Complexity of the winner determination problem in judgment aggregation: Kemeny, slater, tideman, young. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 117–125. International Foundation for Autonomous Agents and Multiagent Systems, 2015.
- [83] Ulle Endriss, Umberto Grandi, Ronald De Haan, and Jérôme Lang. Succinctness of languages for judgment aggregation. In *Fifteenth International Conference on the Principles of Knowledge Representation and Reasoning*, 2016.
- [84] Ulle Endriss, Umberto Grandi, and Daniele Porello. Complexity of judgment aggregation. *Journal of Artificial Intelligence Research*, pages 481–514, 2012.
- [85] Paul Erdős and László Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. *Infinite and finite sets*, 10:609–627, 1975.
- [86] Paul Erdős and Joel Spencer. Lopsided Lovász local lemma and Latin transversals. *Discrete Applied Mathematics*, 30(2-3):151–154, 1991.
- [87] Louis Esperet and Aline Parreau. Acyclic edge-coloring using entropy compression. *European Journal of Combinatorics*, 34(6):1019–1027, 2013.
- [88] L Euler. Letter to goldbach, dated september 4, 1751. published as “lettre CXL, Euler á Goldbach”.
- [89] Ronald Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In *Complexity of Computation*, ed. R. Karp, SIAM-AMS Proceedings, volume 7, pages 43–73, 1974.

- [90] Tomás Feder and Moshe Y Vardi. The computational structure of monotone monadic snp and constraint satisfaction: A study through datalog and group theory. *SIAM Journal on Computing*, 28(1):57–104, 1998.
- [91] Marcel Fernandez and John Livieratos. Algorithmic aspects on the construction of separating codes. In *International Symposium on Experimental Algorithms*, Springer, Cham, 2019.
- [92] Paula Fialho, Bernardo NB de Lima, and Aldo Procacci. A new bound on the acyclic edge chromatic index. *arXiv preprint arXiv:1912.04436*, 2019.
- [93] J Fiamcik. The acyclic chromatic class of a graph. *Math. Slovaca*, 28(2):139–145, 1978.
- [94] Philippe Flajolet and Robert Sedgewick. *Analytic combinatorics*. Cambridge University press, 2009.
- [95] G David Forney. *Concatenated codes*. Cambridge, Massachusetts: MIT Press, 1967.
- [96] Arnaldo Garcia and Henning Stichtenoth. A tower of artin - schreier extensions of function fields attaining the drinfeld - vlâdut bound. *Inventiones Mathematicae*, 121:211–222, 01 1995.
- [97] Arnaldo Garcia and Henning Stichtenoth. On the asymptotic behaviour of some towers of function fields over finite fields. *Journal of Number Theory*, 61(2):248 – 273, 1996.
- [98] William Gasarch and Haeupler Bernhard. Lower bounds on van der waerden numbers. *the electronic journal of combinatorics*, 16(R00), 2009.
- [99] John Geanakoplos. Three brief proofs of arrow’s impossibility theorem. *Economic Theory*, 26(1):211–215, 2005.
- [100] Heidi Gebauer, Robin A. Moser, Dominik Scheder, and Emo Welzl. The Lovász local lemma and satisfiability. In *Efficient Algorithms*, pages 30–54. Springer, 2009.

- [101] Heidi Gebauer, Tibor Szabó, and Gábor Tardos. The local lemma is tight for SAT. In *Proceedings 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 664–674. SIAM, 2011.
- [102] David Geiger. Closed systems of functions and predicates. *Pacific journal of mathematics*, 27(1):95–100, 1968.
- [103] Ira M Gessel. A combinatorial proof of the multivariable lagrange inversion formula. *Journal of Combinatorial Theory, Series A*, 45(2):178–195, 1987.
- [104] Ioannis Giotis, Lefteris Kirousis, Kostas I. Psaromiligkos, and Dimitrios M. Thilikos. On the algorithmic Lovász local lemma and acyclic edge coloring. In *Proceedings of the twelfth workshop on analytic algorithmics and combinatorics*. Society for Industrial and Applied Mathematics, 2015. Available: <http://epubs.siam.org/doi/pdf/10.1137/1.9781611973761.2>.
- [105] Ioannis Giotis, Lefteris Kirousis, Kostas I Psaromiligkos, and Dimitrios M Thilikos. Acyclic edge coloring through the Lovász local lemma. *Theoretical Computer Science*, 665:40–50, 2017.
- [106] Ioannis Giotis, Lefteris M. Kirousis, John Livieratos, Kostas I. Psaromiligkos, and Dimitrios M. Thilikos. Alternative proofs of the asymmetric Lovász local lemma and Shearer’s lemma. In *Proceedings of the 11th International Conference on Random and Exhaustive Generation of Combinatorial Structures, GASCom*, 2018. Available: <http://ceur-ws.org/Vol-2113/paper15.pdf>.
- [107] Ioannis Giotis, Lefteris M. Kirousis, Kostas I. Psaromiligkos, and Dimitrios M. Thilikos. An alternative proof for the constructive asymmetric Lovász local lemma. In *13th Cologne Twente Workshop on Graphs and Combinatorial Optimization*, 2015.
- [108] Martin Goldstern and Michael Pinsker. A survey of clones on infinite sets. *Algebra universalis*, 59:365–403, 2008.
- [109] V. D. Goppa. Codes on algebraic curves. *Sov. Math.-Dokl.*, 24:170–172, 1981.

- [110] Ian P Goulden and Devadatta M Kulkarni. Multivariable lagrange inversion, gessel-viennot cancellation, and the matrix tree theorem. *Journal of Combinatorial Theory, Series A*, 80(2):295–308, 1997.
- [111] Umberto Grandi and Ulle Endriss. Binary aggregation with integrity constraints. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 204, 2011.
- [112] Umberto Grandi and Ulle Endriss. Lifting integrity constraints in binary aggregation. *Artificial Intelligence*, 199:45–66, 2013.
- [113] Umberto Grandi, Ulle Endriss, et al. Lifting rationality assumptions in binary aggregation. In *AAAI*, 2010.
- [114] Davide Grossi and Gabriella Pigozzi. Judgment aggregation: a primer. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(2):1–151, 2014.
- [115] Grzegorz Gutowski, Jakub Kozik, and Xuding Zhu. Acyclic edge coloring using entropy compression. In *Abstracts of the 7th Polish Combinatorial Conference*, 2018. Available: <https://7pcc.tcs.uj.edu.pl/program.php>.
- [116] Mark Haiman and William Schmitt. Incidence algebra antipodes and lagrange inversion in one and several variables. *Journal of Combinatorial Theory, Series A*, 50(2):172–185, 1989.
- [117] Paul R. Halmos. *Introduction to Hilbert Space and the Theory of Spectral Multiplicity*. Benediction Classics, 2016.
- [118] David G Harris. Lopsidedependency in the Moser-Tardos framework: beyond the lopsided Lovász local lemma. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1792–1808. SIAM, 2015.
- [119] David G Harris and Aravind Srinivasan. A constructive algorithm for the Lovász local lemma on permutations. In *Proceedings 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 907–925. SIAM, 2014.

- [120] Nicholas Harvey and Christopher Liaw. Rainbow hamilton cycles and lopsidedependency. *Discrete Mathematics*, 340(6):1261–1270, 2017.
- [121] Nicholas JA Harvey and Jan Vondrák. An algorithmic proof of the Lovász local lemma via resampling oracles. In *Proceedings 56th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1327–1346. IEEE, 2015.
- [122] Nicholas JA Harvey and Jan Vondrák. Short proofs for generalizations of the Lovász local lemma: Shearer’s condition and cluster expansion. *arXiv preprint arXiv:1711.06797*, 2017.
- [123] Kun He, Liang Li, Xingwu Liu, Yuyi Wang, and Mingji Xia. Variable-version Lovász local lemma: Beyond shearer’s bound. In *58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 451–462. IEEE, 2017.
- [124] Peter Henrici. Applied and computational complex analysis. *Bull. Amer. Math. Soc*, 81:647–652, 1975.
- [125] Frederik S Herzberg. Universal algebra for general aggregation theory: Many-valued propositional-attitude aggregators as MV-homomorphisms. *Journal of Logic and Computation*, 25(3):965–977, 2013.
- [126] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 1990.
- [127] Paweł Idziak, Petar Marković, Ralph McKenzie, Matthew Valeriote, and Ross Willard. Tractability and learnability arising from algebras with few subpowers. *SIAM Journal on Computing*, 39(7):3023–3037, 2010.
- [128] Zverovich Igor’E. Characterizations of closed classes of boolean functions in terms of forbidden subfunctions and post classes. *Discrete Applied Mathematics*, 149(1-3):200–218, 2005.
- [129] Neil Immerman. *Descriptive complexity*. Graduate texts in computer science. Springer, 1999.

- [130] Peter Jeavons. On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, 200(1-2):185–204, 1998.
- [131] Peter Jeavons and David Cohen. An algebraic characterization of tractable constraints. In *International Computing and Combinatorics Conference*, pages 633–642. Springer, 1995.
- [132] Peter Jeavons, David Cohen, and Martin C Cooper. Constraints, consistency and closure. *Artificial Intelligence*, 101(1-2):251–265, 1998.
- [133] Peter Jeavons, David Cohen, and Marc Gyssens. Closure properties of constraints. *Journal of the ACM (JACM)*, 44(4):527–548, 1997.
- [134] Peter Jeavons, David Cohen, and Marc Gyssens. How to determine the expressive power of constraints. *Constraints*, 4(2):113–131, 1999.
- [135] Peter Jeavons, David Cohen, and Justin Pearson. Constraints and universal algebra. *Annals of Mathematics and Artificial Intelligence*, 24(1-4):51–67, 1998.
- [136] LA Kaluznin and R Pöschel. Funktionen-und relationenalgebren. *VEB Deutscher Verlag der Wissenschaften, Berlin*, 19:79, 1979.
- [137] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.
- [138] Sebastian Kerkhoff, Reinhard Pöschel, and Friedrich Martin Schneider. A short introduction to clones. *Electron. Notes Theor. Comput. Sci.*, 303:107–120, 2014.
- [139] Lefteris Kirousis and Phokion G. Kolaitis. The complexity of minimal satisfiability problems. *Information and Computation*, 187(1):20–39, 2003.
- [140] Lefteris Kirousis and Phokion G. Kolaitis. A dichotomy in the complexity of propositional circumscription. *Theory of Computing Systems*, 37(6):695–715, 2004.
- [141] Lefteris Kirousis, Phokion G Kolaitis, and John Livieratos. Aggregation of votes with multiple positions on each issue. In *Proceedings 16th International Conference on Relational and Algebraic Methods in*

- Computer Science*, pages 209–225. Springer, 2017. Expanded version to appear in *ACM Transactions on Economics and Computation*.
- [142] Lefteris Kirousis, Phokion G. Kolaitis, and John Livieratos. On the computational complexity of non-dictatorial aggregation. *arXiv preprint arXiv: 1711.01574*, 2017.
- [143] Lefteris Kirousis, Phokion G. Kolaitis, and John Livieratos. On the computational complexity of non-dictatorial aggregation. In *Proceedings 17th International Conference on Relational and Algebraic Methods in Computer Science*. Springer, 2018.
- [144] Lefteris Kirousis, Phokion G. Kolaitis, and John Livieratos. Aggregation of votes with multiple positions on each issue. *ACM Transactions on Economics and Computation (TEAC)*, 7(1):1, 2019.
- [145] Lefteris Kirousis and John Livieratos. A simple algorithmic proof of the symmetric lopsided Lovász local lemma. In *International Conference on Learning and Intelligent Optimization*, pages 49–63. Springer, 2018.
- [146] Lefteris Kirousis and John Livieratos. The acyclic chromatic index is less than the double of the max degree. *arXiv preprint arXiv:1901.07856*, 2019.
- [147] Lefteris Kirousis, John Livieratos, and Kostas I. Psaromiligkos. Directed Lovász local lemma and Shearer’s lemma. *Annals of Mathematics and Artificial Intelligence*, 88(1-3):133–155, 2020.
- [148] Jon Kleinberg and Eva Tardos. *Algorithm Design: Pearson New International Edition*. Pearson Higher Ed, 2013.
- [149] Phokion Kolaitis and Moshe Vardi. The decision problem for the probabilities of higher-order properties. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 425–435, 1987.
- [150] Kashyap Kolipaka, Babu Rao, and Mario Szegedy. Moser and Tardos meet Lovász. In *Proceedings 43rd Annual ACM symposium on Theory of Computing (STOC)*, pages 235–244. ACM, 2011.
- [151] János Körner and Gábor Simonyi. Separating partition systems and locally different sequences. *SIAM journal on discrete mathematics*, 1(3):355–359, 1988.



- [152] Lewis A Kornhauser and Lawrence G Sager. The one and the many: Adjudication in collegial courts. *Calif. L. Rev.*, 81:1, 1993.
- [153] Melven R Krom. The decision problem for a class of first-order formulas in which all disjunctions are binary. *Mathematical Logic Quarterly*, 13(1-2):15–20, 1967.
- [154] Gábor Kun and Mario Szegedy. A new line of attack on the dichotomy conjecture. *European Journal of Combinatorics*, 52:338–367, 2016.
- [155] Richard E Ladner. On the structure of polynomial time reducibility. *Journal of the ACM (JACM)*, 22(1):155–171, 1975.
- [156] Jérôme Lang, Paolo Liberatore, and Pierre Marquis. Conditional independence in propositional logic. *Artificial Intelligence*, 141(1-2):79–121, 2002.
- [157] Jérôme Lang, Paolo Liberatore, and Pierre Marquis. Propositional independence-formula-variable independence and forgetting. *Journal of Artificial Intelligence Research*, 18:391–443, 2003.
- [158] Jérôme Lang, Marija Slavkovic, and Srdjan Vesic. Agenda separability in judgment aggregation. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.
- [159] Benoit Larose and László Zádori. Bounded width problems and algebras. *Algebra universalis*, 56(3-4):439–466, 2007.
- [160] Harry R Lewis. Renaming a set of clauses as a Horn set. *Journal of the ACM (JACM)*, 25(1):134–135, 1978.
- [161] Leonid Libkin. *Elements of Finite Model Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004.
- [162] Christian List. The theory of judgment aggregation: An introductory review. *Synthese*, 187(1):179–207, 2012.
- [163] Christian List and Philip Pettit. Aggregating sets of judgments: An impossibility result. *Economics & Philosophy*, 18(1):89–110, 2002.
- [164] Christian List and Philip Pettit. Aggregating sets of judgments: Two impossibility results compared. *Synthese*, 140(1-2):207–235, 2004.

- [165] Christian List and Clemens Puppe. Judgment aggregation. In Paul Anand, Prasanta Pattanaik, and Clemens Puppe, editors, *The Handbook of Rational and Social Choice*, pages 457–482. Oxford University Press, 2009.
- [166] Christian List and Clemens Puppe. Judgment aggregation: A survey. In Christian List and Clemens Puppe, editors, *Handbook of Rational and Social Choice*. Oxford University Press, 2009.
- [167] H Machida and IG Rosenberg. Classifying essentially minimal clones. In *Proceeding 14th Internat. Symp. on Multiple-Valued Logic, (Winnipeg, 1984)*, pages 4–7, 1984.
- [168] Florence Jessie MacWilliams and Neil James Alexander Sloane. *The theory of error-correcting codes*, volume 16. Elsevier, 1977.
- [169] Miklós Maróti and Ralph McKenzie. Existence theorems for weakly symmetric operations. *Algebra universalis*, 59(3-4):463–489, 2008.
- [170] Jose Moreira, Marcel Fernández, and Grigory Kabatiansky. Constructions of almost secure frameproof codes with applications to fingerprinting schemes. *Designs, Codes and Cryptography*, 86, 04 2017.
- [171] Robin A. Moser. A constructive proof of the Lovász local lemma. In *Proceedings 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 343–350. ACM, 2009.
- [172] Robin A. Moser and Gábor Tardos. A constructive proof of the general Lovász local lemma. *Journal of the ACM (JACM)*, 57(2):11, 2010.
- [173] Klaus Nehring and Clemens Puppe. Strategy-proof social choice on single-peaked domains: Possibility, impossibility and the space between, 2002. University of California at Davis; available at: <http://vw11.ets.kit.edu/puppe.php>.
- [174] Klaus Nehring and Clemens Puppe. The structure of strategy-proof social choice—part I: General characterization and possibility results on median spaces. *Journal of Economic Theory*, 135(1):269–305, 2007.
- [175] Klaus Nehring and Clemens Puppe. Abstract arrowian aggregation. *Journal of Economic Theory*, 145(2):467–494, 2010.

- [176] Jaroslav Nešetřil and Nicholas C Wormald. The acyclic edge chromatic number of a random  $d$ -regular graph is  $d+1$ . *Journal of Graph Theory*, 49(1):69–74, 2005.
- [177] Christos H Papadimitriou. *Computational complexity*. John Wiley and Sons Ltd., 2003.
- [178] Christos H Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *Journal of computer and system sciences*, 43(3):425–440, 1991.
- [179] Marc Pauly and Martin Van Hees. Logical constraints on judgement aggregation. *Journal of Philosophical logic*, 35(6):569–585, 2006.
- [180] Wesley Pegden. An extension of the Moser–Tardos algorithmic local lemma. *SIAM Journal on Discrete Mathematics*, 28(2):911–917, 2014.
- [181] Francis Jeffry Pelletier and Norman M. Martin. Post’s functional completeness theorem. *Notre Dame Journal of Formal Logic*, 31(2):462–475, 1990.
- [182] Philip Pettit and Wlodek Rabinowicz. Deliberative democracy and the discursive dilemma. *Philosophical Issues*, 11:268–299, 2001.
- [183] Gabriella Pigozzi. Belief merging and the discursive dilemma: an argument-based account to paradoxes of judgment aggregation. *Synthese*, 152(2):285–298, 2006.
- [184] Reinhard Pöschel. Concrete representation of algebraic structures and a general galois theory. *Contributions to general algebra*, 1:249–272, 1979.
- [185] Reinhard Pöschel. A general galois theory for operations and relations and concrete characterization of related algebraic structures, 1980.
- [186] Emil Leon Post. *The two-valued iterative systems of mathematical logic*. Number 5 in Annals of Mathematics Studies. Princeton University Press, 1941.
- [187] Willard V Quine. On cores and prime implicants of truth functions. *The American Mathematical Monthly*, 66(9):755–760, 1959.

- [188] Ivo Rosenberg. *Über die funktionale Vollständigkeit in den mehrwertigen Logiken*. Academia, 1970.
- [189] Ivo G Rosenberg. Minimal clones i: the five types. In *Lectures in universal algebra*, pages 405–427. Elsevier, 1986.
- [190] Yurii L’vovich Sagalovich. “separating systems”. *Problems Inform. Transmission*, 30(2):105–123, 1994.
- [191] Kaushik Sarkar and Charles J Colbourn. Upper bounds on the size of covering arrays. *SIAM Journal on Discrete Mathematics*, 31(2):1277–1293, 2017.
- [192] Thomas J. Schaefer. The complexity of satisfiability problems. In *Proc. of the 10th Annual ACM Symp. on Theory of Computing*, pages 216–226, 1978.
- [193] Robert Sedgewick and Philippe Flajolet. *An introduction to the analysis of algorithms*. Pearson Education India, 2013.
- [194] Claude E. Shannon. A mathematical theory of communication. *The Bell system technical journal*, 24(3):379–423, 1948.
- [195] Micha Sharir. A strong-connectivity algorithm and its applications in data flow analysis. *Computers & Mathematics with Applications*, 7(1):67–72, 1981.
- [196] James B. Shearer. On a problem of Spencer. *Combinatorica*, 5(3):241–245, 1985.
- [197] Kenneth Shum, Ilya Aleshnikov, P. Kumar, Henning Stichtenoth, and Vinay Deolalikar. A low-complexity algorithm for the construction of algebraic-geometric codes better than the gilbert-varshamov bound. *Information Theory, IEEE Transactions on*, 47:2225 – 2241, 10 2001.
- [198] Mark H Siggers. A strong Mal’cev condition for varieties omitting unary type. *Algebra Universalis*, 64(1):15–20, 2010.
- [199] Michael Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 2012.

- [200] Joel Spencer. Ten lectures on the probabilistic method. In *SIAM*, volume 64, 1994.
- [201] Joel Spencer. Robin Moser makes Lovász local lemma algorithmic!, 2010.
- [202] Aravind Srivinsan. Improved algorithmic versions of the Lovász local lemma. In *Proceedings 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 611–620. SIAM, 2008.
- [203] Jessica N Staddon, Douglas R Stinson, and Ruizhong Wei. Combinatorial properties of frameproof and traceability codes. *IEEE transactions on information theory*, 47(3):1042–1049, 2001.
- [204] Larry J Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976.
- [205] Zoltán Szabó. An application of Lovász local lemma—a new lower bound for the van der Waerden number. *Random Structures & Algorithms*, 1(3):343–360, 1990.
- [206] Mario Szegedy. The Lovász local lemma—a survey. In *International Computer Science Symposium in Russia*, pages 1–11. Springer, 2013.
- [207] Mario Szegedy and Yixin Xu. Impossibility theorems and the universal algebraic toolkit. *CoRR*, abs/1506.01315, 2015.
- [208] Ágnes Szendrei. *Clones in universal algebra*, volume 99. Presses de l’Université de Montréal, 1986.
- [209] Terence Tao. Moser’s entropy compression argument, 2009. Available: <https://terrytao.wordpress.com/2009/08/05/mosers-entropy-compression-argument/>.
- [210] Terence Tao. Arrow’s theorem, 2012. <https://www.math.ucla.edu/~tao/arrow.pdf>, UCLA.
- [211] Robert Endre Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.
- [212] P. Tchebichef. Des valeurs moyennes. *Journal de Mathématiques Pures et Appliquées*, 2(12):177–184, 1867.

- [213] Zoi Terzopoulou, Ulle Endriss, and Ronald de Haan. Aggregating incomplete judgments: Axiomatisations for scoring rules. In *Proceedings of the 7th international workshop on computational social choice (COMSOC)*, 2018.
- [214] Boris A Trakhtenbrot. A survey of Russian approaches to perebor (brute-force searches) algorithms. *Annals of the History of Computing*, 6(4):384–400, 1984.
- [215] Vadim G Vizing. Critical graphs with given chromatic class. *Metody Discret. Analiz.*, 5:9–17, 1965.
- [216] Tao Wang and Yaqiong Zhang. Further result on acyclic chromatic index of planar graphs. *Discrete Applied Mathematics*, 201:228–247, 2016.
- [217] Robert Wilson. On the theory of aggregation. *Journal of Economic Theory*, 10(1):89–99, 1975.
- [218] Susumu Yamasaki and Shuji Doshita. The satisfiability problem for a class consisting of Horn sentences and some non-Horn sentences in propositional logic. *Information and Control*, 59(1-3):1–12, 1983.
- [219] Bruno Zanuttini and Jean-Jacques Hébrard. A unified framework for structure identification. *Information Processing Letters*, 81(6):335–339, 2002.