Automated Search for Impossibility Theorems in Choice Theory: Ranking Sets of Objects

MSc Thesis (*Afstudeerscriptie*)

written by

Christian Geist
(born 24 September 1983 in Dieburg, Germany)

under the supervision of **Ulle Endriss**, and submitted to the Board of Examiners in partial fulfillment of the requirements for the degree of

MSc in Logic

at the *Universiteit van Amsterdam.*

Date of the public defense:	**Members of the Thesis Committee:**
2 July 2010	Frank Veltman (chair)
	Krzysztof Apt
	Ulle Endriss
	Umberto Grandi
	Robert van Rooij

INSTITUTE FOR LOGIC, LANGUAGE AND COMPUTATION

Acknowledgements

I would like to express my sincere gratuity to all the individuals and institutions without whom this thesis project would not have been possible.

In particular, I would like to thank my supervisor, Ulle Endriss, for his support and guidance during all phases of this thesis project. It is also him who was responsible for directing my interest to the matter of computational social choice by teaching an extraordinary course on this subject in 2009, and who shaped the initial direction of my project. Ulle has been approachable at any time and always offered clear and more than helpful advice, from which I benefitted significantly.

I am also indebted to my thesis committee, consisting of Krzysztof Apt, Ulle Endriss, Umberto Grandi, Robert van Rooij, and Frank Veltman, for their valuable questions and comments during my thesis defense.

I would further like to express my gratitude and appreciation to my friends and colleagues, in particular Melanie Görner, Umberto Grandi, and Charlotte Vlek, not only for the many helpful discussions that substantially contributed to this thesis, but also for their moral support.

Nuffic, Siemens Management Consulting, the German National Academic Foundation, and the ILLC have generously provided financial assistance for my project and studies, which I gratefully acknowledge.

Amsterdam, July 2010 *Christian Geist*

Abstract

In the subarea of (social) choice theory commonly referred to as *ranking sets of objects* the question arises whether, given preferences over some domain, there is a preference relation on the power set of this domain that is compatible with certain axioms. The Kannai-Peleg Theorem (*Journal of Economic Theory*, 1984) gives a negative answer to this question for the case of six or more elements in the domain in combination with the two (very intuitive) axioms called *dominance* and *independence*. It is the main (and earliest) impossibility theorem in this particular area.

Our initial goal was to find a suitable formulation of the Kannai-Peleg Theorem in logic that would facilitate an automatic verification of the result. It turned out, however, that developing a first-order formulation of this problem and feeding it to a first-order theorem prover was not very effective. Therefore, we successfully tried an inductive proof consisting of a manual inductive step together with a formalization of the base case in propositional logic (in order to use a SAT solver). This particular way of using automated theorem proving in social choice theory is due to Tang and Lin (*Artificial Intelligence Journal*, 2009), who have in this way proved major impossibility results, like Arrow's Theorem, the Muller-Satterthwaite Theorem, and Sen's Theorem. We extended their technique to be able to treat the Kannai-Peleg Theorem and were able to verify it with a verification time of less than ten seconds.

With our initial objective met, we then developed a further extension of the method in order to discover new impossibility results. Using tools from model theory, a universal form of the previous inductive step (now applicable to a large class of axioms) allows for a fully automated theorem search, which has produced 84 impossibility theorems on a space of 20 axioms from the literature. Many of these results are variations of known impossibilities, but also a few new results were obtained. Interestingly, one of the impossibility theorems found by our program had even wrongly been published as a *possibility* result earlier (*Economic Theory*, 2000, 2003). Finally, we give some manual proofs of the new results to underline the fruitfulness of this computer-aided method of searching for impossibility results in the field of *ranking sets of objects*.

Contents

Introduction

"Our long term goal is to automate the discovery of theorems in social choice theory, game theory, and others."
TANG and LIN, 2009 [73]

Intrigued by TANG and LIN's vision and their novel approach to proving impossibility theorems in voting theory by employing computer-aided induction, we extended this technique to the area of *ranking sets of objects*. Based on this extension, we developed an automated and exhaustive theorem search relying on a model-theoretic result reducing impossibility theorems to small instances. Whereas TANG and LIN have used separate reduction theorems for each impossibility, our result is universally applicable to a whole class of axioms.

This thesis combines a report on the design of the search method and the utilized techniques with an overview of its initial results, comprising a set of 84 automatically discovered or verified impossibility theorems.

In order to shed some more light on the above concepts and ideas it is beneficial to put them in a broader context, which we are going to do in the next section.

Motivation and Area of Research

Our work addresses two general questions of wide interest, each of which corresponds to a whole stream of ideas and disciplines:

- How can we use computers to automatize theorem proving and theorem discovery?
- What are the principles governing a "rational" decision?

The first question concerns the methods applied in this thesis, the second stands for the actual subject matter we treated with those methods.

Let us concentrate on the subject matter first. The extensive field of *economic theory* is concerned with "rational decisions" in a variety of different ways. In the area of *decision theory* [53] mostly normative theories of rationality are developed, whereas *(rational) choice theory* [2] tries to model social and economic behaviour. Its subfield *social choice theory* [7, 27] is concerned with collective decision making, covering ideas like voting, preference and judgement aggregation, resource allocation, and others. In all these areas of *mathematical economics* it can be necessary to have a model for how preferences over single objects are related to preferences over sets of these objects when assuming certain principles of rationality. This is exactly what the discipline of *ranking sets of objects* (which can be considered a subfield of choice theory) is interested in, and even though its framework and the questions to be asked are quite simple, its results are powerful and non-trivial. For an extensive overview of the field and its results, see the survey by Barberà, Bossert and Pattanaik [11].

But what is the nature of the results we can expect from the above areas of economic theory? The two most common categories of results are *impossibility theorems* and *characterization results*. The latter say that certain principles fully specify a particular method, feature, or class of these. The former, on the other hand, express that certain (natural) principles are incompatible and can therefore not be used in combination. The most prominent representative of this class of results is Arrow's Impossibility Theorem [5], a theorem due to Kenneth Arrow, Nobel Prize laureate in economics.[1] Intuitively, it states that three desirable principles for preference aggregation cannot be satisfied by any aggregation rule.

Also in the field of *ranking sets of objects* such impossibility theorems exist. But before we can comprehend them, we have to understand the main concern of this discipline: given some preferences of an agent over individual objects, what can "rationally" be inferred about her preferences over sets of these objects? Put in a very small example, what does the preference of x over y tell us for the ranking of the respective sets $\{x\}$, $\{x, y\}$, and $\{y\}$? Aiming at answering these questions, the literature has brought forward many different principles of how to extend preferences from elements to sets. An example is the idea of dominance: adding a preferred alternative to a set makes the set better, and, similarly, adding a disliked alternative makes it worse. In the small setting above, this means that $\{x\}$ has to be preferred to $\{x, y\}$, which in turn is considered better than $\{y\}$.

Sometimes, however, even very natural combinations of such principles turn out to be incompatible (in the sense that no ranking of sets satisfies them). Most prominent among the few known impossibilities in this field is the Kannai-Peleg Theorem [37], which was proved in 1984 and flags the highly intuitive combination of dominance and an independence principle as inconsistent. The classical method employed for deriving such impossibilities in (social) choice theory is the so-called *axiomatic method*, which describes the use of formal axioms in order to precisely capture intuitive ideas and then to explore their logical consequences. Even though the axioms are usually not

[1] Further examples are Sen's Theorem [64] and the Gibbard-Satterthwaite Theorem [31, 62], both regarding the theory of voting.

specified in a particular logic, but rather in a semi-formal language, the gap towards a completely formal description within a certain logic is rather small.

This brings us directly to the second general question stated above. Since axiomatic descriptions can quickly be transferred into a completely formal and computer-readable format, it makes particular sense to apply methods from automated reasoning in order to verify and discover such theorems. It should also be noted that the proofs of impossibilities are often quite technical and little insightful, such that a computer-aided approach is even more motivated.

Automated reasoning [55] is the subfield of *artificial intelligence* [57] that is concerned with such theorem verifications and automated as well as interactive proving. At least since the automated discovery of a proof for the Four Colour Theorem by Appel and Haken [3] in the 1970s, automated theorem proving has demonstrated to be strong enough for attacking open problems in mathematics and related fields.

The popularity of applying automated reasoning for the verification and discovery of results can be explained by different factors. We have already mentioned domain-specific factors, like the axiomatic method and the prevalence of technicalities in proofs, for the case of (social) choice theory. On a more general level, reliability, speed and clarity are the main reasons for choosing a computer-aided approach.

When successfully applied as a verification tool, automated reasoning can improve the reliability of results. Even though peer-reviewing does help to keep the publication of wrong results to a minimum, time and again we can find examples of published work that later on turns out to have been flawed. A prominent example in the area of social choice theory is Arrow's Theorem [5], the first proof of which was flawed (as noted by Blau [14]) and corrected later by Arrow himself [6]. Also in the context of *ranking sets of objects* a result by Bossert, Pattanaik and Xu [16] needed a correction by Arlegi [4], which was confirmed in our theorem search.

The advantage in the speed of theorem discovery derives from at least two features of machines: on the one hand, computers are very good at handling technicalities that otherwise take very long to be handled manually. As an example consider computer algebra systems that can calculate even complex derivatives or solve (systems of) equations within seconds. On the other hand, even large sets of variants of a given task can efficiently be carried out on a computer. That we were able to check more than eight million problem instances as part of this thesis clearly supports this claim.

That clarity and gaining additional insights could be factors supporting automated reasoning might at first appear a bit surprising. Creating a formalization of a given setting, however, promotes a deeper understanding since even minor details have to be made explicit. Furthermore, being able to use automated methods for quickly checking slight modifications of a result on a computer helps researchers to better comprehend the exact nature of assumptions and theorems, and computer-aided proving additionally focusses their attention to results instead of technicalities in the proofs.

In general, many different approaches to applying automated reasoning are possible – from employing different logics for the formalization, to the question whether proof verification or theorem discovery is the ultimate goal. In this thesis, we first observe that a first-order formalization of the framework of *ranking sets of objects* is not well-suited for automated theorem proving. Then we describe our automated theorem search method, which relies on a universal step, manually-proven with tools from *model theory*[2] and reducing theorems to small instances for a large class of axioms. The small instances (base cases) are then checked using a SAT solver[3] on a computer-generated formalization in propositional logic.

Our results show that this new technique can reliably guide or even autonomously perform the discovery of unknown impossibility theorems. In this way, it is possible to comfortably explore the borderline between the possible and the impossible.

We thus see our work at the interface of *automated reasoning* and *choice theory*. It could therefore also be considered part of the relatively young area of *computational social choice* [19], which comprises the application of ideas from computer science to social choice theory on the one hand, and importing concepts from social choice theory into computing and artificial intelligence on the other hand.

In the following section of this introduction we give a literature review for *ranking sets of objects* and the application of *automated reasoning* to *(social) choice theory* and *game theory*. Afterwards we concentrate on our own contribution to the field and finally complete the introduction with an overview of the thesis.

Related Work

Ranking Sets of Objects

A vast body of literature has developed in the field of *ranking sets of objects*. Contributions are concerned with the framework itself, reasonable principles for different interpretations of sets, and its various applications in voting, matching and others. Since a complete overview of these lines of work would exceed the scope of this introduction, the reader is referred to the previously mentioned extensive survey by Barberà, Bossert and Pattanaik [11]. The

[2] The discipline of *Model theory* (situated within the field of *mathematical logic*) formally analyzes and classifies mathematical structures in order to understand which formulas are true in these.

[3] A SAT ($\cong$ Boolean satisfiability problem) solver is a program that can check whether a formula in *propositional logic* has a *satisfying assignment*.

different interpretations of sets as uncertain outcomes, opportunities, or collective outcomes will be discussed in Section 1.2. Here, we are going to focus on a review of the work that has lead to impossibility results in the field.

We have already touched upon the Kannai-Peleg Theorem, the most recognized impossibility in *ranking sets of objects*. The corresponding seminal paper by Kannai and Peleg is usually said to have started the whole field by first treating the problem of extending an order to sets of objects in its own right. Also note that it immediately triggered an array of responses and further contributions in the same year it was published [10, 12, 26, 35, 49, 51].

One of these responses by Barberà and Pattanaik [12] contains a second impossibility related to the original theorem. In this result the principle of *independence* was strengthened to a strict form, whereas dominance was weakened to apply to small sets only.

But also more recently, further impossibility results have been added. Puppe demonstrated in 1995 that the principle of *preference for freedom* is incompatible with the assumption of continuous preferences (which mostly plays a role for infinite sets of alternatives). Another impossibility, lowering the minimal number of elements in the Kannai-Peleg Theorem by adding the axiom of *neutrality* [49, 51], was proven in the aforementioned survey from 2004 [11].

In a slightly modified framework, in which pairs of sets and objects are ranked, Gravel [33] observes that (a strong form of) *freedom of choice* contradicts principles deriving from indirect utilities, i.e., the idea that a set is preferred if it offers a better alternative than all alternatives the other set contains.

Apart from these examples, we are not aware of other impossibility theorems in the literature on *ranking sets of objects*.

Applications of Automated Reasoning

Classically, successful applications of were mostly to be found in pure mathematics (theorem proving), circuit design (program verification) and medical diagnosis (expert systems) [80]. The more recent contributions in the area of applying automated reasoning and logic to (social) choice theory are, however, not less successful and span a variety of different approaches. Multiple criteria can be used for the classification of these approaches: the logic chosen for formalization, the utilized methods and software, and whether they generate proofs or just verify them. In the following we present an overview of the recent work in this field and describe the employed approaches with respect to the above dimensions. Left out of more detailed considerations are pure formalizations without the application of any automated method, like the works by Rubinstein [56] and Ågotnes et al. [1] (the former employing first-order, the latter modal logic).

Let us start by reviewing approaches in *higher-order logic* (HOL), which is the most expressive and thus human-readable formal framework we encountered. Three authors independently formalized impossibility theorems and their proofs in this framework, but to a different extent and on different systems. Nipkow [47] and Gammie [28] used the *Isabelle* [48] system, a highly automatized *proof checker*, whereas Wiedijk [78] utilized the automated proof checker *Mizar* [75]. They all presented a formalization of Arrow's Theorem; Gammie additionally verified the related results of Sen's [64] and May's Theorem [42].

Instead of verifying *given* proofs, Grandi and Endriss [32] attempted to use automated theorem provers for the discovery of these. Still offering a human-readable formalization, but yet with powerful automated provers available, their choice of language fell on first-order logic (FOL). Even though Grandi and Endriss were able to automatically generate a proof in their formalization of a subproblem of Arrow's Theorem, the employed automated provers did not succeed to prove the full result in an initial experiment.

The approach that is least human-readable, but also most successful and most related to our work has been pursued by Tang and Lin, who formalized settings of theorems in social choice and game theory. They always employed proofs by induction, of which the inductive step was established manually and the base cases checked on a computer. In social choice theory they mostly used (computer-instantiated) formalizations in propositional logic together with SAT solvers for the automated verification. Like this, Tang and Lin automatically re-proved Arrow's, Sen's, and the Muller-Satterthwaite Theorem [73], as well as the Gibbard-Satterthwaite Theorem and Maskin's Theorem [41, 72]. Additionally, Lin and Tang [39] were able to relax each of the assumptions of *unanimity* and *independence of irrelevant alternatives* in Arrow's Theorem yielding a strengthening of the result. A summary of the complete work (including the results in game theory) can also be found in Tang's PhD thesis [71].

Contribution

The main contribution of our work is two-fold: for one, we developed an automated search method for impossibility theorems, and for another, we used this method for the discovery of concrete impossibility results in the realm of *choice under complete uncertainty*.

The former is based upon two components: a universal step (Corollary 4.16), which reduces impossibilities to small instances, and a computer-aided formulation of those small instances in propositional logic to be checkable by a SAT solver. The universal step is directly implied by our Preservation Theorem 4.13, which the whole of Chapter 4 is devoted to and which was proved manually with tools from model theory. The formalization and automated instantiation of small instances is explained exemplarily for the Kannai-Peleg Theorem in Chapter 3.

The latter part of our overall results, consisting of 84 concrete impossibilities and a few conjectures about general possibilities, is described in Chapter 5. The highlights of new discoveries are a highly non-intuitive impossibility (even once published as a possibility) in Theorem 5.1 and two impossibilities without any form of dominance (Theorems 5.6 and 5.7).

Overview of the Thesis

This thesis tries to be as self-contained as possible. In the few places where the consultation of external literature might be helpful nonetheless, explicit references are given.

The remainder of this thesis is organized as follows:

Chapter 1 provides an introduction to the field of *ranking sets of objects* as well as its applications. After describing some of the usual interpretations attached to this area and giving some examples of basic concepts, we define the formal mathematical framework as well as our notation. With the mathematical setting defined, the axioms of the Kannai-Peleg Theorem are discussed and the theorem itself is formally stated and proven.

In Chapter 2 we exhibit an initial attempt to automatically verify the Kannai-Peleg Theorem by means of a first-order theorem prover. We first present our first-order formalization of the framework and the axioms of the theorem, which unfortunately did not lead to an automatic proof within a running time limit of five days. Therefore, also a few subproblems and easier problem variants have been formalized and are also presented in this chapter. For a few of these, automatic proofs could be obtained and so we conclude by giving an overview of the results and discussing these. Since the insights from this chapter are not required for an understanding of later chapters, it can be skipped at the reader's discretion.

The approach by Tang and Lin to use computer-aided induction (on a formalization in propositional logic) for proving impossibility theorems is taken up in Chapter 3, where we start with reviewing their technique and ideas at the example of Arrow's famous impossibility theorem about preference aggregation. We proceed by extending their method to also fit the setting of the Kannai-Peleg theorem and successfully verify it automatically with the help of a SAT solver. This process comprises an inductive step and manual proof thereof, as well as the conversion of the theorem's axioms to propositional logic. The chapter concludes by laying out a few additional features of our implementation, like the ability to produce models for possible instances and the automatic verification of a detected unsatisfiability.

In Chapter 4 we generalize the method of the preceding chapter, by reducing a whole class of impossibility theorems to small instances. This allows us to prove full impossibilities by just checking their respective base cases on a computer like shown before. The reduction result and its proof require a few tools from (many-sorted) model theory and so we first introduce these tools and describe the specific language we use. We then proceed by providing two versions of the reduction theorem, both of which imply a universal step (going from base cases to full results) to be used for proving impossibility theorems automatically in the succeeding chapter.

Our automated and exhaustive theorem search is finally introduced and demonstrated in Chapter 5. We first describe how our universal step allows for an efficient search algorithm and explain its implementation. Then an actual run of the program (with a maximal domain size of eight elements and on a space of 20 axioms) is performed and its results of 84 impossibilities are analyzed. The fruitfulness of the approach is underlined by manual proofs of the most interesting results, which also extends our analysis of these.

Thereafter, we conclude this thesis by summing up, putting our two-fold results (consisting of the impossibility theorems and the search method itself) into a broader context, and by providing a selection of ideas for future work.

1 Ranking Sets of Objects

1.1 Introduction

This chapter introduces the reader to the field of *ranking sets of objects* and in particular to its most famous impossibility result: the Kannai-Peleg Theorem [37], which was discovered in 1984.

Ranking sets of objects deals with questions related to how an agent should rank sets of objects, given her preferences over individual objects. As we will see, answers to this question highly depend on the concrete interpretation assigned to sets. The three most common interpretations and how they shape the extension of preferences over elements to preferences over sets will be explained in the next section. Later we focus on the idea of *complete uncertainty* only, where sets are interpreted as containing mutually exclusive alternatives from which a random device (that the agent has no control over) picks a final outcome. We chose this particular interpretation for two different reasons: for one, since this interpretation is the natural interpretation for analyzing *strategy-proofness*[4] of voting correspondences, and for another, since this area features the most famous and surprising impossibility results known to date.

We also present the notation and mathematical framework usually employed to treat the problems in *ranking sets of objects*. This will then also enable us to formally state and prove the Kannai-Peleg Theorem, which says that, in general, no weak order can be found that satisfies the seemingly innocuous combination of the *Gärdenfors principle* and the principle of *independence*.

[4] The concept of *strategy-proofness* (or *non-manipulability*) describes voting procedures or correspondences that make agents vote sincerely, i.e., do not give agents an incentive to misrepresent their preferences.

1.2 Motivating Examples and Basic Concepts

Being mostly based on a survey by Barberà, Bossert and Pattanaik [11], this section offers an overview on the different ways, in which one can look at the problem of extending preferences from elements to sets.

Although the interpretation of sets plays a large role for how to approach the problem, the questions and the basic framework remains the same for all (common) interpretations. All interpretations assume that an agent has some kind of preference over individual objects and then analyze properties a "compatible" ranking of sets of objects should fulfill. These properties, or *principles*, vary a lot between different interpretations of sets.

Consider the simple setting of only two objects x and y, of which an agent prefers the former to the latter. What does this imply for the relative rankings of $\{x\}$, $\{y\}$, and $\{x, y\}$?

Barberà et al. [11] intuitively justify each of the possible rankings by giving examples of settings, in which the respective rankings appear as the natural choice. Here we just describe a few different situations corresponding to the three most common interpretations of sets:

Complete uncertainty. In this setting sets are considered as containing mutually exclusive alternatives from which the final outcome is chosen at a later stage. The agent, however, does not have any influence on the selection procedure. One usually assumes that some random device, of which the agent has no information, will make the choice. Applications of this interpretation are, for instance, voting with unspecified tie-breaking rules, or approval voting, where agents submit a set of candidates (they approve of).

In the two-element case, the scenario of ranking $\{x\}$ above $\{x, y\}$, and $\{x, y\}$ above $\{y\}$ can intuitively be supported, since the singletons just represent the actual outcomes, and for $\{x, y\}$ there is a chance that either of them is selected, raising its value above $\{y\}$, but not as high as $\{x\}$.

But also the extremely pessimistic view of being indifferent with respect to $\{x, y\}$ and $\{y\}$ might have its justification, since both sets have the same worst-case outcome.

Opportunity sets. Here again sets contain mutually exclusive alternatives, but this time the agent can himself choose a final outcome from the set. Sets are therefore considered as collections of opportunities for the agent. This approach is particularly helpful when modeling choice situations in which other criteria than choosing the best available option are desired. Flexibility and a preference for freedom of choice are two such principles that can easily be implemented in this setting.

A decision maker could, thus, not only care about the outcome but also (or even more) about the opportunities offered. Ranking $\{x, y\}$ above $\{x\}$, and $\{x\}$ above $\{y\}$ reflects such behaviour, in that it shows a preference for more freedom and flexibility.

Sets as final outcomes. In this interpretation sets contain compatible objects, which are hence not mutually exclusive. All objects contained in the same set are hence assumed to materialize simultaneously. Examples are, for instance, the constitution of a committee, an assignment of tasks among agents, or a company hiring a set of new employees based on the ranking of the individual applicants.
For desirable items, the above example for *opportunity sets* could also be considered sensible here as getting both items is clearly better than just getting one. If, on the other hand, items come with obligations or are undesirable in general, then $\{x\}$ above $\{y\}$, and $\{y\}$ above $\{x, y\}$ makes even more sense.

Thus, we can see that, depending on the interpretation, different principles of extending an order on elements to an order on sets have to be considered natural. Impossibilities, i.e., incompatible principles, can occur in any interpretation, however. We will now, as mentioned above already, focus on only one of the interpretations: *complete uncertainty.*

In the framework of *complete uncertainty* a very natural and basic concept is the principle of *extension*. It says that the relative ranking of singleton outcome sets has to agree with the relative rankings of the corresponding outcomes themselves, i.e., if an agent prefers alternative x over alternative y, then she should also prefer the singleton set $\{x\}$ to the singleton set $\{y\}$. Intuitively, this has a lot of support because sets are interpreted as collection of outcomes from which a random device will choose, and therefore singleton sets do not leave any choice and the contained alternative will immediately be selected as the final outcome.

The *extension rule* can therefore be considered a very natural axiom for any set extension of preferences.[5] Taking this idea further one could also ask the ranking of sets to be such that a set is considered (strictly) better than another one if the agent (strictly) prefers all of its elements to all elements of the second set. Can et al. [17] call this idea the *Kelly principle* and show that it is satisfied for any utility-maximizing agent regardless of its utility function and the assumed (non-degenerate) probability distribution over elements. In other words, the *Kelly principle* can be considered rational in the setting of *choice under complete uncertainty* even without any assumptions about the random device making the "final" decision.

[5] It should be noted, however, that this intuitiveness of the axioms relies on our interpretation of sets as uncertain outcomes. If one, for instance, models a setting in which the agent can freely choose a final outcome from the sets, an axiom saying that the agent is indifferent between no-choice situations, i.e., singleton sets, can make sense, too. In fact, such an axiom was introduced by Jones and Sugden [36] and later (independently) by Pattanaik and Xu [52].

Endriss [23] notes that the *Kelly principle* can be characterized by the *extension rule* together with two axioms saying that the singleton containing just the maximal (minimal) element of a set has to be considered at least as good (bad) as the set itself. The formal axiom statements are given in the next Section 1.3, before their connection to another important axiom called the *Gärdenfors principle* is explored in Section 1.4.1.

1.3 Notation and Mathematical Setting

The mathematical framework for the questions as formulated in the previous Section 1.2 is relatively easy to define and to understand. All we need is a finite set X of *alternatives*, on which a (preference) order $\dot{\geq}$ is defined. The order $\dot{\geq}$ is assumed to be *linear*,[6] i.e., it is a binary relation fulfilling the axioms

$$\begin{array}{lrl} (\texttt{LIN}) & x \dot{\geq} x & \text{for all } x \in X \text{ (reflexivity)}, \\ & x \dot{\geq} y \vee x \dot{\leq} y & \text{for all } x \neq y \in X \text{ (completeness)}, \\ & x \dot{\geq} y \wedge y \dot{\geq} z \Rightarrow x \dot{\geq} z & \text{for all } x, y, z \in X \text{ (transitivity)}, \\ & x \dot{\geq} y \wedge y \dot{\geq} x \Rightarrow x = y & \text{for all } x, y \in X \text{ (antisymmetry)}. \end{array}$$

The symbol $\dot{>}$ will denote the strict component of $\dot{\geq}$, i.e.,

$$x \dot{>} y \text{ abbreviates } x \dot{\geq} y \wedge y \dot{\ngeq} x.$$

The interpretation of $\dot{\geq}$ is such that $x \dot{\geq} y$ if and only if x is considered at least as good as y by the decision maker.

Similarly, we have an order $\succeq$ on the set of non-empty subsets of X (denoted by $\mathcal{X} := 2^X \setminus \{\emptyset\}$). This relation, however, is only required to be a *weak order*,[7] i.e., a binary relation fulfilling only the smaller axiom set

$$\begin{array}{lrl} (\texttt{WEAK}) & A \succeq A & \text{for all } A \in \mathcal{X} \text{ (reflexivity)}, \\ & A \succeq B \vee A \preceq B & \text{for all } A \neq B \in \mathcal{X} \text{ (completeness)}, \\ & A \succeq B \wedge B \succeq C \Rightarrow A \succeq C & \text{for all } A, B, C \in \mathcal{X} \text{ (transitivity)}. \end{array}$$

Like above, we use $\succ$ for the strict component of $\succeq$, i.e.,

$$A \succ B \text{ abbreviates } A \succeq B \wedge B \not\succeq A.$$

Additionally, we also define indifference $\sim$ by setting

$$A \sim B \text{ if and only if } A \succeq B \wedge B \succeq A.$$

[6] Other names for the same concept include *total order* and *simple order*.

[7] Again there exist other names for the same concept, like *total preorder* or just *ordering*.

Remark 1.1. Note that — since we are going to use this fact later on — transitivity of the non-strict relations $\dot{\geq}$ and $\succeq$ carries over to their strict components $\dot{>}$ and $\succ$, respectively. Actually, even if just one comparison is strict, transitivity yields a strict inequality, which we are going to prove for the case of a general transitive binary relation $\geq$.

Proof. Let $\geq$ be a binary, transitive relation on a set D and define $>$ as its strict component, i.e.,

$$x > y \iff x \geq y \land y \ngeq x \text{ for all } x, y \in D.$$

Let furthermore a, b, c be three elements of D such that $a > b$ and $b \geq c$. Then, since $x > y$ implies $x \geq y$ for all $x, y \in D$, we have $a \geq c$ by transitivity of $\geq$. Thus, it only remains to show that $a \nleq c$. By way of contradiction, suppose $a \leq c$. But then, since by assumption $c \leq b$, also $a \leq b$ and therefore $a \ngtr b$ holds and we have our desired contradiction with $a > b$.

For the case of $a \geq b$ and $b > c$ the proof is completely analogous. Purely strict transitivity (i.e., $a > b > c \Rightarrow a > c$) follows from either of the two cases. $\square$

Having set up the mathematical basics, we can easily formalize the first concepts, properties, or requirements (from here on called *axioms*), which we stated in the previous section.

The *Kelly principle*, for example, can be described by the following axioms, where $\max(A)$ and $\min(A)$ denote the (unique) best and worst element according to $\dot{\geq}$, respectively:[8]

$$\begin{array}{lcl}
(\texttt{EXT}) & x \mathrel{\dot{\geq}} y \iff \{x\} \succeq \{y\} & \text{for all } x, y \in X, \\
(\texttt{MAX}) & \{\max(A)\} \succeq A & \text{for all } A \in \mathcal{X}, \\
(\texttt{MIN}) & A \succeq \{\min(A)\} & \text{for all } A \in \mathcal{X}.
\end{array}$$

Despite the simplicity of the mathematical model, consequences of axioms formulated in this framework are far from being obvious. The fact that sometimes very intuitive and straightforward axioms can turn out to be inconsistent, in the sense that no compatible weak order $\succeq$ on the set $\mathcal{X}$ of non-empty subsets can be found, strongly supports this claim. We are going to exemplify such an inconsistency in the following section by looking at the example of the Kannai-Peleg Theorem, which was proved only 26 years ago [37]. It shows that already three simple and intuitive axioms suffice to create this kind of impossibility.

[8] In formal terms,

$$\begin{aligned}
\max(A) = x &\iff x \in A \text{ and } x \mathrel{\dot{\geq}} a \text{ for all } a \in A, \text{ and analogously} \\
\min(A) = x &\iff x \in A \text{ and } x \mathrel{\dot{\leq}} a \text{ for all } a \in A.
\end{aligned}$$

That this is well-defined follows from the fact that the maximum and the minimum element are unique in a linear order.

1.4 The Kannai-Peleg Theorem

In order to state and prove the Kannai-Peleg Theorem, we first have to turn towards the axioms involved.

1.4.1 Axioms

The Kannai-Peleg Theorem makes use of two axioms (or principles), which both have very plausible intuitive interpretations. On the one hand there is the *Gärdenfors principle*,[9] also called *dominance*. It consists of two parts and requires that

1. adding an element, that is strictly better ($\dot{>}$) than all the elements in a given set, to that given set produces a *strictly* better set with respect to the order $\succeq$,
2. adding an element, that is strictly worse ($\dot{<}$) than all the elements in a given set, to that given set produces a *strictly* worse set with respect to the order $\succeq$.

Formally, the Gärdenfors principle (`GF`) can be written as the following two axioms:

$$(\texttt{GF1}) \quad ((\forall a \in A) x \mathrel{\dot{>}} a) \Rightarrow A \cup \{x\} \succ A \quad \text{for all } x \in X \text{ and } A \in \mathcal{X},$$
$$(\texttt{GF2}) \quad ((\forall a \in A) x \mathrel{\dot{<}} a) \Rightarrow A \cup \{x\} \prec A \quad \text{for all } x \in X \text{ and } A \in \mathcal{X}.$$

On the other hand, we have a monotonicity principle called *independence*, which states that, if a set is strictly better than another one, then adding the same alternative (which was not contained in either of the sets before) to both sets simultaneously does not reverse this strict order. An equivalent way of stating this (in light of completeness of the order) is to require that at least a non-strict preference remains of the original strict preference (such that $\succ$ becomes $\succeq$). The formal statement reads as follows:

$$(\texttt{IND}) \quad A \succ B \Rightarrow A \cup \{x\} \succeq B \cup \{x\} \quad \text{for all } A, B \in \mathcal{X} \text{ and } x \in X \setminus (A \cup B).$$

That this indeed defines a monotonicity condition can best be seen when considering the function $\varphi_x : \{A \in \mathcal{X} \mid x \notin A\} \to \mathcal{X}$, $\varphi_x(A) := A \cup \{x\}$ for a fixed $x \in X$. Axiom (`IND`) exactly guarantees that φ_x is (non-strictly) monotone with respect to $\succeq$ for any $x \in X$, i.e., for all sets $A, B \in \text{dom}(\varphi_x)$ such that $A \prec B$ we have that $\varphi_x(A) \preceq \varphi_x(B)$.

One might wonder what the relation is between these axioms and the Kelly principle ((`EXT`), (`MIN`), (`MAX`)), which was introduced in Section 1.3. A partial answer can be given by the following well-known proposition that identifies the Kelly principle as a consequence of the Gärdenfors principle.

[9] According to Barberà et al. [11] the name was coined by Kannai and Peleg [37] "in recognition of the use of this axiom in Gärdenfors [29]".

Proposition 1.2. *The Gärdenfors principle entails the Kelly principle, i.e., if the weak order $\succeq$ satisfies* (`GF1`) *and* (`GF2`) *then it also satisfies* (`EXT`)*,* (`MIN`) *and* (`MAX`)*.*

Proof. We prove the axioms of the Kelly principle one by one:
(`EXT`): Let x, y be elements of X and suppose that $x \mathrel{\dot{\geq}} y$. Since $\dot{\geq}$ is a linear order, this is equivalent to $x \mathrel{\dot{>}} y$ by antisymmetry. Then it follows from (`GF1`) that $\{x, y\} = \{y\} \cup \{x\} \succ \{y\}$, and similarly from (`GF2`) that $\{x\} \succ \{x\} \cup \{y\} = \{x, y\}$. Together, this directly gives $\{x\} \succ \{x, y\} \succ \{y\}$, and one application of transitivity finishes one direction of the proof. Let now x, y be elements of X such that $\{x\} \succeq \{y\}$. Aiming at a contradiction, suppose $x \mathrel{\dot{\ngeq}} y$. Completeness implies that $y \mathrel{\dot{>}} x$ and so we can, in the same way like before, derive $\{y\} \succ \{x\}$ via (`GF1`) and (`GF2`). This, however, implies that $\{x\} \not\succeq \{y\}$. Contradiction!
(`MIN`): Let $A \in \mathcal{X}$ be a subset of X with cardinality $|A| = n \in \mathbb{N}$ and denote its elements by a_i, $i \in \{1, 2, \ldots, n\}$ such that these are ordered descendingly by $\dot{>}$ with respect to their index, i.e., $\max(A) = a_1 \mathrel{\dot{>}} a_2 \mathrel{\dot{>}} \ldots \mathrel{\dot{>}} a_n = \min(A)$. Note first that for the case of $n = 1$ the claim is trivially true by (and actually equivalent to) reflexivity of $\succeq$. For $n \geq 2$, we get a chain of inequalities by (`GF1`):

$$\begin{aligned}\{a_1\} \succ \{a_1\} \cup \{a_2\} = \{a_1, a_2\} \succ \{a_1, a_2\} \cup \{a_3\} = \{a_1, a_2, a_3\} \succ \ldots \\ \succ \{a_1, a_2, \ldots, a_{n-1}\} \cup \{a_n\} = \{a_1, a_2, \ldots, a_n\}.\end{aligned}$$

The claim then follows by transitivity of $\succ$.
(`MAX`): The case of (`MAX`) is completely analogous to the one of (`MIN`), just using (`GF2`) instead of (`GF1`). □

Remark 1.3. In the preceding Proposition 1.2, we actually proved strict (and hence stronger) versions of (`MIN`) and (`MAX`), which we will denote by (`SMIN`) and (`SMAX`), respectively:

$$\begin{array}{lll} (\texttt{SMAX}) & \{\max(A)\} \succ A & \text{for all } A \in \mathcal{X} \text{ with } |A| > 1, \\ (\texttt{SMIN}) & A \succ \{\min(A)\} & \text{for all } A \in \mathcal{X} \text{ with } |A| > 1. \end{array}$$

A simple example with three elements $x \mathrel{\dot{>}} y \mathrel{\dot{>}} z$ can be used to show that the other direction of Proposition 1.2 does not hold. The set $\{x, y\}$ must be strictly preferred to the full set $X = \{x, y, z\}$ by (`GF2`), whereas according to the Kelly principle nothing is required of the relative ranking of the two, and therefore they could be ordered exactly oppositely ($\{x, y\} \prec X$). Thus, one can easily construct an example of a model that complies with the Kelly principle but fails with respect to the Gärdenfors principle. The latter is hence a strictly stronger set of axioms than the former.

Since the principle of independence does (as a monotonicity requirement) not say anything about the order of singleton sets, it can easily be seen to be logically independent from any of the axioms of the Kelly principle.[10]

[10] In the sense that none of the axioms is implied by nor implies any of the others.

1.4.2 The Theorem

With the axioms formally defined we can now state and prove the Kannai-Peleg Theorem, a result due to Kannai and Peleg [37]. They were probably the first to treat the specific problem of extending preferences from elements to subsets as a problem in its own right in an axiomatic fashion. In previous work, other authors had regarded this problem as more of a side issue of another problem (see, e.g., Fishburn [25] and Gärdenfors [30]) or had merely axiomatized specific methods of extension without considering the general problem (see, e.g., Packard [50]).

Regarding its nature, the Kannai-Peleg Theorem is an impossibility result because it states that certain axioms, viz. (GF) and (IND), are incompatible from a certain domain size upwards. Other famous impossibility theorems in social choice theory include Arrow's Theorem, the Gibbard-Satterthwaite Theorem, and Sen's Impossibility of a Paretian Liberal. For a good overview on the area in general, and the aforementioned theorems in particular, see Gaertner [27].

But before we get to the main theorem, we give a lemma saying that only very specific rankings can satisfy the conditions (SMIN), (SMAX) and (IND). The proof idea has been taken from Kannai and Peleg [37], who used a similar lemma (which we will state as Corollary 1.5).

Lemma 1.4. *If $\succeq$ satisfies the axioms* (SMIN)*,* (SMAX) *and* (IND)*, then $A \sim \{\max(A), \min(A)\}$ for all $A \in \mathcal{X}$.*

Proof. Let A be a non-empty subset of X. If $|A| \leq 2$ then the lemma holds trivially by reflexivity of $\succeq$ since then $A = \{\max(A), \min(A)\}$. So suppose $|A| \geq 3$ and define $A_- := A \setminus \max(A)$. Note that, because $|A| \geq 3$, the set A_- is non-empty and thus it is easy to see that $\{\min(A)\} = \{\min(A_-)\}$. Axiom (SMIN) applied to A_- then yields

$$\{\min(A)\} = \{\min(A_-)\} \prec A_-. \tag{1.1}$$

We can then add $\max(A)$ to both sides, showing that $\{\min(A), \max(A)\} \preceq A_- \cup \{\max(A)\} = A$ by (IND). In a completely analogous way we get $\{\min(A), \max(A)\} \succeq A_+ \cup \{\min(A)\} = A$ from (SMAX), where $A_+ := A \setminus \min(A)$. □

Corollary 1.5 (Kannai and Peleg, 1984). *If $\succeq$ satisfies the Gärdenfors principle* (GF) *and independence* (IND)*, then $A \sim \{\max(A), \min(A)\}$ for all $A \in \mathcal{X}$.*

Proof. By Proposition 1.2 and Remark 1.3, the Gärdenfors principle (GF) entails (SMIN) and (SMAX). The statement of the corollary hence follows from Lemma 1.4 immediately. □

The importance of Lemma 1.4, and in particular of the condition

$$A \sim \{\max(A), \min(A)\} \text{ for all } A \in \mathcal{X},$$

should be stressed. The fact that every set is ranked exactly like the set containing just its best and worst element means that the ranking of subsets is completely determined by their worst and best elements. The relative ranking of any two sets A and B can therefore be determined by looking at the order of the (maximally two-element) sets $\{\max(A), \min(A)\}$ and $\{\max(B), \min(B)\}$ only.

We are now finally in the position to state and prove the main theorem.

Theorem 1.6 (Kannai and Peleg, 1984). *Let X be a linearly ordered set with $|X| \geq 6$. Then there exists no weak order $\succeq$ on $\mathcal{X}$ satisfying the Gärdenfors principle* (GF) *and independence* (IND).

Proof. Let x_i, $i \in \{1, 2, \dots, 6\}$ denote six distinct elements of X such that they are ordered descendingly by $\dot{>}$ with respect to their index, i.e., $x_1 \dot{>} x_2 \dot{>} x_3 \dot{>} x_4 \dot{>} x_5 \dot{>} x_6$. By way of contradiction, suppose there exists a weak order $\succeq$ on $\mathcal{X}$ satisfying the Gärdenfors principle (GF) and independence (IND). We first claim that then

$$\{x_2, x_5\} \succeq \{x_3\}. \tag{1.2}$$

In order to prove this claim suppose that the contrary is the case, which by completeness of $\succeq$ is $\{x_3\} \succ \{x_2, x_5\}$. We can then, by (IND), include x_6, which yields

$$\{x_3, x_6\} \succeq \{x_2, x_5, x_6\}. \tag{1.3}$$

Note now that by Lemma 1.4,

$$\{x_3, x_4, x_5, x_6\} \sim \{x_3, x_6\} \text{ and} \{x_2, x_5, x_6\} \sim \{x_2, x_6\} \sim \{x_2, x_3, x_4, x_5, x_6\},$$

which, together with (1.3) gives

$$\{x_3, x_4, x_5, x_6\} \sim \{x_3, x_6\} \succeq \{x_2, x_5, x_6\} \sim \{x_2, x_3, x_4, x_5, x_6\}$$

by transitivity. But this contradicts (GF1), which states that $\{x_3, x_4, x_5, x_6\} \prec \{x_2, x_3, x_4, x_5, x_6\}$, since $x_2 \dot{>} x_j$ for $j \in \{3, 4, 5, 6\}$.

Therefore, claim (1.2) must be true and it follows from $\{x_3\} \succ \{x_4\}$ (which is a consequence of the extension axiom, see Proposition 1.2) together with transitivity that $\{x_2, x_5\} \succ \{x_4\}$. Using (IND) again, we can add (the so far unused) x_1 and get $\{x_1, x_2, x_5\} \succeq \{x_1, x_4\}$ at the price of strictness of the inequality.

Like before, we again fill in the intermediate elements to both sets and obtain, by Lemma 1.4 and transitivity, that $\{x_1, x_2, x_3, x_4, x_5\} \succeq \{x_1, x_2, x_3, x_4\}$, which this time contradicts the second axiom (GF2) of the Gärdenfors principle.

Thus, there cannot be any weak order $\succeq$ on $\mathcal{X}$ satisfying the Gärdenfors principle (GF) and independence (IND). □

Remark 1.7. Even though Lemma 1.4 and Corollary 1.5 only hold for finite sets $A \in \mathcal{X}$, the proof of the Kannai-Peleg Theorem works even for the case of an infinite domain. This is noteworthy since, for instance, Arrow's Theorem [5], does not have this feature.[11]

Put in different words, the Kannai-Peleg Theorem says that the Gärdenfors principle (`GF`) and independence (`IND`) are inconsistent for a domain of six or more elements. Thus, there is no way of extending a linear order on a set of at least six elements to a weak order on the collection of all non-empty sets of these elements. Note that whereas a strengthening of the theorem to weak orders on X is possible as long as there are at least six $\dot{>}$-distinguishable elements, a weakening of the order on $\mathcal{X}$ to a preorder (dropping completeness) breaks the theorem as already Kannai and Peleg [37] observe.[12]

1.5 Summary

This chapter has offered an overview on the framework of *ranking sets of objects*, in which the Kannai-Peleg Theorem is placed.

We first laid out the general ideas behind *ranking sets of objects* and gave basic examples motivating the approach, followed by the definition of a mathematical framework to capture these ideas and first formalizations of simple concepts like the Kelly principle.

Finally, the axioms of the Kannai-Peleg Theorem were introduced and explained in order to formally state and prove it (Theorem 1.6). The theorem is an impossibility result saying that a certain combination of natural principles (axioms) is incompatible, in the sense that no weak order can be constructed satisfying these principles.

The next chapter will exhibit a first (unfortunately unsuccessful) approach to automatically verifying the theorem.

[11] Fishburn [24] was the first to point out that Arrow's Theorem fails for the case of an infinite number of voters.

[12] One possible such preorder is given by

$$A \succeq B \iff \min(A) \dot{\geq} \min(B) \wedge \max(A) \dot{\geq} \max(B)$$

for all $A, B \in \mathcal{X}$.

2

First-Order Automated Theorem Proving

2.1 Introduction

Our first attempt to automatically verify the Kannai-Peleg Theorem is discussed in this chapter. We used different first-order theorem provers on a first-order formalization of the theorem. Unfortunately, none of the automated theorem provers was able to deliver a proof in the given time limit of 120 hours (5 days). Nevertheless, we would like to present our formalization since it is non-trivial, and itself or a modification could potentially still lead to an automated proof on a different system or with more powerful computers (or even computing clusters).[13]

Intuitively, the most suitable logic to express the setting of the Kannai-Peleg is theorem is second-order logic (since in most of the axioms quantifications over subsets occur). Non-monadic[14] second-order theorem provers, however, are not quite as developed as their first-order counterparts and so they are no option for us. On the other hand, formalizing the problem in a less expressive logic than first-order logic will drastically reduce readability, which is why a first-order formalization appears to be the natural choice for verifying the Kannai-Peleg theorem.

A similar approach has also been taken by Grandi and Endriss [32], who have formalized Arrow's Theorem [5] in first-order logic and have experimented with automated theorem provers to obtain a proof thereof. Even though in practice the theorem provers have so far not been successful in providing a proof of the full result either, Grandi and Endriss were able to automatically produce a proof of a lemma saying that certain axioms imply others.

[13] We are, however, not very optimistic about this, unless a formalization leading to a more succinct representation can be found. See also Section 2.3.

[14] Monadic second-order logic is not sufficient since it only allows monadic, i.e., one-place, second-order predicates, while the order $\succeq$ on $\mathcal{X}$ is a two-place relation.

We are going to first present and discuss our first-order axiomatization, before we give the results of running automated theorem provers on a few problems based on this axiomatization.

2.2 Axioms

The main problem one encounters when trying to craft a first-order axiomatization of the Kannai-Peleg Theorem is how to treat subsets in a way that they can be quantified over. The idea we used is similar to the one employed by Grandi and Endriss [32] in order to model preference profiles (i.e., tuples of linear orders). They defined a partition on the domain, splitting it into the three sorts[15] *alternatives*, *individuals*, and *situations* and then have an axiom (`PERM`) that makes sure all preference profiles (now called *situations*) are considered in any model of the axioms.

Our partitioning of the domain will distinguish *elements* and *sets*, i.e., we just treat sets as atomic elements of the domain. In order for this to work (and then also be able to perform operations like *union*, *singleton*, etc.) we need further axioms that define the relation between elements and sets and that ensure that all non-empty subsets are present in any structure modeling the axioms.

Formally, we define structures consisting of

- constants `a1` to `a6` and `0` (ensuring the minimal size of X, and serving as a special object, respectively),
- one-place relations `S` and `E` (marking *sets* and *elements*, respectively),
- two-place relations `l`, `w`, `el` (the orders on X and $\mathcal{X}$, and an *element-of* relation, respectively),
- a one-place function `singleton` (transforming elements into singleton sets), and
- a two-place function `union` (the usual *set union*).

In the following, we are going to describe and explain our axiomatization – first of the framework, then of the particular axioms for the Kannai-Peleg Theorem – step by step to make clear how it exactly works. All axioms will be stated in the LADR-syntax used by *Prover9* [44] since it is quite natural and should be readable without further explanation.[16] Furthermore, the axioms can this way serve as direct input for the prover.[17] We start with the axioms for the partitioning of the domain:

[15] The idea of using sorts will be applied again in Chapter 4 where we lay the foundation for our automated theorem search by generalizing a result concerning *ranking sets of objects* using tools from (many-sorted) model theory.

[16] Only note that unbound variables are automatically quantified over universally.

[17] For *Prover E* and *iProver* the (less-intuitive) TPTP-format [70] is necessary, which can be obtained automatically [68] from the LADR-syntax.

```
%PART
E(x) -> -S(x) & x!=0.
S(u) -> -E(u) & u!=0.
x=0 -> -S(x) & -E(x).
E(x) | S(x) | x=0.
```

We can see that we actually have *three* sorts, into which the whole domain is partitioned: the two aforementioned *elements* and *sets*, as well as a special object `0`, which we need since first-order functions will have to be defined on the whole domain and need to mark "illegal" applications. What exactly "legal" applications of functions and relations are, is treated in the next axiom, called `SORT`:

```
%SORT
w(u,v) -> S(u) & S(v).
l(x,y) -> E(x) & E(y).
el(x,u) -> E(x) & S(u).
S(u) & S(v) & union(u,v)=w -> S(w).
(-S(u) | -S(v)) -> union(u,v)=0.
E(x) & singleton(x)=u -> S(u).
-E(x) -> singleton(x)=0.
```

It makes sure that relations are only true of the right sorts and that functions yield the right types (including that applications to unqualified sorts yield the special object `0`).

Next we need to establish that we actually have all (finite) non-empty subsets in our *set*-domain and that their relations and operations are defined in the right way. This is taken care of by the following axioms:

```
%ALL NON-EMPTY SETS
E(x) -> ( exists u (S(u) & u=singleton(x) & all v (S(v) & v!=u ->
v!=singleton(x))) ).
S(u) & S(v) -> ( exists w (S(w) & w=union(u,v) & all w1 (S(w1) & w1!=w
-> w1!=union(u,v))) ).
S(u) -> ( exists x (E(x) & el(x,u)) )

%UNIQUENESS OF SETS
S(u) & S(v) -> ( (all x (E(x) -> (el(x,u) <-> el(x,v)))) -> u=v ).
```

The first part `ALL NON-EMPTY SETS` says that for all elements there is *exactly* one singleton *set* containing it, similarly that for any pair of sets there is *exactly* one *set* that is the union of the two, and finally that there is no empty set. The second part `UNIQUENESS OF SETS` declares that sets with the same elements are equal. This prevents the existence of "copies" of sets, which would certainly be problematic.

The two self-explanatory axioms defining the functions singleton and union finish our modeling of the element-set relations and functions. The missing relation `el` can remain implicitly defined through the above axioms `ALL NON-EMPTY SETS` and `UNIQUENESS OF SETS` in connection with the following two definitions of the functions `singleton` and `union`.

```
%singleton
E(x) & S(u) -> ( singleton(x)=u <-> el(x,u) & (all y ((y!=x) ->
-el(y,u))) ).

%union
S(u) & S(v) & S(w) -> ( union(u,v)=w <-> all x (E(x) -> (el(x,w) <->
(el(x,u) | el(x,v)))) ).
S(u) & S(v) -> ( union(u,v)=union(v,u) ).
```

We complete the axiomatization of the general framework of *ranking sets of objects* by making the relations $\dot{\geq}$ and $\succeq$ linear and weak orders, respectively.

```
%LINl
E(x) -> l(x,x). %Reflexivity
E(x) & E(y) & (x!=y) -> l(x,y) | l(y,x). %Completeness
E(x) & E(y) & E(z) & l(x,y) & l(y,z) -> l(x,z). %Transitivity
E(x) & E(y) & l(x,y) & l(y,x) -> (x=y). %Antisymmetry

%WEAKw
S(u) -> w(u,u). %Reflexivity
S(u) & S(v) & (u!=v) -> w(u,v) | w(v,u). %Completeness
S(u) & S(v) & S(w) & w(u,v) & w(v,w) -> w(u,w). %Transitivity
```

Any finite model of the above axioms, which we denote by Γ_{frame}, will now correspond to a linearly-ordered set X with a weak order on the space of all (finite) non-empty subsets $\mathcal{X}$. In order to formalize the Kannai-Peleg Theorem we still need to include its particular axioms and postulate a domain size of at least six elements. We start with the latter since it can be obtained quickly using constant symbols:

```
%MIN SIZE
E(a1) & E(a2) & E(a3) & E(a4) & E(a5) & E(a6).
a1!=a2 & a1!=a3 & a1!=a4 & a1!=a5 & a1!=a6.
a2!=a3 & a2!=a4 & a2!=a5 & a2!=a6.
a3!=a4 & a3!=a5 & a3!=a6.
a4!=a5 & a4!=a6.
a5!=a6.
```

Although straightforward, requiring all constants to be interpreted distinctly is somewhat tedious compared to other axioms since all pairwise comparisons have to be written out explicitly.

Given our formalization, it is an easy task to formalize the *Gärdenfors principle* and *independence* starting from their representation that was introduced in Section 1.4:

```
%GF
S(u) & E(y) -> ( (all x (E(x) & el(x,u) -> l(y,x) & -l(x,y))) ->
w(union(u,singleton(y)),u) & -w(u,union(u,singleton(y))) ).
S(u) & E(y) -> ( (all x (E(x) & el(x,u) -> l(x,y) & -l(y,x))) ->
w(u,union(u,singleton(y))) & -w(union(u,singleton(y)),u) ).
```

Note that we use `l(x,y) & -l(y,x)` and `w(u,v) & -w(v,u)` to refer to the strict versions of the orders `l` and `w`, respectively.

```
%IND
S(u) & S(v) & E(z) -> ( (w(u,v) & -w(v,u) & -el(z,union(u,v))) ->
w(union(u,singleton(z)),union(v,singleton(z))) ).
```

With these additional axioms any model satisfying all of the above axioms will contain a (finite) counterexample to the Kannai-Peleg Theorem, and any counterexample, viz. a weak order on $\mathcal{X}$ fulfilling the *Gärdenfors principle* and *independence*, will result in such a model. Consequently, we can — denoting the Kannai-Peleg axioms by Γ_{KP} — reformulate the Kannai-Peleg Theorem as:

Theorem 2.1 (formerly Theorem 1.6). *The theory* $\Gamma_{\mathrm{frame}} \cup \Gamma_{\mathrm{KP}}$ *is inconsistent, i.e., it does not have any models.*

This should now theoretically be checkable by a first-order theorem prover and we are going to report our — unfortunately negative — results in the next section.

2.3 Experimental (Practical) Results

All our experiments were carried out on 2.26 GHz Intel Xeon machines with 24GB of main memory.[18] We ran our formalization of the Kannai-Peleg Theorem on the three different first-order theorem provers *Prover9* [44] (successor of *Otter* [43]), *Prover E* [63], and *iProver* [38],[19] but neither of them was able to produce a proof within the given time limit of five days running time.[20]

[18] The machines are part of the Dutch national compute cluster *Lisa* [58].

[19] *Prover E* and *iProver* won the second and fifth price, respectively, in the first-order formula category of the *CADE ATP System Competition 2009* [69]. The first and third price provers are versions of the *Vampire* prover [76], which is not publicly available.

[20] We used the provers in fully automatic mode and the maximally allowed memory limits. Only in the case of *Prover E* this required special command line options, which are `-tAuto -xAuto --tstp-in --resources-info --memory-limit=2048`.

Therefore, we decided to also create a few (easier) problem variants and subproblems to achieve a partial success at least. As an additional problem we chose for (the quite straightforward) Proposition 1.2, which says that the *Kelly principle* is a consequence of the *Gärdenfors principle*, and exemplarily formalized the two subcases (`GF`) $\Rightarrow$ (`EXT`) and (`GF`) $\Rightarrow$ (`MAX`) (see Section 1.3 for the exact statements). Their formalization, which we are going to display and explain next, requires a slightly extended framework, because the Kelly principle contains statements about maximal and minimal elements of a set.

We can define the functions *max* and *min* in a straightforward fashion, but have to also set the corresponding sorts for the functions, i.e., that only applied to sets they yield an element and in all other cases just the special object 0. The actual definitions then just say that an element is the maximum/minimum of a set if and only if it belongs to the set and all other elements in the set are ranked strictly lower/higher by $\dot{\geq}$.

```
%SORT2
S(u) & min(u)=x -> E(x).
-S(u) -> min(u)=0.
S(u) & max(u)=x -> E(x).
-S(u) -> max(u)=0.

%min
S(u) & E(x) -> ( min(u)=x <-> el(x,u) & (all y ((el(y,u) & y!=x) ->
l(y,x) & -l(x,y))) ).

%max
S(u) & E(x) -> ( max(u)=x <-> el(x,u) & (all y ((el(y,u) & y!=x) ->
l(x,y) & -l(y,x))) ).
```

With these additional framework axioms, we can now create different modifications of the problem. One possibility is to drop independence and add a goal formula for the axiom (`MAX`) or (`EXT`), respectively, in order to prove that these follow from (`GF`).[21]

```
%GOAL: MAX
S(u) -> w(singleton(max(u)),u).

%GOAL: EXT
E(x) & E(y) -> ((l(x,y) & -l(y,x)) -> ( w(singleton(x),singleton(y)) &
-w(singleton(y),singleton(x)) )).
```

Another option to create simpler problems is to split the Kannai-Peleg Theorem in two steps: first derive Lemma 1.4 (saying that sets are considered equally good as the set containing just their minimum and maximum),

[21] Adding a formula as a goal technically means to add its negation to the assumptions and then create a proof by contradiction.

and then prove the inconsistency with this lemma as an assumption. Technically, this means to add the lemma as a goal formula in the first, and as an assumption in the second case.

```
%LEMMA
S(u) -> w(u,union(singleton(min(u)),singleton(max(u)))) &
w(union(singleton(min(u)),singleton(max(u))),u).
```

Additionally varying the minimal and maximal domain sizes of X, which can be achieved by an axiom like `MAX SIZE` (as shown below) and by modifying the above axiom `MIN SIZE`, respectively, we have created a selection of problem variants and subproblems.

```
%MAX SIZE
E(x) -> x=a6 | x=a5 | x=a4 | x=a3 | x=a2 | x=a1.
```

The problems we finally experimented with are given in Table 2.1 together with the performance of *Prover E*, which provided the best results among the three provers tested.

Theorem	Options			TTP
	min_el	max_el	Lemma 1.4	
(GF) ⇒ (EXT)				40:08:46
	2	2		109:38:39
(GF) ⇒ (MAX)				n/a
Lemma 1.4				n/a
	2	2		00:00:35
	3	3		n/a
Kannai-Peleg	(6)			n/a
	(6)	6	given	n/a
	(6)	6		n/a
	(6)		given	n/a

Table 2.1. Time until a proof was found (time to proof, TTP) with *Prover E* (at a maximum running time of 120 hours wall clock).

We can see here that apart from very elementary results – like the implication (GF) ⇒ (EXT) or special cases for two elements only – the automated theorem provers were not able to produce proofs for the given problems. At first sight this might appear to be somewhat surprising considering the simplicity of some of the results. On the other hand, a comparison of the constructed proof of Lemma 1.4 for two elements (see Appendix A.1), which consists of 710 lines, to the one-line proof we have given in Section 1.4.2 provides a possible explanation. It seems to be the case that the explicit implementation of "set theory" creates such a large overhead that even the simplest proofs become very complicated. Already the construction of small sets (as they appear a lot in the proof of the Kannai-Peleg Theorem) is quite complex as the following example shows. Consider the simple statement

$$\{x_2, x_5, x_6\} \sim \{x_2, x_3, x_4, x_5, x_6\},$$

which is taken from the proof of the Kannai-Peleg Theorem (see Theorem 1.6). In the framework as presented above, this line has to be expressed in a much more complicated way by repeated function application to create the sets:

```
w(union(union(singleton(a2),singleton(a5)),singleton(a6)),
  union(union(union(union(singleton(a2),singleton(a3)),singleton(a4)),
              singleton(a5)),singleton(a6)))
&
w(union(union(union(union(singleton(a2),singleton(a3)),singleton(a4)),
              singleton(a5)),singleton(a6)),
  union(union(singleton(a2),singleton(a5)),singleton(a6)))
```

That the prover needed longer to build a proof for the special case of two elements in (GF) $\Rightarrow$ (EXT) than for the general results shows how additional axioms can make the search for a proof more difficult, even though they are supposed to make the task easier.

With these observations in mind, it is not so surprising anymore that automated provers struggle to create proofs in this particular framework. Thus, we are also not very optimistic that faster or different systems can lead to automatically created first-order proofs easily, unless one either finds a prover that deals particularly well with this structure, or one develops a modification of the framework that does not create so much overhead.

2.4 Summary

We have seen in this chapter that a first-order formalization of the Kannai-Peleg Theorem is possible, but does unfortunately not lead to the desired automated verification when given to first-order theorem provers.

The chapter provided an explanation of the formalization of the framework of *ranking sets of objects* and of all axioms involved in the Kannai-Peleg Theorem. We additionally crafted some further (simpler) problems in order to explore the capabilities of the theorem provers with respect to *ranking sets of objects*. Also their axioms were presented.

Table 2.1 exhibits the mostly negative results of our experiments with *Prover E*. Other provers performed even worse on our formalization.

We have conjectured that the problems with automatically finding a proof are mostly due to the large overhead created in the formalization by artificially creating the set-element structure, in which sets are considered as elements of the domain and therefore the size of the universe increases exponentially. Thus, even if faster or better systems are eventually able to derive proofs automatically in this framework, these objects will be highly unreadable as the proof given in Appendix A.1 of an easy statement showed.

In the next chapter we are going to show, however, that a different approach employing propositional logic (instead of first-order logic) can overcome some of the practical difficulties and will succeed in providing an automated verification of the Kannai-Peleg Theorem.

3

Theorem Proving Using a SAT Solver

3.1 Introduction

The approach to automated theorem proving in the field of social choice theory presented in this chapter is due to Tang and Lin, who have proved some major impossibility results (Arrow, Muller-Satterthwaite, Sen): using manual inductive lemmas to reduce the theorems to small instances, they finally use a *satisfiability checker* (or: *SAT solver*) to verify these base cases. Whereas in their proceedings- [39] and their full paper on this approach [73] they concentrate on the inductive steps and proofs thereof, we present this novel proof technique in this chapter by reporting their results for Arrow's Theorem and then extending and translating it to the setting of the Kannai-Peleg Theorem with a strong focus on the computer-aided part, which was only introduced on a rather conceptual level by Tang and Lin. We present the full axiomatization and describe the necessary steps to obtain a representation that is accepted as the input to a SAT solver, which then does the actual verification.

What makes the new approach most valuable is the fact that base case verification appears to be very efficient and flexible in the sense that it is very easy to modify the axioms used in the theorems in order to look for stronger or completely new results. This can be done, not only because the result for the base case is potentially a good indication on whether to hope for the full result, but also since in the cases where the inductive argument carries over it will even yield the full result. Tang and Lin have already started working in this direction and were able to weaken Arrow's conditions, i.e., *strengthen* the impossibility theorem. Also we have engaged in this kind of research and will report our findings about a universal step and an automated and exhaustive proof search in Chapters 4 and 5.

It is remarkable that Tang and Lin have been able to formulate the whole base case of Arrow's Theorem in propositional logic only; even though some of the axioms intuitively are second-order statements. The trick they used was to introduce "situations" as names for preference profiles, which transforms the second-order axioms into first-order statements, which can then (because of the finiteness of the base case) be translated into propositional logic. We are going to use a very similar approach as the axioms of the Kannai-Peleg Theorem are stated in somewhat enriched[22] second-order, too. The setting of the Kannai-Peleg Theorem will, however, require a somewhat different treatment since we also have to apply functions like union ($\cup$) and singleton set ($\{\cdot\}$) to sets and elements, respectively, whereas no functions needed to be applied to Tang and Lin's situations.

This chapter is structured into four sections. After this introduction we are first reviewing Tang and Lin's new proof of Arrow's Theorem, before we, in Section 3.3, turn to how their approach can be extended in order to be applicable to the framework of *ranking sets of objects* in general, and the Kannai-Peleg Theorem in particular. We are further going to lay out the full formalization of this theorem and give details regarding our program and some of its implementation-specific features. The last section of this chapter will be devoted to additional insights and features, which our program and method can offer.

3.2 Tang and Lin's Proof Technique

As indicated, we are first going to recapitulate and briefly demonstrate Tang and Lin's initial result of re-proving Arrow's Theorem [5].[23] Before we can show how they applied the new proof method to this particular theorem, however, we have to briefly introduce some notation and concepts regarding its substance: preference aggregation. After this, we will state Arrow's Theorem, give the inductive steps due to Tang and Lin, and, finally, exemplify the base case formalization for the axiom of non-dictatorship.

3.2.1 Social Welfare Functions

Social welfare functions are the core concept of Arrow's Theorem and, roughly speaking, describe how n individuals can aggregate their individual preferences (assumed to be strict linear orders) to obtain a single common preference, the social preference.

We use the following notation/definitions: A finite set of *alternatives* will be denoted by A; and $I = \{1, \ldots, n\}$ shall be a finite set of *individuals*. As stated above we will only consider preferences in the form of strict linear orders (*preference orderings*) over all alternatives, and we write $\mathcal{P}$ for the set of all these. Every voter has an *individual preference ordering* $P_i \in \mathcal{P}$ and so the full picture will be an n-tuple $\langle P_1, P_2, \ldots, P_n \rangle \in \mathcal{P}^n$, a *preference profile*. The unique alternative that is ranked highest in a preference ordering P will be called $top(P)$.

[22] There is also an order (i.e., a relation) on sets.

[23] They also successfully used their technique to automatically verify the Muller-Satterthwaite Theorem [46] as well as Sen's Theorem [64].

With these prerequisites in place, we are able to make the concept of a social welfare function precise as well as some properties that social welfare functions can have and which we need to state Arrow's Theorem.

Definition 3.1. A *social welfare function* is a function $f : \mathcal{P}^n \to \mathcal{P}$, which assigns to any preference profile $s = \langle P_1, P_2, \ldots, P_n \rangle$ a unique (social) preference ordering $f(s) \in \mathcal{P}$.

Definition 3.2. A social welfare function $f : \mathcal{P}^n \to \mathcal{P}$ is

1. *unanimous* (or *Pareto efficient*), if, whenever in a preference profile $s = \langle P_1, P_2, \ldots, P_n \rangle$ all individuals agree on the ordering of two alternatives $a, b \in A$, i.e., whenever $a >_{P_i} b$ for all $i \in I$, then the social welfare function copies this ranking, i.e., $a >_{f(s)} b$.
2. *independent of irrelevant alternatives* (IIA), if, whenever two preference profiles $s = \langle P_1, P_2, \ldots, P_n \rangle$, $t = \langle P'_1, P'_2, \ldots, P'_n \rangle$ agree on the ordering of two alternatives $a, b \in A$ per individual, i.e., if $a >_{P_i} b \iff a >_{P'_i} b$ for all $i \in I$, then the social ordering of a and b agrees for both preference profiles, i.e., $a >_{f(s)} b \iff a >_{f(t)} b$.
3. *non-dictatorial*, if there is *no* individual $i \in I$ such that for any preference profile $s = \langle P_1, P_2, \ldots, P_n \rangle$ the social welfare function just picks individual i's preference ordering, i.e., $f(s) = P_i$.

A social welfare function $f : \mathcal{P}^n \to \mathcal{P}$ that has all of the three above properties will be said to *satisfy Arrow's conditions*.

3.2.2 Arrow's Theorem and the Reduction to its Base Case

Having set the stage, we are now in the position to state Arrow's Theorem and to report the inductive results due to TANG and LIN, which have made the new computer-aided approach possible at all.

Theorem 3.3 (ARROW, 1950). *There is no social welfare function for three or more alternatives that is unanimous, IIA and non-dictatorial.*

TANG and LIN have proved the following two lemmas, which together form the inductive step in their proof of Arrow's Theorem:[24]

Lemma 3.4 (TANG and LIN, 2008). *For $n \leq 2, m \leq 3$, if there is a social welfare function for n individuals and* **m + 1 alternatives** *satisfying Arrow's conditions, then there is one for n individuals and just* **m alternatives** *that also satisfies all of Arrow's conditions.*

[24] It is an interesting fact that already in his original paper [5] from 1950 ARROW proved his theorem for the base case only and correctly mentions that this is sufficient as it carries over to larger instances. But since his paper is very brief about this detail, we do not know whether he was aware of the possibility of inductive lemmas or was just thinking of his direct general proof, which was published later [6].

He certainly did not provide any inductive step there.

Lemma 3.5 (TANG **and** LIN**, 2008).** *For $n \leq 2, m \leq 3$, if there is a social welfare function for* $\mathbf{n+1}$ **individuals** *and m alternatives satisfying Arrow's conditions, then there is one for just* **n individuals** *and m alternatives that also satisfies all of Arrow's conditions.*

Reading these two lemmas together and contrapositively, we get that if there is no social welfare function satisfying Arrow's conditions for two individuals and three alternatives, then there is no such function for any number of individuals and alternatives ($|A| \geq 3, |I| \geq 2$). But note that, strictly speaking, we would have to consider the case of exactly *one* individual as well. It is, however, very clear that in this case any unanimous social welfare function is automatically dictatorial.

Hence, only a small base case remains to be checked:

Lemma 3.6 (base case of Arrow's Theorem). *For exactly* **three alternatives** *and* **two individuals** *there is no social welfare function that is unanimous, IIA and non-dictatorial.*

As mentioned before, TANG and LIN used a computer to verify this base case. They observed that a direct check of all $|\mathcal{P}|^{\left(|\mathcal{P}|^2\right)} = 3!^{\left(3!^2\right)} = 6^{36} \approx 10^{28}$ possible social welfare functions is, of course, not tractable. Hence they treated this problem in two distinct and clever ways: First, they explicitly generated all social welfare functions that are unanimous and IIA, and then checked for non-dictatorship, all implemented as a constraint satisfaction problem with a backtracking depth-first search (see TANG and LIN [39, 73] for details).

This approach, however, is not nearly as universal as their second technique, which we are going to concentrate on: a formalization of all axioms and additional conditions in *propositional logic* and then using a SAT solver to verify the result of the formalization. Following their brief description, we were able to reproduce their results for the base case of Arrow's Theorem. We will briefly demonstrate the mechanics of this technique here; but then, rather quickly, move forward and extend it in such a way that it applies to the Kannai-Peleg Theorem.

3.2.3 Formalization of the Base Case of Arrow's Theorem

The trick, in order to convert what are intuitively second-order axioms into *propositional logic*, is moving to a kind of situation calculus, where preference profiles are treated as primitives. TANG and LIN did this by introducing a "name" $s_{\boldsymbol{P}} \in S$ for each preference profile $\boldsymbol{P} \in \mathcal{P}^n$, and then calling these names "situations". This enabled them to use predicates $p(x, a, b, s)$ and $w(a, s)$, which intuitively stand for individual x ranking alternative a higher than alternative b in situation s, and alternative a being ranked above alternative b in situation s according to the social welfare function, respectively. A complete list of the axioms involved can be found in TANG and LIN [73] and we are only demonstrating the conversion for the axiom of non-dictatorship here (where a, b stand for alternatives in A; i for individuals in I; and s for situations in S):

$$\begin{aligned} \texttt{(ND)} \qquad & \neg\exists i \forall s\,(f(s) = P_i) \\ & \equiv \neg\exists i \forall s, a, b\,(p(i, a, b, s) \leftrightarrow w(a, b, s))\,. \end{aligned}$$

For the additional requirement of an unrestricted domain, i.e., that every preference profile is indeed represented by a situation, Tang and Lin remark that this is coded easiest by representing the $3!^2 = 36$ situations (a situation here represents a pair of strict linear orders, i.e., a pair of permutations, of the three alternatives) explicitly:[25]

For all pairs of permutations $\langle \pi, \rho \rangle$ of the alternatives $a, b, c \in A$ (these are in a direct one-to-one correspondence with the strict linear orders on A) we can define a corresponding situation $s_{\pi,\rho}$ by

$$\begin{aligned} \texttt{(SIT)} \qquad & p(1, \pi_1, \pi_2, s_{\pi,\rho}) \wedge p(1, \pi_2, \pi_3, s_{\pi,\rho}) \wedge \\ & p(2, \rho_1, \rho_2, s_{\pi,\rho}) \wedge p(2, \rho_2, \rho_3, s_{\pi,\rho}). \end{aligned}$$

Observe that this is completely sufficient, since with the axioms defining p as a strict linear order, the other formulas like $p(1, \pi_1, \pi_3, s_{\pi,\rho})$ and $\neg p(1, \pi_2, \pi_1, s_{\pi,\rho})$ follow by transitivity and antisymmetry. Alternatively, one could also have used the full description (of $36 \cdot 2 \cdot 3^2 = 648$ literals) instead of the axioms defining the strict linear order.[26]

The subsequent conversion to propositional logic is relatively straightforward when interpreting the predicates $p(i, a, b, s)$ and $w(a, b, s)$ as propositional variables ($p_{i,a,b,s}$ and $w_{a,b,s}$) indexed by their arguments. As the domains of the quantifiers are finite, these can then be translated to finite conjunctions or disjunctions respectively. Hence, this whole procedure yields finite formulas only with at most 972 different propositional variables (since there are exactly $2 \cdot 3 \cdot 3 \cdot 36 + 3 \cdot 3 \cdot 36 = 972$ combinations of indices).

As an example, consider the formula for non-dictatorship again, which will then have the form:

[25] They also describe a way of formalizing the unrestricted domain by means of an action $swap(x, a, b)$, which interchanges the positions of a and b in the preference ordering of individual x and thus yields any possible preference ordering from an initial one S_0 by a sequence of swapping operations. But even though this idea is useful for first-order formulations (see for instance [32]), it is unclear how to convert it to propositional logic, and, therefore, the direct coding of situations is much easier here.

[26] Tang and Lin chose an intermediate formalization and also explicitly gave the two propositions $p(1, \pi_1, \pi_3, s_{\pi,\rho})$ and $p(2, \rho_1, \rho_3, s_{\pi,\rho})$, which would also have followed by transitivity.

$$\begin{aligned}
&\neg\exists i \forall s, a, b\, (p(i,a,b,s) \leftrightarrow w(a,b,s)) \\
\equiv\ &\neg\exists i \forall s, a, b\, (p_{i,a,b,s} \leftrightarrow w_{a,b,s}) \\
\cong\ &\neg \bigvee_i \bigwedge_{s,a,b} (p_{i,a,b,s} \leftrightarrow w_{a,b,s}) \\
\equiv\ &\bigwedge_i \bigvee_{s,a,b} \neg\, (p_{i,a,b,s} \leftrightarrow w_{a,b,s})\, .
\end{aligned}$$

There is one final step in the translation process which leads to problems as the formula for non-dictatorship exhibits: the standard conversion to conjunctive normal form[27] (applying De Morgan's Law to move negations inwards and then using the distributive law) might only lead to an equivalent formula that is exponentially larger than the original formula.[28] Hence we use the well known result that there is an equisatisfiable[29] formula that is only linearly larger. The technique to obtain such a formula is to introduce new (auxiliary) variable symbols representing whole parts of the formula to convert. In the example of the formula for non-dictatorship this means that we replace $\neg\,(p_{i,a,b,s} \leftrightarrow w_{a,b,s})$ by a single variable $h_{i,a,b,s}$ and add the additional defining formulas for all i, a, b, s:

$$\begin{aligned}
&h_{i,a,b,s} \leftrightarrow \neg\, (p_{i,a,b,s} \leftrightarrow w_{a,b,s}) \\
\equiv\ &(h_{i,a,b,s} \vee \neg w_{a,b,s} \vee p_{i,a,b,s}) \wedge (h_{i,a,b,s} \vee w_{a,b,s} \vee \neg p_{i,a,b,s}) \wedge \\
&(\neg h_{i,a,b,s} \vee w_{a,b,s} \vee p_{i,a,b,s}) \wedge (\neg h_{i,a,b,s} \vee \neg w_{a,b,s} \vee \neg p_{i,a,b,s}).
\end{aligned}$$

We then, like Tang and Lin, instantiate the whole set of propositional formulas using a computer, which yields a single formula in conjunctive normal form with, in our case, a total of 36,612 different variables and 158,582 clauses. If we denote this formula by φ_{Arrow} the base case of Arrow's Theorem can be restated as the following simple lemma:

Lemma 3.7 (base case restated). *The formula φ_{Arrow} is unsatisfiable.*

In order to verify this lemma we — again like Tang and Lin — fed φ to the SAT solver *zChaff* [61] (an implementation of the Chaff algorithm by Moskewicz et al. [45]), which returned surprisingly quickly (in less than one second on a MacBook with 2.2GHz and 2GB RAM) that φ is not satisfiable. This is precisely the desired result that Tang and Lin had obtained before.

[27] A propositional formula is in *conjunctive normal form* if it is a conjunction of disjunctions of literals, where literals are variable symbols or negations thereof. Disjunctions of literals are also called *clauses*.

[28] Consider the example $\psi = (X_1 \wedge Y_1) \vee (X_2 \wedge Y_2) \vee \cdots \vee (X_n \wedge Y_n)$, whose normal form obtained by using the standard rules for conversion is $\psi' = (X_1 \vee \cdots \vee X_{n-1} \vee X_n) \wedge (X_1 \vee \cdots \vee X_{n-1} \vee Y_n) \wedge \cdots \wedge (Y_1 \vee \cdots \vee Y_{n-1} \vee Y_n)$.

[29] Two formulas φ, ψ are equisatisfiable if and only if both formulae are satisfiable or both are not. In other words, one formula has to be satisfiable if and only if the other one is satisfiable.

3.3 Extended Approach for the Kannai-Peleg Theorem

First, recall the notation used in Chapter 1 and the Kannai-Peleg Theorem as stated there in Section 1.4.2:

Theorem 3.8 (formerly Theorem 1.6). *Let X be a linearly ordered set with $|X| \geq 6$. Then there exists no weak order $\succeq$ on $\mathcal{X}$ satisfying the Gärdenfors principle* (`GF`) *and independence* (`IND`).

Even though the setting of the Kannai-Peleg Theorem appears to be very similar to the one of Arrow's Theorem — in that it is also an impossibility theorem about finite but arbitrarily large domains that uses second-order quantification in its axioms — we could not find a way to directly apply Tang and Lin's method as it is. This is due to the requirement of having functions and operators on X (such as *singleton set*) and on the space of subsets $\mathcal{X}$ (such as *union* and *complement*), respectively, which we could not directly encode in the language proposed by Tang and Lin. There is, however, a way to avoid these issues: instead of coding sets *within* the language, we let the program that generates the final formula handle them.

But before we get to those issues involved in the formalization of the base case, we should first reduce the problem to its base case by stating and proving the inductive step, which we are going to do in the following subsection.

3.3.1 Inductive Step

Somewhat different from Arrow's Theorem, the inductive proof of the Kannai-Peleg Theorem only requires one inductive step since the outmost universal quantification only ranges over one natural number (the cardinality of X) instead of two (the number of individuals and the number of alternatives). Recall that for the case of Arrow's Theorem the idea was to construct a social welfare function for a smaller domain from a larger one by somehow restricting it. The very same idea is used here, where we use the usual restriction of a relation to a subset of its domain and show that it still fulfills the necessary axioms.

Lemma 3.9. *If X is a linearly ordered set with $n+1$ elements ($n \in \mathbb{N}$) and there is a corresponding weak order $\succeq$ on $\mathcal{X}$ that satisfies the Gärdenfors principle* (`GF`) *and independence* (`IND`)*, then we can find another linearly ordered set Y with n elements only, as well as a corresponding weak order on $\mathcal{Y} := 2^Y \setminus \{\emptyset\}$ satisfying the same axioms* (`GF`) *and* (`IND`).

Proof. Let n be a natural number and X be a set with $n+1$ elements. Without loss of generality, we may write $X = X_n \mathbin{\dot\cup} \{x_{n+1}\}$ as the disjoint union of an n-element set $X_n = \{x_1, x_2, \ldots, x_n\}$ and the singleton $\{x_{n+1}\}$. Let furthermore $\succeq$ be a weak order on $\mathcal{X}$ that satisfies the Gärdenfors principle (`GF`) and independence (`IND`). It is clear that $Y := X_n$ is a linearly ordered set containing n elements and thus we are going to show that the restriction $\succeq_n$ of $\succeq$ to $\mathcal{Y} = 2^Y \setminus \{\emptyset\} \subseteq \mathcal{X}$ is a weak order again and also fulfills the axioms (`GF`) and (`IND`).

Formally, we define the restriction $\succeq_n$ of $\succeq$ to $\mathcal{Y}$ by setting

$$A \succeq_n B \iff A \succeq B \tag{3.1}$$

for all $A, B \in \mathcal{Y}$.

Showing that all three axioms for $\succeq_n$ being a weak order are true is straightforward.[30] Therefore, we only give the proof for transitivity as an exemplification: let A, B, C be sets in $\mathcal{Y}$ such that $A \succeq_n B$ and $B \succeq_n C$. Reading (3.1) from left to right, we see that also $A \succeq B$ and $B \succeq C$ hold, and since A, B, C are subsets of $Y \subseteq X$, they are also in $\mathcal{X}$. Hence, we can use transitivity of $\succeq$ and obtain $A \succeq C$, which finally — reading (3.1) from right to left — gives $A \succeq_n C$.

Now, let us turn to the Gärdenfors principle (`GF`) and independence (`IND`):

- (`GF`): We show the claim for (`GF1`); the proof for (`GF2`) works analogously. Let $A \in \mathcal{Y}$ and $x \in Y$ be a non-empty subset and an element of Y, respectively, such that $x \dot{>} a$ for all $a \in A$. Since A is also a member of $\mathcal{X}$ and so is x of X, we can use (`GF1`) for $\succeq$, which yields
$$A \cup \{x\} \succ A.$$
From this it directly follows by the definition of $\succeq_n$ — as given in (3.1) — that $A \cup \{x\} \succ_n A$ since both $A \subseteq Y$ and $A \cup \{x\} \subseteq Y$ are in $\mathcal{Y}$.
- (`IND`): Let $A, B \in \mathcal{Y}$ be non-empty subsets of Y such that $A \succ_n B$. Let furthermore $x \in Y \setminus A \cup B$ be an element of Y that is neither already contained in A nor in B. Since A and B are clearly also members of $\mathcal{X}$, and similarly is x of $X \setminus A \cup B$, we can use (`IND`) for $\succeq$ to obtain
$$A \cup \{x\} \succeq B \cup \{x\}.$$
Again, the very definition of $\succeq_n$ — as given in (3.1) — implies that also $A \cup \{x\} \succeq_n B \cup \{x\}$; and so we are done. □

The similarities between the parts of the proof for different axioms already suggest the existence of a more general inductive step, which is applicable to a whole class of axioms; and indeed, in Chapter 4, we are going to prove such a theorem and explore its consequences, which finally enable an automated theorem search as described in Chapter 5. But for now, Lemma 3.9 suffices and reading it contrapositively yields the following corollary, which is the final form of the inductive step for the Kannai-Peleg Theorem.

[30] Actually, these and also the axioms (`GF`) and (`IND`) are automatically true by their mere logical form together with the fact that $Y \subseteq X$ and $\mathcal{Y} = 2^Y \setminus \{\emptyset\}$. This idea will be explored further and proved in Chapter 4.

Corollary 3.10. *If, for any linearly ordered set Y with n elements, there exists no weak order on $\mathcal{Y} = 2^Y \setminus \{\emptyset\}$ satisfying the Gärdenfors principle* (`GF`) *and independence* (`IND`), *then also for any linearly ordered set X with $|X| = n+1$ there is no weak order on $\mathcal{X} = 2^X \setminus \{\emptyset\}$ that satisfies these axioms.*

Proof. This corollary is nothing but the contraposition of Lemma 3.9. □

The remaining base case, which will be stated in the next section below, now has to be converted to propositional logic such that it can be fed to a SAT solver. In the next subsection we will therefore describe all the necessary steps to transform the axioms into a propositional language as it is accepted by a SAT solver.

3.3.2 Base Case Formulation in Propositional Logic

By the inductive step as exhibited and proved in Lemma 3.9 we are only left with a rather small base case of six elements:

Lemma 3.11 (Base case of the Kannai-Peleg Theorem). *Let X be a linearly ordered set with exactly* 6 *elements. Then there exists no weak order $\succeq$ on $\mathcal{X}$ satisfying the Gärdenfors principle* (`GF`) *and independence* (`IND`).

It might seem tempting to perform a direct check of the involved axioms on all weak orders over the non-empty subsets of a six-element space. This, however, can be seen to be practically impossible as there are $p(2^{|X|} - 1) = p(2^6 - 1) = p(63) \approx 1.5254 \cdot 10^{97}$ such orderings.[31]

Therefore, we should stick to the idea of transforming the axioms of the Kannai-Peleg Theorem such that they can be checked by a SAT solver which operates on propositional formulas in conjunctive normal form only. We will describe in the following how instances of the axioms as stated in Sections 1.3 and 1.4 can be converted to that language.

The main challenge in the conversion of the axioms, again, is the second-order quantification (this time paired with functions and relations like *union* and *element of*) as it appears, for instance, in the Gärdenfors principle (`GF`) and independence (`IND`). In contrast, first-order quantifications are not problematic because of the finite instances we are dealing with: we can simply replace universal and existential quantifiers by conjunctions and disjunctions, respectively.

It will be sufficient for our formalization to have two kinds of propositions only: $w(A, B)$ and $l(x, y)$ (corresponding to propositional variables $w_{A,B}$ and

[31] The notation $p(n)$ is used for the number of weak orders on a finite set with n elements. This number can be computed following Bailey [8], or via the corresponding sequence `A000670` in the on-line encyclopedia of integer sequences [66]. The exact number (calculated with *Mathematica* [79]) of weak orderings is 1525 3777976254320482680028798302887920587781886722393040786663126590823609598384497028003529713213.

$l_{x,y}$) with intended meanings A *is ranked at least as high as* B *by the weak order* $\succeq$ (or short: $A \succeq B$), and x *is ranked at least as high as* y *by the linear order* $\dot{\geq}$ (or short: $x \mathrel{\dot{\geq}} y$), respectively. This then leads to a maximum of $|\mathcal{X}|^2 + |X|^2 = (2^6 - 1)^2 + 6^2 = 4005$ different propositional variables for the base case of the Kannai-Peleg Theorem.

As indicated already, the axioms for a *linear order* on X are entirely unproblematic as they only contain first-order quantifications and, thus, they can be transformed as follows:

$$\texttt{(LIN)}\quad (\forall x \in X)\ x \mathrel{\dot{\geq}} x \qquad \text{(reflexivity)}$$
$$\cong \bigwedge_{x \in X} l_{x,x},$$

$$(\forall x \in X)(\forall y \in X)\left[x \neq y \to x \mathrel{\dot{\geq}} y \vee x \mathrel{\dot{\leq}} y\right] \qquad \text{(completeness)}$$
$$\cong \bigwedge_{x \in X} \bigwedge_{y \in X} [x \neq y \to l_{x,y} \vee l_{y,x}]$$
$$\equiv \bigwedge_{x \in X} \bigwedge_{\substack{y \in X\\ y \neq x}} [l_{x,y} \vee l_{y,x}]\,,$$

$$(\forall x \in X)(\forall y \in X)(\forall z \in X)\left[x \mathrel{\dot{\geq}} y \wedge y \mathrel{\dot{\geq}} z \to x \mathrel{\dot{\geq}} z\right] \qquad \text{(transitivity)}$$
$$\cong \bigwedge_{x \in X} \bigwedge_{y \in X} \bigwedge_{z \in X} [(l_{x,y} \wedge l_{y,z}) \to l_{x,z}]$$
$$\equiv \bigwedge_{x \in X} \bigwedge_{y \in X} \bigwedge_{z \in X} [\neg\,(l_{x,y} \wedge l_{y,z}) \vee l_{x,z}]$$
$$\equiv \bigwedge_{x \in X} \bigwedge_{y \in X} \bigwedge_{z \in X} [\neg l_{x,y} \vee \neg l_{y,z} \vee l_{x,z}]\,,$$

$$(\forall x \in X)(\forall y \in X)\left[\left(x \mathrel{\dot{\geq}} y \wedge y \mathrel{\dot{\geq}} x\right) \to x = y\right] \qquad \text{(antisymmetry)}$$
$$\cong \bigwedge_{x \in X} \bigwedge_{y \in X} [(l_{x,y} \wedge l_{y,x}) \to x = y]$$
$$\equiv \bigwedge_{x \in X} \bigwedge_{y \in X} [x \neq y \to \neg\,(l_{x,y} \wedge l_{y,x})]$$
$$\equiv \bigwedge_{x \in X} \bigwedge_{\substack{y \in X\\ y \neq x}} [\neg l_{x,y} \vee \neg l_{y,x}]\,.$$

We should remark here that, by the above conversion, we even got rid of the equalities and inequalities by treating them within the quantification domains.

The axioms for a *weak order* on $\mathcal{X}$, which only contain simple second-order quantifications,[32]

$$\begin{aligned} (\texttt{WEAK}) \quad & (\forall A \in \mathcal{X})\ A \succeq A, \\ & (\forall A \in \mathcal{X})(\forall B \in \mathcal{X})\,[A \neq B \rightarrow A \succeq B \vee A \preceq B]\,, \\ & (\forall A \in \mathcal{X})(\forall B \in \mathcal{X})(\forall C \in \mathcal{X})\,[A \succeq B \wedge B \succeq C \rightarrow A \succeq C] \end{aligned}$$

can similarly be rewritten as

$$\begin{aligned} (\texttt{WEAK}) \quad & \bigwedge_{A \in \mathcal{X}} w_{A,A}, \\ & \bigwedge_{A \in \mathcal{X}} \bigwedge_{\substack{B \in \mathcal{X} \\ B \neq A}} [w_{A,B} \vee w_{B,A}]\,, \\ & \bigwedge_{A \in \mathcal{X}} \bigwedge_{B \in \mathcal{X}} \bigwedge_{C \in \mathcal{X}} [\neg w_{A,B} \vee \neg w_{B,C} \vee w_{A,C}]\,. \end{aligned}$$

Note, that, due to the finiteness of X (and thus also $\mathcal{X}$), the derived formulas above are actually finite objects and can therefore be instantiated by hand (which would require a lot of effort) or using a computer. Furthermore, only very little or no work was needed to convert them to conjunctive normal form, as we can see from the conversions above.

The main axioms (GF) and (IND) appear to be more difficult to transform completely because of functions like singleton set $\{\cdot\} : X \rightarrow \mathcal{X}$ and set union $\cup : \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{X}$, which occur within these axioms. In fact, however, the same simple conversion technique as above can be applied since we are going to take care of those (and similar) functions automatically in our computer program for the instantiation of the axioms. The exact way of how we do this will be described in the following Section 3.3.3 and for now we treat terms like $A \cup B$ as if they were just the corresponding objects from the functions range, i.e., images under the respective functions.

This leads to the following conversion of (GF1):

[32] Simple, in the sense that the variables bound by them only occur in order relations and (in-)equalities, but not in functions (as we will see is the case for (GF) and (IND)).

$$
\begin{aligned}
(\texttt{GF1}) \quad & (\forall A \in \mathcal{X})(\forall x \in X)[((\forall a \in A) x \mathrel{\dot{>}} a) \to A \cup \{x\} \succ A] \\
& \equiv (\forall A \in \mathcal{X})(\forall x \in X)[((\forall a \in A) x \mathrel{\dot{\geq}} a \wedge a \mathrel{\dot{\ngeq}} x) \to \\
& \qquad\qquad (A \cup \{x\} \succeq A \wedge A \not\succeq A \cup \{x\})] \\
& \cong \bigwedge_{A \in \mathcal{X}} \bigwedge_{x \in X} \left[\left(\bigwedge_{a \in A} l_{x,a} \wedge \neg l_{a,x} \right) \to \left(w_{A \cup \{x\}, A} \wedge \neg w_{A, A \cup \{x\}} \right) \right] \\
& \equiv \bigwedge_{A \in \mathcal{X}} \bigwedge_{x \in X} \left[\neg \left(\bigwedge_{a \in A} l_{x,a} \wedge \neg l_{a,x} \right) \vee \left(w_{A \cup \{x\}, A} \wedge \neg w_{A, A \cup \{x\}} \right) \right] \\
& \equiv \bigwedge_{A \in \mathcal{X}} \bigwedge_{x \in X} \left[\left(\bigvee_{a \in A} \neg l_{x,a} \vee l_{a,x} \right) \vee \left(w_{A \cup \{x\}, A} \wedge \neg w_{A, A \cup \{x\}} \right) \right] \\
& \equiv \bigwedge_{A \in \mathcal{X}} \bigwedge_{x \in X} \left[\left(\left(\bigvee_{a \in A} \neg l_{x,a} \vee l_{a,x} \right) \vee w_{A \cup \{x\}, A} \right) \wedge \right. \\
& \qquad\qquad \left. \left(\left(\bigvee_{a \in A} \neg l_{x,a} \vee l_{a,x} \right) \vee \neg w_{A, A \cup \{x\}} \right) \right],
\end{aligned}
$$

where the last three steps only serve the purpose of converting the formula to conjunctive normal form.

The remaining problematic parts of the formula are the propositional variable index $A \cup \{x\}$, and the disjunction domain criterion $a \in A$. In order to write out the formula explicitly (which we need to do to be able to feed it to a SAT solver) we have to determine which set is represented by $A \cup \{x\}$ and also decide whether $a \in A$ for any $a \in X$. In different words, what we need is explicit access to the elements of a set and also we have to be able to manipulate them. This would theoretically be possible by hand; practically, however, the instantiation of the formula is way too large to be written out manually.[33] Therefore, we need a computer program for the final conversion step, which we are going to describe in the following section.

The conversion of (GF2) works analogously and yields:

[33] Just for (GF1), this procedure yields a propositional formula with a total number of

$$\sum_{k=1}^{n} \left[\binom{n}{k} \cdot n \cdot (2 \cdot (2k+1)) \right] = 2n \cdot \sum_{k=1}^{n} \left[2k \binom{n}{k} + \binom{n}{k} \right]$$

literals. This corresponds to 5364 literals for the base case of the Kannai-Peleg Theorem ($n = 6$).

$$
\begin{aligned}
(\texttt{GF2}) \quad & (\forall A \in \mathcal{X})(\forall x \in X)\,[((\forall a \in A) x \dot{<} a) \to A \cup \{x\} \prec A] \\
\cong & \bigwedge_{A \in \mathcal{X}} \bigwedge_{x \in X} \left[\left(\left(\bigvee_{a \in A} \neg l_{a,x} \vee l_{x,a} \right) \vee w_{A, A \cup \{x\}} \right) \wedge \right. \\
& \qquad \left. \left(\left(\bigvee_{a \in A} \neg l_{a,x} \vee l_{x,a} \right) \vee \neg w_{A \cup \{x\}, A} \right) \right],
\end{aligned}
$$

again leaving us with the two critical terms $A \cup \{x\}$ and $a \in A$.

The axiom of independence (`IND`) can also be transformed in the above fashion by first using finiteness to replace the quantifiers and the orders, and then normalizing the formula to conjunctive normal form:

$$
\begin{aligned}
(\texttt{IND}) \quad & (\forall A, B \in \mathcal{X})(\forall x \in X \setminus (A \cup B))\,[A \succ B \to A \cup \{x\} \succeq B \cup \{x\}] \\
& \equiv (\forall A, B \in \mathcal{X})(\forall x \in X \setminus (A \cup B))[(A \succeq B \wedge B \not\succeq A) \to \\
& \qquad\qquad A \cup \{x\} \succeq B \cup \{x\}] \\
& \cong \bigwedge_{A,B \in \mathcal{X}} \bigwedge_{\substack{x \in X \\ x \notin (A \cup B)}} \left[(w_{A,B} \wedge \neg w_{B,A}) \to w_{A \cup \{x\}, B \cup \{x\}} \right] \\
& \equiv \bigwedge_{A,B \in \mathcal{X}} \bigwedge_{\substack{x \in X \\ x \notin (A \cup B)}} \left[\neg (w_{A,B} \wedge \neg w_{B,A}) \vee w_{A \cup \{x\}, B \cup \{x\}} \right] \\
& \equiv \bigwedge_{A,B \in \mathcal{X}} \bigwedge_{\substack{x \in X \\ x \notin (A \cup B)}} \left[(\neg w_{A,B} \vee w_{B,A}) \vee w_{A \cup \{x\}, B \cup \{x\}} \right] \\
& \equiv \bigwedge_{A,B \in \mathcal{X}} \bigwedge_{\substack{x \in X \\ x \notin (A \cup B)}} \left[\neg w_{A,B} \vee w_{B,A} \vee w_{A \cup \{x\}, B \cup \{x\}} \right].
\end{aligned}
$$

The problematic terms here are $x \notin (A \cup B)$, $A \cup \{x\}$ and $B \cup \{x\}$. But, as we will see, they can — just like the critical terms mentioned before — be handled by our implementation of a program that writes out the axioms explicitly.

3.3.3 Instantiation of the Axioms on a Computer

In order to write out the formulas of the previous section explicitly, we make use of a computer program, which we implemented in the programming language *Java* [67]. This section discusses the implementation and, in particular, presents the methods employed to cater for previously problematic expressions, like $A \cup \{x\}$ and $a \in A$.

First note what the prescribed format for the output of our generating program has to be. Like other SAT solvers, *zChaff* [61] works on input files written according to the DIMACS CNF format [18]. In a few words, this format requires the propositional variables to be represented by natural numbers (starting from 1, since 0 is used as a separator) with a minus (-) in front for negated literals. Furthermore, the whole file needs to be in conjunctive normal form: it has to contain exactly one clause per line.

In the following, we will describe how we achieved a formulation of the axioms in this target format. The main idea is to fix an enumeration of all propositional variables (of type $l_{x,y}$ and $w_{A,B}$ with $x, y \in X$ and $A, B \in \mathcal{X}$) by first enumerating sets and elements and then combining numbers of pairs of these. The numberings will be designed in such a way that functions and relations like *union* and *element of* can be applied to numbers directly, yielding numbers of the corresponding images or a truth value, respectively. Quantifiers will then correspond to iterations over their respective domains and easily readable source code will be capable of instantiating the axioms. An instructive example of source code can be found in Appendix B.1, but will be referred to later again, when more has been said about the particular enumerations we used.

Since the numberings of the items under consideration (here: elements, sets and later propositional variables) form the core of our implementation, we start the translation process by first fixing an (arbitrary) numbering of the n elements $x \in X$, i.e., a bijective function $c_n : X \to \{0, 1, \dots, n-1\}$.[34] For the Kannai-Peleg Theorem with its six elements the codes will hence range from 0 to 5.

We can then specify the corresponding numbering of sets in $\mathcal{X}$. This requires special attention because we want to define it in such a way that treating the problematic terms (as mentioned above) is as easy as possible. An apparent way to do this is by looking at a set as its characteristic function and converting the corresponding finite string of zeros and ones to a natural number.

Definition 3.12. *Let X contain n elements, $n \in \mathbb{N}$.*

- *Then the* size-n characteristic string *of a set $A \in \mathcal{X}$ is the vector $\chi_n(A) := (\chi_A(0), \chi_A(1), \dots, \chi_A(n-1))$, where*

$$\chi_A(i) = \begin{cases} 1, & \text{if } c_n^{-1}(i) \in A, \\ 0, & \text{otherwise,} \end{cases}$$

  for any $i \leq n-1$.
- *The* size-n code *of A is the natural number $c'_n(A)$ represented by the size-n characteristic string of A decremented by one,*[35] *i.e., it is given by the bijective function $c'_n : \mathcal{X} \to \{0, 1, \dots, 2^n - 2\}$,*

$$c'_n(A) := \operatorname{int}(\chi_n(A)) - 1,$$

[34] In contrast to the propositional variables, there are no numbering constraints for the elements and so their numbers are allowed to start from 0.

[35] Since $\mathcal{X}$ does not contain the empty set, which would have 0 as its code, all the codes are shifted downwards by 1.

with $\mathrm{int}(\chi_n(A))$ *being the* integer value *of* $\chi_n(A)$, *i.e.,*

$$\mathrm{int}(\chi_n(A)) = \sum_{i=0}^{n-1} \chi_A(i) \cdot 2^i.$$

If the size of X as well as the type of argument (element or set) is clear from the context, we will, for any $x \in X$ and $A \in \mathcal{X}$, refer to the size-n codes $c_n(x)$ and $c'_n(A)$ simply by $\#(x)$ and $\#(A)$, respectively.

But let us quickly look at an example to make things clearer:

Example 3.13. For the case of the Kannai-Peleg Theorem, we have 6 elements in X, say $X = \{a, b, c, d, e, f\}$. As described, we number them from 0 to $n - 1 = 5$, say

$$\begin{bmatrix} a & b & c & d & e & f \\ 0 & 1 & 2 & 3 & 4 & 5 \end{bmatrix}.$$

The size-6 characteristic string of, for example, the set $A := \{a, c, f\} \in \mathcal{X}$ then is the vector $\chi_6(A) = (\chi_A(0), \chi_A(1), \ldots, \chi_A(5)) = (1, 0, 1, 0, 0, 1)$ and, thus, the size-6 code of A is given by

$$\begin{aligned} c'_6(A) &:= \mathrm{int}(\chi_6(A)) - 1 \\ &= \sum_{i=0}^{n-1} \chi_A(i) \cdot 2^i - 1 \\ &= 1 \cdot 2^0 + 0 \cdot 2^1 + 1 \cdot 2^2 + 0 \cdot 2^3 + 0 \cdot 2^4 + 1 \cdot 2^5 - 1 \\ &= 1 + 4 + 32 - 1 = 36. \end{aligned}$$

In total, this procedure gives us a numbering from 0 to 62 of the $2^6 - 1 = 63$ non-empty sets in $\mathcal{X}$.

The benefit of our particular numbering is that our program can conveniently perform operations (e.g., union and complement) on the size-n codes directly. Internally on a digital computer, the codes have to be stored in binary anyway and we can take advantage of this fact and manipulate these codes. Obtaining an explicit representation of an expression like $A \cup \{x\}$ is possible by just creating the code for $\{x\}$ from the one for x, and then constructing the union together with the code for A. In the following, we give some algorithmic descriptions of functions, like the aforementioned *union* function, that we used in our implementation.

Definition 3.14. *Let X contain n elements, $n \in \mathbb{N}$.*

- *The* size-n union *of two size-n codes $j = \#(A)$ and $k = \#(B)$ of the sets $A, B \in \mathcal{X}$ is given by*

  $$\cup(j, k) = \mathrm{int}\left(\max\left(\mathrm{int}^{-1}(j + 1), \mathrm{int}^{-1}(k + 1)\right)\right) - 1,$$

  where max *is the usual* bitwise maximum.

- *The* size-n complement *of a size-n code $j = \#(A)$ of a set $A \in \mathcal{X}$ is given by*
$$\mathrm{inv}(j) = \mathrm{int}\left(\mathbb{1} - \mathrm{int}^{-1}(j+1)\right) - 1,$$
where $\mathbb{1} = (1, 1, \ldots, 1) \in \mathbb{N}^n$.
- *The* size-n singleton set *of a size-n code $j = \#(x)$ of an element $x \in X$ is given by*
$$\mathrm{single}(j) = \mathrm{int}\left(e_j\right) - 1,$$
where $e_j = (0, \ldots, 0, 1, 0, \ldots, 0) \in \mathbb{N}^n$ is the j-th standard base vector of $\mathbb{N}^n$.
- *The* size-n truth value *of whether a set $A \in \mathcal{X}$ contains an element $x \in X$ is given by*
$$\in(j,k) = (\mathrm{int}^{-1}(j+1))_k$$
for their respective size-n codes $j = \#(A)$ and $k = \#(x)$.

It should be clear that these functions indeed serve their intuitive purpose, i.e., that for example $\cup(\#(A), \#(B))$ indeed gives the code of the usual set union $A \cup B$. We do therefore not provide all proofs here, but just support this intuition with an example:

Example 3.15. Consider again a set $X = \{a, b, c, d, e, f\}$ with the numbering

$$\begin{bmatrix} a & b & c & d & e & f \\ 0 & 1 & 2 & 3 & 4 & 5 \end{bmatrix}$$

as well as the subset $A := \{a, c, f\} \in \mathcal{X}$ with size-6 characteristic string $\chi_6(A) = (1, 0, 1, 0, 0, 1)$ and, thus, the size-6 code $\#(A) = c_6'(A) = 36$. Now take also into account a second subset $B := \{c, e\} \in \mathcal{X}$ with size-6 characteristic string $\chi_6(B) = (0, 0, 1, 0, 1, 0)$ and, thus, the size-6 code $\#(B) = c_6'(B) = 19$. We can then compute

$$\begin{aligned} \cup(36, 19) &= \mathrm{int}\left(\max\left(\mathrm{int}^{-1}(36+1), \mathrm{int}^{-1}(19+1)\right)\right) - 1 \\ &= \mathrm{int}\left(\max\left((1,0,1,0,0,1), (0,0,1,0,1,0)\right)\right) - 1 \\ &= \mathrm{int}\left((1,0,1,0,1,1)\right) - 1 \\ &= 53 - 1 = 52, \end{aligned}$$

which represents the set $\{a, c, e, f\} = A \cup B$.

Similar functions and operations can be constructed analogously or by combining already defined ones.[36]

[36] For instance, *size-n intersection* can be defined in terms of *size-n union* and *size-n complement* by De Morgan's law, i.e.,

$$\cap(j,k) = \mathrm{inv}\left(\cup\left(\mathrm{inv}(j), \mathrm{inv}(k)\right)\right)$$

for any pair (j, k) of size-n codes of sets in $\mathcal{X}$.

With the codes of subsets and elements as well as some operations on these codes defined, we can turn to the encoding of propositional variables. Let us denote the set of all propositional variables for a given size n of X by $\mathcal{V}_n$ (recall that we have the types $l_{x,y}$ and $w_{A,B}$ with $x, y \in X$ and $A, B \in \mathcal{X}$). As every such propositional variable is indexed by either two elements from X or two sets from $\mathcal{X}$, we can use the following bijective function to encode all possible propositional variables as natural numbers in the range $\{1, 2, \ldots, n^2 + m^2\}$.

Definition 3.16. *Let X contain n elements, $n \in \mathbb{N}$, and denote the cardinality of $\mathcal{X} = 2^X \setminus \{\emptyset\}$ by $m(= 2^n - 1)$. Then the* size-n encoding *$\mathrm{enc}_n(p)$ of a propositional variable $p \in \mathcal{V}_n$ is given by the bijective function $\mathrm{enc}_n : \mathcal{V}_n \to \{1, 2, \ldots, n^2 + m^2\}$,*

$$\mathrm{enc}_n(p) := \begin{cases} \#(x) \cdot n + \#(y) + 1, & \textit{if } p \textit{ is of type } l_{x,y} \textit{ with } x, y \in X, \\ \#(A) \cdot m + \#(B) + n^2 + 1, & \textit{if } p \textit{ is of type } w_{A,B} \textit{ with } A, B \in \mathcal{X}. \end{cases}$$

By this definition, the function enc_n just enumerates all pairs of elements first and then all pairs of sets second, each set in a lexicographic fashion. But any other bijective[37] function (as long as it is computable with reasonable effort) would have sufficed, too.

To support this explanation of the encoding function enc_n we give another example:

Example 3.17. Like in the previous example 3.15 consider the set $X = \{a, b, c, d, e, f\}$ with the numbering

$$\begin{bmatrix} a & b & c & d & e & f \\ 0 & 1 & 2 & 3 & 4 & 5 \end{bmatrix}$$

as well as the subsets $A := \{a, c, f\}$ with $\#(A) = 36$ and $B := \{c, e\}$ with $\#(B) = 19$. If we want to encode the propositional variable $w_{A,B}$ (meaning that A is at least as good as B), then we calculate

$$\mathrm{enc}_6(w_{A,B}) = \#(A) \cdot m + \#(B) + n^2 + 1 = 36 \cdot (2^6 - 1) + 19 + 6^2 + 1 = 2324.$$

For propositional symbols of type $l_{x,y}$ we, for instance, have

$$\mathrm{enc}_6(l_{f,f}) = \#(f) \cdot n + \#(f) + 1 = 5 \cdot 6 + 5 + 1 = 36, \text{ and}$$
$$\mathrm{enc}_6(l_{a,b}) = \#(a) \cdot n + \#(b) + 1 = 0 \cdot 6 + 1 + 1 = 2.$$

With all these tools it is now entirely unproblematic to instantiate the formulas from Section 3.3.2 automatically. All that needs to be done is translating them to our specific source code style. Quantifiers then correspond to *for* loops over all elements or sets, respectively, and restrictions of the quantification domain as well as operations on elements and sets can be taken care of by functions like to the ones described in Definition 3.14. For an illustrating example of source code the reader is referred to Appendix B.1.

[37] Surjectivity is not even strictly needed, but makes the whole representation more compact and avoids loading extra work on the SAT solver later, which would assign arbitrary values to the left out variables.

3.3.4 Verification of Instances using a SAT Solver

In the previous subsections 3.3.2 and 3.3.3 we described how to generate a (long) formula in propositional logic representing the base case of the Kannai-Peleg Theorem (see Lemma 3.11). Let us denote this formula by φ_{KP}, with the index KP standing for 'Kannai-Peleg'. The formula φ_{KP} describes a model with a linear order on its universe of six elements and a weak order satisfying the axioms (`GF`) and (`IND`) on the set of the non-empty subsets of its universe.[38] If such a model exists, φ_{KP} has a satisfying assignment[39] (the explicit description of both orders) and, thus, a SAT solver will discover this (assuming that there are no time and memory bounds).[40] If, conversely, such a model does not exist — which is exactly the statement of the Kannai-Peleg Theorem — then φ_{KP} is unsatisfiable and, again, a SAT solver will theoretically be able to detect this.

The base case of the Kannai-Peleg Theorem is therefore equivalent to the following lemma:

Lemma 3.18 (base case restated). *The formula* φ_{KP} *is unsatisfiable.*

Indeed, feeding φ_{KP} (as generated by our program) to the SAT solver *zChaff* returns the correct result ('UNSAT') in about 5 seconds (see Appendix B.3) and, hence, the automatic verification of this theorem is complete.

3.4 Additional Features of our Implementation

The output of a SAT solver like *zChaff* usually contains more information than just the mere satisfiability status of a formula. In particular, any satisfiable instance will lead to the production of a satisfying assignment, which can be used to gain a better understanding of the problem encoded in the formula. But also for unsatisfiable instances additional output is produced. We will see that this output can be considered a proof object allowing for automatic verification of the unsatisfiability.

In the next two subsections we briefly explain how these two kinds of additional output can be analyzed to provide helpful insights concerning the Kannai-Peleg Theorem in particular and also arbitrary encoded problems of this sort in general.

[38] Strictly speaking, of course, the model can be much larger, but it has to contain the model we described as a substructure.

[39] A satisfying assignment is a mapping $v : \mathcal{V}_n \to \{0, 1\}$ such that the truth value of the whole formula is 1.

[40] Some SAT solvers even return a satisfying assignment that can then be "decoded" into a human-readable description of the model. How exactly this can be done will be laid out in Section 3.4.1.

3.4.1 Models for Possibilities

As indicated already, when the SAT solver *zChaff* judges a formula to be satisfiable it automatically outputs the satisfying assignment it found. Since in our setting the individual propositional variables stand for propositions of the type $A \succeq B$ with $A, B \in \mathcal{X}$ and $x \mathrel{\dot{\geq}} y$ with $x, y \in X$, respectively, we can convert this satisfying assignment, which is nothing but an involved representation of the linear order $\dot{\geq}$ and the corresponding weak order $\succeq$, to a form that is more common in mathematics and human-readable.

Since we are exclusively dealing with linear and weak orders, a natural description of the orders can be written down in a linear fashion in the style $x_1 \square x_2 \square x_3 \square \ldots \square x_n$. Here, each occurrence of the symbol $\square$ stands for a relation symbol; for the linear order this is only the strict relation $\dot{>}$, whereas for the weak order both the strict relation $\succ$ as well as the indifference relation $\sim$ are possible.

We implemented a procedure to automatically convert a given satisfying assignment (in the form as printed by the SAT solver *zChaff*)[41] to this much more intuitive notation. Thereby, we added an important feature to our program: examples of orders satisfying a certain set of axioms can now easily be generated in an intuitive format. The actual conversion procedure works by first sorting the elements and sets according to their order using a standard sorting method[42] that only requires pairwise comparisons, and then listing them in this new order (again using pairwise comparisons to find out whether neighboring items are strictly ordered or considered equal). The pairwise comparisons can be carried out by inspecting the truth values of the corresponding two propositions in the satisfying assignment. So, for instance, to find out how x_i and x_j are ranked we have to look at the truth values of l_{x_i,x_j} and l_{x_j,x_i} (telling us whether $x_i \mathrel{\dot{>}} x_j$ or $x_j \mathrel{\dot{>}} x_i$).[43] Finding the corresponding number of the propositional symbols can be taken care of by the same methods used for encoding the input formula (see Section 3.3.3).

In order to achieve an even better readability of the model, we also added a procedure generating a second representation, with the elements renamed according to their position in the order $\dot{\geq}$.

For the specific case of the Kannai-Peleg Theorem, the above method can be used to automatically generate a "counterexample" to the Kannai-Peleg Theorem for 5 elements, i.e., a weak order satisfying the axioms (`GF`) and

[41] Alternatively, also the SAT solver *PrecoSAT* [13] can be used.

[42] We used the *Java* implementation of the standard sorting method *mergesort*.

[43] As we know that $\dot{\geq}$ is a linear order, a single check of just one truth value would suffice. Since, for a weak order, however, this is no longer the case and also to avoid bugs in our implementation, we chose this slightly more costly, but general option.

(IND), thereby showing that 6 is indeed the smallest size of X leading to the impossibility.[44] The model returned by the SAT solver is the following:[45]

- Elements:
  $x_1 \dot{>} x_2 \dot{>} x_3 \dot{>} x_4 \dot{>} x_5$,

- Sets:
  $\{x_1\} \succ \{x_1, x_2\} \succ \{x_1, x_3\} \sim \{x_1, x_2, x_3\} \succ \{x_1, x_4\} \sim \{x_1, x_2, x_4\} \sim \{x_1, x_3, x_4\} \sim \{x_1, x_2, x_3, x_4\} \succ \{x_2\} \succ \{x_2, x_3\} \succ \{x_2, x_4\} \sim \{x_3\} \sim \{x_2, x_3, x_4\} \succ \{x_3, x_4\} \succ \{x_4\} \succ \{x_1, x_5\} \sim \{x_1, x_2, x_5\} \sim \{x_1, x_4, x_5\} \sim \{x_1, x_2, x_4, x_5\} \sim \{x_1, x_3, x_5\} \sim \{x_1, x_2, x_3, x_5\} \sim \{x_1, x_3, x_4, x_5\} \sim \{x_1, x_2, x_3, x_4, x_5\} \succ \{x_2, x_5\} \sim \{x_2, x_4, x_5\} \sim \{x_2, x_3, x_5\} \sim \{x_2, x_3, x_4, x_5\} \succ \{x_3, x_5\} \sim \{x_3, x_4, x_5\} \succ \{x_4, x_5\} \succ \{x_5\}$.

In this representation it can be checked quickly that indeed all the sets are ordered only accordingly to their maximal and minimal elements as it was predicted by Lemma 1.4.

We will make further use of this convenient procedure in Section 5.6 (where we analyze the impossibility theorems found by our theorem search) for the construction of models that can show certain axioms to be compatible and also logically independent, i.e., that none of the axioms follows from the others.

3.4.2 Verification of the Unsatisfiability of Formulas

The SAT solver *zChaff* is able to generate a resolution-style proof object called *proof trace*.[46] This proof trace can then be checked by a third-party verification system. It is evident that this is a very helpful feature in order to detect potential bugs in the SAT solver and to increase the degree of confidence in the unsatisfiability.[47] In our implementation, we integrated the verification system *zverify_df* as included in the *zChaff* distribution and successfully verified all unsatisfiable instances found.[48]

Our program generating the formula to be checked by the SAT solver could, of course, contain errors, too. But even though it would possibly also

[44] Recall the inductive step, Lemma 3.9, which — given that for 5 elements we can find a model — entails the existence of models for all sizes smaller than 5.

[45] The original and complete output of our program can be found in Appendix B.2.

[46] A good documentation and examples of this format are presented in Section 2.2f of a paper by Weber and Amjad about how to integrate a SAT solver with higher-order logic (HOL) theorem provers [77].

[47] For any *satisfiable* instance, *zChaff* runs a verification procedure automatically since this can be done in linear time (verify for every clause whether at least one of its literals is set to true) and the corresponding source code is short and easy to check manually.

[48] For technical reasons, all clauses containing both polarities of a literal (i.e., p and $\neg p$) have to be removed prior to verification. Note that these clauses do not affect the satisfiability of a formula as they are automatically true for any assignment.

be interesting to look into ways of formally verifying our source code, this does not seem to give any major new insights: the code (see Appendix B.1 for excerpts), which is easily seen to be very close to the formula schemes as derived in Subsection 3.3.2, can without any difficulties be compared to these by hand.

3.5 Summary

In this chapter we have seen how a SAT solver can be used to automatically verify small instances of a problem, which lead to a proof of the full result together with a suitable lemma.

We have recapitulated how Tang and Lin applied this method, which they had developed, to Arrow's Theorem by first providing an inductive step and then formalizing the corresponding base case in propositional logic. Subsequently, we extended the method to be able to treat problems in the framework of *ranking sets of objects* and, in particular, to automatically verify the Kannai-Peleg Theorem. For this we — just like Tang and Lin — proved an inductive step and verified the base case on a computer using a SAT solver on an automatically created propositional instance. For the formalization in propositional logic, the main challenge laid in handling second order quantification over sets in such a way that functions like *union* and relations like *element of* could be applied to the sets. Our solution to this problem, the implementation of the program capable of automatic instantiation of axioms in the field of *ranking sets of objects*, was described in detail.

We closed with explanations and observations of additional features of our software, viz. the automated construction of orders satisfying a set of axioms, and the automated verification of unsatisfiabilities as found by the SAT solver.

In the following two chapters we will show how to generalize the methodology introduced in this chapter to a wide range of axioms and therefore impossibility theorems.

4

Reduction of Impossibilities to Small Instances

4.1 Introduction

The technique presented in the previous chapter is so appealing as it can very quickly be transferred to similar impossibility theorems. More importantly, already LIN and TANG [39] remarked that reversing the order of their inductive proofs, i.e., by looking at the base case first (without having an inductive step ready), can help guide the search for new results. We will take this idea one step further and provide a universal step that can directly serve as the inductive step for a large class of axioms. This way, it will be sufficient to only check base cases in order to find new impossibility theorems and this can be done very efficiently on a computer, as we have experienced for the Kannai-Peleg Theorem.[49] We will prove a Preservation Theorem, which says that certain axioms are preserved in specific substructures. This is exactly what we need since we want axioms to be preserved when moving from a larger linearly ordered set to a smaller one (cf. Lemma 3.9). Reading this result contrapositively, does then give our universal step, which reduces impossibility theorems to small instances.

For the proof of the universal lemma we have to proceed in the framework of model theory in order to have access to the syntactic as well as semantic features of axioms. We will therefore in Section 4.2 first describe a many-sorted language for our specific problem of *ranking sets of objects*, before we give a weak version of the universal lemma in Section 4.3, which is a consequence of the well-known Łoś-Tarski Theorem for first-order logic and thus does not require any problem-specific insights. In Section 4.4 we make use of our particular problem structure and prove our main result Theorem 4.13 directly, which has the stronger form of our universal step as a corollary. This (strong) universal step is then powerful enough to cater for all the axioms from the literature that we were able to formalize in our language and will

[49] Further evidence will be given in Chapter 5.

then finally enable us to carry out a fully automated theorem search, which we are going to describe in the following Chapter 5.

4.2 The Language MSLSP for Preferences over Sets

A natural and well-understood language for our problem domain is many-sorted (first-order) logic, which has, compared to first-order logic, more and different quantifiers (allowing for quantification over different domains containing the elements of a respective *sort*), but is still reducible to first-order logic. Apart from the quantifiers, many-sorted logic is practically equivalent to first-order logic[50] and thus many results (e.g., soundness, completeness, compactness, Löwenheim-Skolem properties, etc.) can be transferred from first-order logic or can be directly proven. Later on, we will also transfer one direction of a result (the Łoś-Tarski Theorem) from standard (i.e., first-order) model theory to many-sorted logic in order to apply it to our problem setting.

As indicated, many-sorted logic is characterized by the use of a set $\boldsymbol{S}$ of different sorts $s \in \boldsymbol{S}$. These sorts are also quite often used implicitly in the mathematical practice: examples are points, lines, squares, etc. in geometry, vectors and scalars in vector spaces, or natural and real numbers in real analysis, to list just a few.

A structure[51] $\mathfrak{A}$ for many-sorted logic is thus just like one for first-order logic, but with separate domains $\mathrm{dom}_s(\mathfrak{A})$ for each sort $s \in \boldsymbol{S}$ instead of one single domain. We then have corresponding quantifiers $\forall_s$ and $\exists_s$ for each sort s, equipped with the intuitive semantics:

- $\mathfrak{A} \models \forall_s x \varphi(x)$ if and only if $\mathfrak{A} \models \varphi(a)$ for all $a \in \mathrm{dom}_s(\mathfrak{A})$,
- $\mathfrak{A} \models \exists_s x \varphi(x)$ if and only if $\mathfrak{A} \models \varphi(a)$ for some $a \in \mathrm{dom}_s(\mathfrak{A})$.

Other than that, many-sorted logic is analogous to first-order logic with the slight difference of having separate variable, function, and relation symbols for the different sorts or combinations of sorts.

For a completely formal definition of many-sorted logic see Chapter VI of Manzano [40] and §4.3 of Enderton [22].

In our case, we will have two sorts ($\boldsymbol{S} = \{\varepsilon, \sigma\}$): elements ($\varepsilon$) and sets ($\sigma$). We further demand that there is a relation $\in$ of type $\langle \varepsilon, \sigma \rangle$ as well as two relations $\dot{\geq}$ and $\succeq$ of type $\langle \varepsilon, \varepsilon \rangle$ and $\langle \sigma, \sigma \rangle$, respectively. These will then later be interpreted as the usual membership relation and our linear and weak orders, respectively. But, of course, one can have many more relations and functions in the signature, and we will use the following time and again (even though at this stage they are just symbols, we also indicate their natural interpretations in parentheses already):

[50] In strict terminology, many-sorted logic is not an extension to first-order logic. See Manzano [40] for an in-depth discussion of the relation between first-order and many-sorted logic.

[51] The names 'structure' and 'model' are typically used synonymously.

- Relations:
  - $\subseteq$, type $\langle\sigma,\sigma\rangle$ (intuitively: *set inclusion*)
  - disjoint, type $\langle\sigma,\sigma\rangle$ (intuitively: true iff sets are disjoint)
  - evencard, type $\langle\sigma\rangle$ (intuitively: true iff the cardinality of a set is even)
  - equalcard, type $\langle\sigma,\sigma\rangle$ (intuitively: true if sets have the same cardinality)
- Functions:
  - $\cup$, type $\langle\sigma,\sigma,\sigma\rangle$ (intuitively: *set union*)
  - $\{\cdot\}$, type $\langle\varepsilon,\sigma\rangle$ (intuitively: transforms an element into the singleton set containing it)
  - replaceInBy, type $\langle\varepsilon,\sigma,\varepsilon\rangle$ (intuitively: replace an element in a set by another element; e.g., $(A \setminus \{a\}) \cup \{b\}$)

We denote this language of many-sorted logic (with the two sorts ε and σ and the signature containing exactly the above relations and functions) by MSLSP (**M**any-**S**orted **L**ogic for **S**et **P**references).[52]

There are more possible (and common) set-functions like the following, which could be included, but have to be handled with care:

- Functions:
  - $\cap$, type $\langle\sigma,\sigma,\sigma\rangle$ (intuitively: *set intersection*)
  - $\setminus$, type $\langle\sigma,\sigma\rangle$ (intuitively: *set difference*)
  - $\cdot^{\mathrm{c}}$, type $\langle\sigma,\sigma\rangle$ (intuitively: *set complement*)
  - $\emptyset$, type $\langle\sigma\rangle$ (intuitively: *empty set*)

These will — in their standard interpretations — not be functions on the space $\mathcal{X} = 2^X \setminus \{\emptyset\}$ as they can produce the empty set $\emptyset$ and must therefore only be allowed when considering the empty set, too (see also Remark 4.8).

Notation-wise we will sometimes use (the more common) infix notation for certain relations and functions. We will for instance write $A \cup B$, $a \in A$, $A \subseteq B$ and $\{x\}$ instead of $\cup(A,B)$, $\in(a,A)$, $\subseteq(A,B)$ and $\{\cdot\}(x)$, respectively. Furthermore, we will sometimes use negated symbols like $x \notin A$ for $\neg(x \in A)$ as well as the strict relation symbols $A \succ B$ and $x \dot{>} y$ when we mean $A \succeq B \wedge \neg(B \succeq A)$ and $x \dot{\geq} y \wedge \neg(y \dot{\geq} x)$, respectively. Generally, we will use the (standard model-theoretic) notation of Hodges [34].

Many axioms in the literature can be directly stated in MSLSP and as an example we give the representations of the principle of *independence* and the *Gärdenfors principle* (see Section 1.4.1):

Example 4.1. (IND) can be formulated in MSLSP:

$$\forall_\sigma A \forall_\sigma B \forall_\varepsilon x \left[(x \notin (A \cup B) \wedge A \succ B) \rightarrow A \cup \{x\} \succeq B \cup \{x\}\right],$$

where, strictly speaking, $x \notin (A \cup B)$ stands for $\neg[x \in (A \cup B)]$ and similarly $A \succ B$ stands for $A \succeq B \wedge \neg(B \succeq A)$.

[52] In the spirit of Hodges [34] we assume to have equality $=$ as a logical symbol and not as a relation in the signature. This means that it will not have to be interpreted by the structure as a relation, but stands for actual equality.

Example 4.2. (GF) can be formulated in MSLSP:

$$\forall_\sigma A \forall_\varepsilon x \, [(\forall_\varepsilon a (a \in A \to x \dot{>} a)) \to A \cup \{x\} \succ A]$$
$$\forall_\sigma A \forall_\varepsilon x \, [(\forall_\varepsilon a (a \in A \to a \dot{>} x)) \to A \succ A \cup \{x\}]$$

Unfortunately, for some axioms there is no easy representation in MSLSP. The concept of *neutrality*, which says that the labeling of alternatives is irrelevant for the ranking $\succeq$, is an example of such an axiom:

(NEU) $\Big[\big[\big(x \dot{\geq} y \iff \varphi(x) \dot{\geq} \varphi(y)$ and $y \dot{\geq} x \iff \varphi(y) \dot{\geq} \varphi(x)\big)$
for all $x \in A$, $y \in B\big] \Rightarrow$
$[A \succeq B \iff \varphi(A) \succeq \varphi(B)$ and $B \succeq A \iff \varphi(B) \succeq \varphi(A)]$
for any two sets $A, B \in \mathcal{X}$ and any injective mapping $\varphi : A \cup B \to X$.

But there are also axioms of such form (involving a quantification over certain functions) that *are* expressible in MSLSP. Consider, for instance, the axiom of *weak preference dominance* (see Puppe [54]), which Sen [65] originally proposed in 1991 (in a slightly stronger form):

(WPD) $\big[\big(|A| = |B|$ and there exists a bijective function $\varphi : A \to B$ such that $a \dot{\geq} \varphi(a)$ for all $a \in A\big) \Rightarrow A \succeq B\big]$ for any two sets $A, B \in \mathcal{X}$.

Like for (NEU) there is again no obvious way to express this axiom in MSLSP. Puppe [54], however, proved in 1995 that (WPD) is actually equivalent to an axiom much closer to our formalism, called *preference-basedness*:[53]

(PB) $\big[b \dot{\geq} a \Rightarrow (A \setminus \{a\}) \cup \{b\} \succeq A\big]$ for all $A \in \mathcal{X}$, $a \in A$, $b \notin A$.

(PB) is easily seen to be expressible in MSLSP[54] and will even have the required format for our universal step, which we are going to introduce in the next section.

4.3 Weak Universal Step by Standard Model Theory

As a first step towards our universal lemma, we can extend a basic result about $\forall_1$-formulas[55] from classical (first-order) model theory to many-sorted logic. It is actually one direction of the *Łoś-Tarski Theorem* and says that:

Lemma 4.3. *$\forall_1$-formulas (and logically equivalent formulas) are preserved in (first-order) substructures.*

[53] Endriss [23] refers to this axiom as the *single-flip axiom*.
[54] $\forall_\sigma A \forall_\varepsilon a \forall_\varepsilon b \, \big[\big(a \in A \wedge b \notin A \wedge b \dot{\geq} a\big) \to \text{replaceInBy}(a, A, b) \succeq A\big]$
[55] A formula is a $\forall_1$-formula if it is in the smallest class of formulas that contains the quantifier-free formulas and is closed under $\bigwedge$, $\bigvee$ and adding universal quantifiers in the front.

In detail, Lemma 4.3 expresses that for all $\forall_1$-formulas $\varphi(\bar{x})$, whenever $\mathfrak{A}$ is a substructure of $\mathfrak{B}$ and $\bar{a}$ is a tuple of elements from $\mathfrak{A}$ such that $\mathfrak{B} \models \varphi[\bar{a}]$, then $\mathfrak{A} \models \varphi[\bar{a}]$, too. In particular, this applies to $\forall_1$-sentences[56] and therefore to all axioms of this form. Lemma 4.3 therefore is a first-order version of where we want to get: a result that guarantees axioms to be preserved in smaller structures (cf. Lemma 3.9). Note, that even though it seems to be a strong restriction that only $\forall_1$-sentences are preserved, this is already sufficient for all axioms of linear and weak orders as well as, for instance, the principle of independence.

The exact statement of Lemma 4.3 can, for example, be found in Hodges [34] as Corollary 2.4.2. and the proof idea is relatively simple: by contradiction it suffices to show that $\exists_1$-formulas are preserved by embeddings (Theorem 2.4.1. in Hodges [34]). The proof thereof proceeds by induction on the complexity of the formula and the critical case of the existential quantifier does not cause any trouble as witnesses are not "lost" when moving to a larger structure.

We are now going to extend this result to many-sorted logic to be able to apply it in our setting. What we need is a conversion from many-sorted logic to one-sorted logic and a correspondence theorem. Both can, for instance, be obtained from Enderton [22] or, in a more detailed form, from Manzano [40]. We quickly recapitulate the main ideas:

In order to move from many-sorted logic to the one-sorted case, both formulas as well as structures need to be converted. The aim is to unify the domain and relativize the quantifiers. Therefore, the syntactic alterations of the formulas consist of replacing all sorted universal quantifiers $\forall_s x \varphi(x)$ by a relativized general quantifier $\forall x(P_s(x) \rightarrow \varphi(x))$, where P_s is a fresh predicate symbol. Similarly, we replace sorted existential quantifiers $\exists_s x \varphi(x)$ by $\exists x(P_s(x) \land \varphi(x))$. We denote this translation of a formula φ in many-sorted logic by φ^*.[57]

On the semantic side we can convert a many-sorted structure $\mathfrak{A}$ into a one-sorted structure $\mathfrak{A}^*$. We do this by taking as the new domain the union of the sorted domains

$$\mathrm{dom}(\mathfrak{A}^*) := \bigcup_{s \in \boldsymbol{S}} \mathrm{dom}_s(\mathfrak{A})$$

and by assigning the sorted domains $\mathrm{dom}_s(\mathfrak{A})$ as the interpretations of the new predicate symbols P_s, i.e.,

$$P_s^{\mathfrak{A}^*} := \mathrm{dom}_s(\mathfrak{A}).$$

[56] Sentences are formulas without free variables.

[57] Note that this translation yields the formulas as in the first-order formalization of Chapter 2.

Furthermore, we have to arbitrarily extend the functions $f^{\mathfrak{A}}$ to the new domain,[58] but we can keep all predicates and constants fixed.

This procedure yields a translation of our many-sorted setting into first-order logic, where predicates P_s indicate the membership in different sorts. But what makes this useful at all is the following well-known lemma:[59]

Lemma 4.4 (Translation Lemma). *A many-sorted sentence φ is true in a structure $\mathfrak{A}$ if and only if its translation φ^* is true in the translated structure $\mathfrak{A}^*$, i.e.,*

$$\mathfrak{A} \models \varphi \iff \mathfrak{A}^* \models \varphi^*.$$

Lemma 4.4 enables us to transfer certain results from first-order model theory to our many-sorted setting, so, for instance, the preservation result mentioned in the beginning of this subsection (Lemma 4.3). The translated result then is:[60]

Theorem 4.5 (Weak Preservation Theorem). *Many-sorted $\forall_1$-sentences are preserved in (many-sorted) substructures, i.e., if $\mathfrak{A}$ is a (many-sorted) substructure of $\mathfrak{B}$ then*

$$\mathfrak{B} \models \varphi \implies \mathfrak{A} \models \varphi$$

for any many-sorted $\forall_1$-sentence φ.

Proof. Let $\mathfrak{A}$ be a (many-sorted) substructure of $\mathfrak{B}$ and φ a many-sorted $\forall_1$-sentence. By the Translation Lemma 4.4 we directly get that the translated formula φ^* is true in the translated structure $\mathfrak{B}^*$. Now it is necessary to see that the translated structure $\mathfrak{A}^*$ is a substructure of $\mathfrak{B}^*$ and that the translated formula φ^* is still (at least logically equivalent to) a $\forall_1$-sentence. The proof of the former is entirely straightforward and only uses the fact that $\mathfrak{A}$ is a substructure of $\mathfrak{B}$.[61] The latter can also be seen immediately when recalling how the syntactic translation worked: the only change was a replacement of all sorted universal quantifiers $\forall_s x\varphi(x)$ by

$$\forall x(P_s(x) \to \varphi(x)) \equiv \forall x(\neg P_s(x) \vee \varphi(x))$$

and so — since $\neg P_s(x)$ as well as $\varphi(x)$ are $\forall_1$-formulas — the whole formula is logically equivalent to a formula in $\forall_1$-form.[62]

[58] Even though this extension is not unique, we will speak of *the* translation of a many-sorted structure as any extension will work and will have the same properties.

[59] Lemma 43A in Enderton [22]; Theorem 8.2.2. in Manzano [40].

[60] In some places the proof will remain slightly informal, but we will later give a direct proof as part of the proof of Theorem 4.13.

[61] One has to make sure though that the functions are extended to the full domain in the same way for $\mathfrak{A}^*$ and $\mathfrak{B}^*$, which is easily possible (see also Footnote 58).

[62] A more rigorous proof would proceed via induction on the complexity of the formula and involve a stronger statement about formulas instead of sentences.

Therefore, we can apply Lemma 4.3 to the first-order structure $\mathfrak{A}^* \subseteq \mathfrak{B}^*$ and get that φ^* is also true in the substructure $\mathfrak{A}^*$. Now it is just one backward application of the Translation Lemma 4.4 to get to $\mathfrak{A} \models \varphi$ and we are done.

$$\begin{tikzcd}
\mathfrak{B} \models \varphi \arrow[r, "\text{Lemma 4.4}"] \arrow[d, "\mathfrak{A} \subseteq \mathfrak{B}"] & \mathfrak{B}^* \models \varphi^* \arrow[d, "\text{Lemma 4.3 } (\mathfrak{A}^* \subseteq \mathfrak{B}^*)"] \\
\mathfrak{A} \models \varphi & \mathfrak{A}^* \models \varphi^* \arrow[l, "\text{Lemma 4.4}"']
\end{tikzcd}$$

□

Theorem 4.5 can indeed be interpreted as a general version of an inductive step like the one (Lemma 3.9) presented in Section 3.3: it implies that when we have a finite set X with a relation $\dot{\geq}$ and the set of all non-empty subsets $\mathcal{X}$ with another relation $\succeq$, then all $\forall_1$-axioms are preserved in substructures. That is, if $\dot{\geq}$ and $\succeq$ fulfill certain $\forall_1$-axioms, then so do the restrictions to subsets of X and $\mathcal{X}$, respectively.

Cast into the familiar shape of Lemma 3.9, we can formulate the following corollary:

Corollary 4.6. *If X is a linearly ordered set and there is a binary relation on $\mathcal{X} = 2^X \setminus \{\emptyset\}$ that satisfies a given set of $\forall_1$-axioms, then we can, for any $n < |X|$, $n \in \mathbb{N}$, find another linearly ordered set Y with n elements only, as well as a corresponding binary relation on $\mathcal{Y} := 2^Y \setminus \{\emptyset\}$ satisfying the same set of axioms.*

Proof. We can view $X \cup \mathcal{X}$ as an MSLSP-structure[63] and by Theorem 4.5 all $\forall_1$-axioms are preserved in its substructures, which any $Y \cup \mathcal{Y}$ with $Y \subset X$ is one of.[64] Since the axioms of a linear order are $\forall_1$-axioms, too, it follows that the restriction of $\dot{\geq}$ to Y is also a linear order. □

The case of a weak order on $\mathcal{X}$ is now just a special case of this result since weak orders are defined by $\forall_1$-axioms.

Reading the previous corollary contrapositively, we obtain an even clearer form of our (still weak) universal step:

Corollary 4.7 (Weak Universal Step). *If, for any linearly ordered set Y with n elements, there exists no binary relation on $\mathcal{Y} = 2^Y \setminus \{\emptyset\}$ satisfying a given set of $\forall_1$-axioms, then for any linearly ordered set X with $|X| \geq n$ there is also no binary relation on $\mathcal{X} = 2^X \setminus \{\emptyset\}$ that satisfies this set of axioms.*

[63] With all function and relation symbols interpreted in the usual/natural way.

[64] The language MSLSP (with its natural interpretation) was chosen to guarantee this property that, for all subsets $Y \subseteq X$, the structure $Y \cup \mathcal{Y}$ is a substructure of $X \cup \mathcal{X}$. See also Remark 4.8.

Remark 4.8. When we say "a given set of $\forall_1$-axioms", then this deserves some further explanation. Of course, we mean axioms that are $\forall_1$ in MSLSP. But the reader will remember that we allowed additional relations and functions to be added, which were not specified in the signature above. This holds some hidden challenges. For instance, it is not possible to include a predicate *isWholeSet*, which is true of the whole domain only, a function $(\cdot)^c$ for the complement, or even just the constant symbol $\dot{X}$ referring to the whole domain, since all three would (in their natural interpretation) prevent $Y \cup \mathcal{Y}$ to be a substructure: for example, for any $Y \subset X$, while isWholeSet(Y) (or equivalently, $Y = \dot{X}$) is false in $X \cup \mathcal{X}$, it is true in $Y \cup \mathcal{Y}$. Similarly and as mentioned before in Section 4.2, we run into problems when including functions like $\cap$, $\setminus$ or $(\cdot)^c$, which are (in their natural interpretation) not functions in the strict sense in a structure like $X \cup \mathcal{X}$ as they can produce the empty set, which is not in $\mathcal{X}$. Therefore, some attention has to be paid when adding new relation or function symbols to the language in order to capture more axioms.

Summing up, we have shown a universal step (and thus a general inductive step) for the class of $\forall_1$-axioms using the tools of classical model theory. We will see in the next section, however, that the result is not yet strong enough to capture all axioms needed for the Kannai-Peleg Theorem. Thus, we are going to prove a further generalization to an even larger class of axioms, which will then also include all axioms of the Kannai-Peleg as well as of other possibility and impossibility theorems.

Note that we have not made use of any domain-specific features so far. The (so far unmentioned) other direction of the Łoś-Tarski Theorem, stating that if a sentence is preserved in substructures then it must be a $\forall_1$-sentence, tells us that we can indeed not hope for more than the class of $\forall_1$-axioms on this general level, but have to make use of properties of our particular domain now.

4.4 Universal Step Based on the Domain Structure

Let us recall the representation of the Gärdenfors principle (`GF`) in MSLSP (as given in Example 4.2):

$$\forall_\sigma A \forall_\varepsilon x \left[(\forall_\varepsilon a (a \in A \rightarrow x \mathrel{\dot{>}} a)) \rightarrow A \cup \{x\} \succ A \right]$$
$$\forall_\sigma A \forall_\varepsilon x \left[(\forall_\varepsilon a (a \in A \rightarrow a \mathrel{\dot{>}} x)) \rightarrow A \succ A \cup \{x\} \right]$$

It is tempting to think that (`GF`) is covered by Corollary 4.7, which was applicable to $\forall_1$-axioms. Unfortunately, however, at closer inspection one notices that (`GF`) actually is not of this form. Converting the axiom (we do this exemplarily for (`GF1`)) to prenex normal form reveals that the innermost quantifier is actually an existential quantifier:

$$\begin{aligned}
&\forall_\sigma A \forall_\varepsilon x \left[(\forall_\varepsilon a(a \in A \to x \mathrel{\dot{>}} a)) \to A \cup \{x\} \succ A\right] \\
\equiv\ &\forall_\sigma A \forall_\varepsilon x \left[\neg\, (\forall_\varepsilon a(a \notin A \vee x \mathrel{\dot{>}} a)) \vee A \cup \{x\} \succ A\right] \\
\equiv\ &\forall_\sigma A \forall_\varepsilon x \left[\exists_\varepsilon a(a \in A \wedge x \mathrel{\dot{\not>}} a) \vee A \cup \{x\} \succ A\right] \\
\equiv\ &\forall_\sigma A \forall_\varepsilon x \exists_\varepsilon a \left[(a \in A \wedge x \mathrel{\dot{\not>}} a) \vee A \cup \{x\} \succ A\right].
\end{aligned}$$

Therefore, we have to find a stronger result than what classical model theory can offer us. The idea is to be able to preserve a larger class of axioms by making use of our problem-specific features: like, for instance, the element-set framework. Thus, we define the concepts of a *structure for set preferences* as well as *subset-consistent* substructures:

Definition 4.9. *An MSLSP-structure* $\mathfrak{B}$ *is a* structure for set preferences *if it fulfills the following criteria:*

1. $\mathrm{dom}_\sigma(\mathfrak{B}) \subseteq 2^{\mathrm{dom}_\varepsilon(\mathfrak{B})}$, *i.e., the domain of sort* σ *contains only sets of elements from* $\mathrm{dom}_\varepsilon(\mathfrak{B})$.
2. *The relation symbol* $\in$ *of type* $\langle \varepsilon, \sigma \rangle$ *is interpreted in its natural way.*

If a substructure $\mathfrak{A}$ *of a structure for set preferences* $\mathfrak{B}$ *is a structure for set preferences, too, then it is called a* subset-consistent *substructure.*

Note that in a substructure $\mathfrak{A}$ of a structure for set preferences $\mathfrak{B}$ we have

$$\in^{\mathfrak{A}} = \in^{\mathfrak{B}} |_{\mathrm{dom}(\mathfrak{A})},$$

i.e., the symbol $\in$ must be interpreted as the restriction of its interpretation in $\mathfrak{B}$. Hence, it is sufficient to fulfill the first condition for being a *subset-consistent* substructure of $\mathfrak{B}$.

These two semantic conditions are all we need in order to generalize our previous result to a larger class of axioms. But what are the axioms that we can now treat? Let us look at the (purely syntactic) definition first and then explain the reasons for choosing this particular class.

Definition 4.10. *The class of* existentially set-guarded *(ESG) formulas is the smallest class of MSLSP-formulas recursively defined as follows:*

- *all quantifier-free formulas are ESG,*
- *if* $\psi(\bar{x})$ *and* $\psi'(\bar{x})$ *are ESG, then* $\varphi(\bar{x}) := (\psi \wedge \psi')(\bar{x})$ *as well as* $\varphi'(\bar{x}) := (\psi \vee \psi')(\bar{x})$ *are ESG,*
- *if* $\psi(y, \bar{x})$ *is ESG, then* $\varphi(\bar{x}) := \forall_s y \psi(y, \bar{x})$ *is ESG for any sort* $s \in \{\varepsilon, \sigma\}$,
- *if* $\psi(y, \bar{x})$ *is ESG, then* $\varphi(\bar{x}) := \exists_\varepsilon y(y \in t(\bar{x}) \wedge \psi(y, \bar{x}))$, *where* t *is a term of sort* σ, *is ESG.*

The atomic formulas $y \in t(\bar{x})$ *of the last condition are called the* set-guards *of the respective quantifiers.*

If we leave out the last condition this gives us exactly the $\forall_1$-formulas in MSLSP. Thus, the ESG formulas are a strictly (and much) larger class. It consists of all MSLSP-formulas that only contain set-guarded existential quantifiers $\exists_\varepsilon$ of sort ε, and no existential quantifiers $\exists_\sigma$ of sort σ at all.

Note that when we write $\varphi(\bar{x})$, we do not necessarily mean that φ contains all the variables in the sequence $\bar{x} = (x_0, x_1, x_2, \dots)$, but just that all (free) variables of φ are among the ones in $\bar{x}$.[65] Soon we will also use the notation $\varphi[\bar{a}]$ with $\bar{a}$ being a sequence of elements, which will mean that the elements $a_0, a_1, a_2, \dots$ are assigned to the variables $x_0, x_1, x_2, \dots$.

Intuitively, we do the following: in our axioms we now also allow existential quantifiers (but only for elements, i.e., of sort ε) as long as they are "guarded" by subformulas saying that the respective witness belongs to some set. The sets can also be unions of sets or formed in a different way by the term t. The important part is that when moving from a structure to a substructure this set-guard now guarantees that the witness of the existential quantifier is not lost. This is because the witness has to be within a set (as required by the set-guard) that will be situated in the substructure.[66]

Before we get to the formal proof of this claim, let us look at some examples of sentences. We first check that we have now indeed covered at least all axioms of the Kannai-Peleg Theorem. Recall that all order axioms as well as the axiom of independence were already in the $\forall_1$-class and are hence also ESG. We were only missing the Gärdenfors principle (`GF`):

Example 4.11. The axiom (`GF1`) (and similarly (`GF2`)) is an ESG sentence:

$$\begin{array}{rll}
 & x \mathrel{\dot{\not\succ}} a & \text{(quantifier-free)} \\
\exists_\varepsilon a(a \in A \;\wedge & x \mathrel{\dot{\not\succ}} a) & \text{(adding } \exists_\varepsilon\text{)} \\
\exists_\varepsilon a(a \in A \;\wedge & x \mathrel{\dot{\not\succ}} a) \vee A \cup \{x\} \succ A & \text{(}\vee\text{ with quantifier-free)} \\
\forall_\sigma A \forall_\varepsilon x[\exists_\varepsilon a(a \in A \;\wedge & x \mathrel{\dot{\not\succ}} a) \vee A \cup \{x\} \succ A] & \text{(adding } \forall_s\text{).}
\end{array}$$

Remembering the proof of Lemma 3.9, where we showed that (`GF`) still holds after removing an element from X, and considering the last line of the above example, one can understand why removing elements from X does not affect this axiom. For universal quantifiers a restriction of the domain was no problem anyway as we have seen in the previous Section 4.3. But also the existential witness is not lost: if we suppose it had been removed for some set A, then so would have been the set A itself, as the σ-domain can only contain sets of elements from the ϵ-domain by Definition 4.9. Contradiction!

That we cannot just allow arbitrary existential quantifiers without set-guards can be seen when considering the following example, which shows a very easy sentence with unguarded existential quantifiers that is not preserved in substructures.

[65] See Section 1.3 of Hodges [34] for a completely formal description.

[66] That the set is still interpreted like in the superstructure is guaranteed by the properties of a substructure.

Example 4.12. The MSLSP-sentence (axiom)

$$\exists_\varepsilon x \exists_\varepsilon y \exists_\varepsilon z \, [x \neq y \wedge x \neq z \wedge y \neq z] \, ,$$

which says that there are at least three distinct elements in the ε-domain of a structure for set preferences, is clearly not preserved in substructures: it holds in all structures for set preferences $\mathfrak{B}$ with at least three elements in $\mathrm{dom}_\varepsilon(\mathfrak{B})$, but fails to hold in any of its substructures $\mathfrak{A}$ with less than three elements in $\mathrm{dom}_\varepsilon(\mathfrak{A})$.

After these examples the reader should have developed some understanding of why ESG sentences are preserved in substructures and why we cannot allow much more. The formal proof of our Preservation Theorem will explain the first part further.

Note that, apart from the case of the existential quantifier, the proof is practically the direct proof of Theorem 4.5, which can be carried out on model-theoretic grounds only. It is just the last part of this proof (the induction step for the existential quantifier) that requires the syntactic and semantic restriction, which we can allow because of our particular problem domain.

Theorem 4.13 (Preservation Theorem). *ESG sentences are preserved in subset-consistent substructures, i.e., if $\mathfrak{A}$ is a subset-consistent substructure of a structure for set preferences $\mathfrak{B}$ then*

$$\mathfrak{B} \models \varphi \implies \mathfrak{A} \models \varphi$$

for any ESG sentence φ.

Proof. We prove a stronger statement for ESG *formulas* (instead of sentences) by induction on the complexity of the formula:

> *If $\mathfrak{A}$ is a subset-consistent substructure of a structure for set preferences $\mathfrak{B}$ then*
> $$\mathfrak{B} \models \varphi[\bar{a}] \implies \mathfrak{A} \models \varphi[\bar{a}]$$
> *for any ESG formula $\varphi(\bar{x})$ and any tuple $\bar{a}$ of elements from* $\mathrm{dom}(\mathfrak{A})$ *(matching the types of $\bar{x}$).*

So let $\mathfrak{B}$ be a structure for set preferences with a subset-consistent substructure $\mathfrak{A}$, let $\varphi(\bar{x})$ be an ESG formula and, furthermore, let $\bar{a}$ be a tuple of elements from $\mathrm{dom}(\mathfrak{A})$ (matching the types of $\bar{x}$).

Quantifier-free Formulas: If $\varphi(\bar{x})$ is quantifier-free, a routine but tedious proof leads to the desired results. One has to carry out a few nested inductions on the complexity of terms and formulas, and examples of such proofs can be found in any textbook of Model Theory (see, e.g., Theorem 1.3.1. in Hodges [34]). First, one shows by one induction that terms are interpreted in the substructure $\mathfrak{A}$ as they are interpreted in its superstructure $\mathfrak{B}$, i.e.,

$$t^{\mathfrak{A}}[\bar{a}] = t^{\mathfrak{B}}[\bar{a}] \tag{4.1}$$

for all terms $t(\bar{x})$. This practically immediately follows from the definition of a substructure.

Then one proceeds by another induction proving that atomic formulas hold in $\mathfrak{A}$ if and only if they hold in $\mathfrak{B}$, i.e.,

$$\mathfrak{A} \models \psi[\bar{a}] \iff \mathfrak{B} \models \psi[\bar{a}] \tag{4.2}$$

for all atomic formulas $\psi(\bar{x})$. As a typical example, suppose that $\varphi(\bar{x})$ is of the form $R(s(\bar{x}), t(\bar{x}))$, where R is a relation symbol and $s(\bar{x})$ as well as $t(\bar{x})$ are terms (matching the type of R). Assume $\mathfrak{A} \models R(s[\bar{a}], t[\bar{a}])$, i.e., it holds that $R^{\mathfrak{A}}(s^{\mathfrak{A}}[\bar{a}], t^{\mathfrak{A}}[\bar{a}])$. By (4.1) this is equivalent to $R^{\mathfrak{A}}(s^{\mathfrak{B}}[\bar{a}], t^{\mathfrak{B}}[\bar{a}])$. Since furthermore $R^{\mathfrak{A}} = R^{\mathfrak{B}}|_{\mathrm{dom}(\mathfrak{A})}$, we even have an equivalence with $R^{\mathfrak{B}}(s^{\mathfrak{B}}[\bar{a}], t^{\mathfrak{B}}[\bar{a}])$, which is just another way of saying $\mathfrak{B} \models R(s[\bar{a}], t[\bar{a}])$.

Finally, one proves the claim for any quantifier-free formula by carrying out induction steps for conjunction $\wedge$, disjunction $\vee$ and negation $\neg$. Note that the step for $\neg$ is why we required both directions in (4.2).

Conjunction and Disjunction: We only show the part for conjunction here; the one for disjunction is completely analogous. If $\varphi(\bar{x})$ is of the form $\psi(\bar{x}) \wedge \psi'(\bar{x})$ and furthermore $\mathfrak{B} \models \varphi[\bar{a}]$, then both $\psi[\bar{a}]$ and $\psi'[\bar{a}]$ must be true in $\mathfrak{B}$. By the induction hypothesis, this carries over to $\mathfrak{A}$ and we get $\mathfrak{A} \models \psi[\bar{a}] \wedge \psi'[\bar{a}]$.

Universal Quantification: If $\varphi(\bar{x})$ is of the form $\forall_s y \psi(y, \bar{x})$ with sort $s \in \{\sigma, \varepsilon\}$ and furthermore $\mathfrak{B} \models \varphi[\bar{a}]$, then for all b of sort s in $\mathrm{dom}_s(\mathfrak{B})$ we have that $\mathfrak{B} \models \psi(b, \bar{a})$. Since $\mathrm{dom}_s(\mathfrak{A}) \subseteq \mathrm{dom}_s(\mathfrak{B})$ we can use the induction hypothesis and obtain $\mathfrak{A} \models \psi(b, \bar{a})$ for any $b \in \mathrm{dom}_s(\mathfrak{A})$. This is the same as saying $\mathfrak{A} \models \forall_s y \psi(y, \bar{a})$, i.e., $\mathfrak{A} \models \varphi[\bar{a}]$.

Existential Quantification: If $\varphi(\bar{x})$ is of the form $\exists_\varepsilon y[y \in t(\bar{x}) \wedge \psi(y, \bar{x})]$, where $t(\bar{x})$ is a term of sort σ, and furthermore $\mathfrak{B} \models \varphi[\bar{a}]$, then there must exist an element b in $\mathrm{dom}_\varepsilon(\mathfrak{B})$ such that

$$\mathfrak{B} \models (y \in t(\bar{x}) \wedge \psi(y, \bar{x}))\, [b, \bar{a}], \text{ i.e.,} \mathfrak{B} \models y \in t(\bar{x})[b, \bar{a}] \text{ and } \mathfrak{B} \models \psi[b, \bar{a}].$$

Hence, if we can show that b is in the ε-domain of $\mathfrak{A}$ and not just of $\mathfrak{B}$, then it follows by the induction hypothesis that also

$$\mathfrak{A} \models \psi[b, \bar{a}],$$

since then $(b, \bar{a})$ is a tuple of elements of $\mathfrak{A}$.

Since $\in$ is interpreted naturally in the structure for set preferences $\mathfrak{B}$ and, additionally, $t^{\mathfrak{A}}[\bar{a}] = t^{\mathfrak{B}}[\bar{a}]$ by (4.1), the statement $\mathfrak{B} \models y \in t(\bar{x})[b, \bar{a}]$ boils down to

$$b \in t^{\mathfrak{B}}[\bar{a}] = t^{\mathfrak{A}}[\bar{a}]. \tag{4.3}$$

The fact that b it is an element of $\mathrm{dom}_\varepsilon(\mathfrak{A})$ (and not just of $\mathrm{dom}_\varepsilon(\mathfrak{B})$) is now implied by $t^{\mathfrak{A}}[\bar{a}]$ being in $\mathrm{dom}_\sigma(\mathfrak{A})$, together with $\mathfrak{A}$ being a subset-consistent substructure:

$$\begin{aligned} & b \in t^{\mathfrak{A}}[\bar{a}] \in \mathrm{dom}_\sigma(\mathfrak{A}) \overset{(*)}{\subseteq} 2^{\mathrm{dom}_\varepsilon(\mathfrak{A})} \\ \Longrightarrow\ & b \in t^{\mathfrak{A}}[\bar{a}] \in 2^{\mathrm{dom}_\varepsilon(\mathfrak{A})} \\ \Longrightarrow\ & b \in t^{\mathfrak{A}}[\bar{a}] \subseteq \mathrm{dom}_\varepsilon(\mathfrak{A}), \end{aligned}$$

where $(*)$ marks the spot where the subset-consistency of $\mathfrak{A}$ is used. Hence, we can, as indicated before, apply the induction hypothesis to $\mathfrak{B} \models \psi[b, \bar{a}]$ and obtain $\mathfrak{A} \models \psi[b, \bar{a}]$. Together with $b \in t^{\mathfrak{A}}[\bar{a}]$ it follows that

$$\mathfrak{A} \models \exists_\varepsilon y(y \in t(\bar{x}) \wedge \psi(y, \bar{x}))[\bar{a}].$$

This way we are done with the proof of the stronger claim (about formulas), which implies the claim of the theorem (about sentences). □

We are now almost ready to apply this theorem to our setting. But we want to make one further strengthening first. The following lemma will enable us to also use axioms that are not ESG themselves, but just equivalent in all structures for set preferences to an ESG sentence. Such axioms will be referred to as *ESG-equivalent* sentences. In order to maintain a slightly higher level of generality we prove the lemma for arbitrary classes Γ and not just for the class of all ESG sentences.

Lemma 4.14. *If Γ is a class of sentences that are preserved in subset-consistent substructures, then also all sentences that are* equivalent in structures for set preferences *to a sentence in Γ are preserved in subset-consistent substructures.*

Proof. Let Γ be a class of sentences that are preserved in subset-consistent substructures. Furthermore, let ψ be a sentence that is equivalent in structures for set preferences to a sentence $\varphi \in \Gamma$, i.e., such that, for all structures for set preferences $\mathfrak{M}$,

$$\mathfrak{M} \models \psi \iff \mathfrak{M} \models \varphi. \tag{4.4}$$

Let now $\mathfrak{A}$ be a subset-consistent substructure of a structure for set preferences $\mathfrak{B}$, and suppose $\mathfrak{B} \models \psi$. This then immediately implies $\mathfrak{B} \models \varphi$ by (4.4) since $\mathfrak{B}$ is a structure for set preferences. Hence, we can now use the assumption that sentences from Γ are preserved in subset-consistent substructures, and obtain $\mathfrak{A} \models \varphi$. This, in turn, has $\mathfrak{A} \models \psi$ as a consequence (by (4.4) again and since subset-consistent substructures are structures for set preferences themselves), which completes the proof of this lemma. □

In particular, of course, Lemma 4.14 applies to sentences that are *logically equivalent* to a sentence in Γ. This is the case as logically equivalent sentences are equivalent in *all* structures by definition, and hence in particular in all structures for set preferences.

Now, we finally state and prove the corollary applying our most general result to the particular problem domain of *ranking sets of objects*.

Corollary 4.15. *If X is a linearly ordered set and there is a binary relation on $\mathcal{X} = 2^X \setminus \{\emptyset\}$ that satisfies a given set of ESG (or ESG-equivalent) axioms, then we can, for any $n < |X|$, $n \in \mathbb{N}$, find another linearly ordered set Y with n elements only, as well as a corresponding binary relation on $\mathcal{Y} := 2^Y \setminus \{\emptyset\}$ satisfying the same set of axioms.*

Proof. We can view $X \cup \mathcal{X}$ as a structure for set preferences[67] and by Theorem 4.13 together with Lemma 4.14 all ESG(-equivalent) axioms are preserved in its subset-consistent substructures, which any $Y \cup \mathcal{Y}$ with $Y \subset X$ is one of (with respect to the signature of MSLSP).[68] Since the axioms of a linear order are $\forall_1$-axioms and therefore ESG, too, it follows that the restriction of $\dot{\geq}$ to Y is also a linear order. □

Reading the previous corollary contrapositively, we obtain an even clearer form of our universal step:

Corollary 4.16 (Universal Step). *If, for any linearly ordered set Y with n elements, there exists no binary relation on $\mathcal{Y} = 2^Y \setminus \{\emptyset\}$ satisfying a given set of ESG (or ESG-equivalent) axioms, then for any linearly ordered set X with $|X| \geq n$ there is also no binary relation on $\mathcal{X} = 2^X \setminus \{\emptyset\}$ that satisfies this set of axioms.*

On the basis of Corollary 4.16 we can finally do what we had been hoping for: in order to prove new impossibility theorems and check existing ones, we only have to look at their base cases (as long as all axioms involved are expressible in MSLSP and ESG-equivalent).[69] These small instances can be efficiently checked on a computer (as we have done it for the case of the Kannai-Peleg Theorem in Sections 3.3.2 and 3.3.3) and so especially from a practical perspective this can lead to an enormous boost in the speed of theorem discovery. We will pursue this idea further in the next chapter, where we present an automated and exhaustive theorem search as well as its results including quite a few new impossibility theorems.

[67] With all function and relation symbols interpreted in the usual/natural way.

[68] As described in Remark 4.8, one has to be careful when adding additional functions and relations as they might prevent $Y \cup \mathcal{Y}$ from being a substructure.

[69] Even though not needed for our purposes, the universal step even applies to infinite domains X. It therefore also reduces the infinite versions of impossibility theorems (cf. Remark 1.7) to small instances. Also note that a single application of the step is sufficient and induction is not needed anymore.

4.5 Summary

In this chapter we have seen how tools from model theory can be used to obtain a universal step, which reduces general impossibilities to small instances. After defining the natural (many-sorted first-order) language MSLSP for expressing axioms formally, we were able to prove a weak version of the universal step only using results from model theory without any application of problem-specific features. This weak version, however, does only cover $\forall_1$-axioms, which does not suffice to treat the Kannai-Peleg Theorem.

Therefore, a stronger version of the universal step was required and could be found using domain features that are specific to *ranking sets of objects*. Now even existential quantifiers can be allowed, as long as they are "guarded" by a term of a particular logical form. The class of formulas covered under the universal step was introduced as the class of *existentially set-guarded* (or short: ESG) formulas. This class is large enough to subsume all axioms from the literature that we were able to formalize in the language MSLSP.

5

Automated and Exhaustive Theorem Search

5.1 Introduction

The main result of the previous chapter, the Preservation Theorem 4.13, allowed us to derive a universal step (Corollary 4.16), which we can now use to prove impossibilities just by checking their base cases. Since this verification of small instances can be done very efficiently on a computer (as we have seen in Section 3.3), we naturally get a range of applications in the area of *ranking sets of objects*.

Not only can we verify known impossibilities by formalizing their axioms and then feeding base case instances to a SAT solver, but we can also weaken or alter axioms slightly in order to find stronger or similar results. It can sometimes even be possible to show certain implications instead of impossibilities.[70]

Using these techniques for single theorems is likely to produce good results and might be a helpful tool from a practical perspective, but we can do even more: we are now going to present a fully automated and exhaustive theorem search for impossibility theorems. Our theorem search will, for a given set of axioms, systematically check which of its subsets are inconsistent and from which smallest domain size these impossibilities occur, thereby completely automatically identifying all impossibility theorems that the given axioms

[70] If we want to show that $\mathcal{A} \to \mathcal{B}$, then we can do this with a proof by contradiction, i.e., by showing that $\mathcal{A} \wedge \neg\mathcal{B}$ is "impossible". A requirement for this to work with our method, however, is that $\neg\mathcal{B}$ can be written as an existentially set-guarded (ESG) formula[71], which fails for most axioms. For instance, all axioms with an unguarded universal quantifier of type σ will produce an unguarded existential quantifier of type σ in their negation, which ESG formulas are not allowed to have.

can produce.[72] In this sense, the search method is exhaustive on the space of given axioms.

For testing the fruitfulness of this approach we ran the search on a set of 20 axioms from the literature on *choice under complete uncertainty*, i.e., the interpretation of our setting in which the decision maker will finally only receive one of the alternatives in the sets and does not know which it will be.[73] Of course, the very same search method can also be run with arbitrary other ESG axioms in the field of *ranking sets of objects*, including, for instance, the ones of the interpretation of sets as *opportunity sets* from which the decision maker can himself select his favorite outcome, or, similarly, with axioms for the case of assuming that the agent receives the whole set of alternatives.

Our search algorithm returned a total of 84 impossibilities, of which a few were known already (and are hence now automatically verified), some are just direct consequences of others, but also some are surprising and new. Especially finding an impossibility that had wrongly been published as a possibility earlier underlines the usefulness of this approach as well as the complexity of the obtained results.

We proceed by first describing our implementation of the theorem search and how it exactly progresses. We then first show and briefly introduce the 20 axioms we used in our final theorem search, before we list all minimal impossibilities as well as potential possibilities that were found, and make some general observations. Finally, we are going to give manual proofs of selected results in order to demonstrate the practical utility of the search method and also to shed some more light on the impossibilities themselves.

5.2 Technique and Implementation

As indicated in the introduction, our search method systematically decides whether subsets of a given axioms set are compatible or incompatible. We will therefore in the following refer to axiom subsets as *problems*, and for a particular domain size we will speak of a *problem instance*.

We designed a computer program (again in *Java* [67]) that can selectively instantiate axioms from a given set and for a given domain size, by just extending our program from Section 3.3.3. After this generation of a problem instance it can then immediately be passed to a SAT solver, which returns whether it is a “possible” instance,[74] or an “impossible” one. We implemented interfaces for the commonly used solvers *PrecoSAT* [13] and *zChaff* [61] – the latter enabling us to automatically verify unsatisfiabilities on the basis of a

[72] Since for practical reasons we can only check base cases up to a certain domain size $|X| = n$, there could theoretically be more impossibilities hidden that only occur from larger domain sizes on.

[73] See also Section 1.2 for an introduction to this idea.

[74] Using the module from Section 3.4.1 we can then also print a satisfying weak order.

generated proof trace, whereas the former is usually faster in practice, but does not have this feature.[75]

We could now just run this program on all possible problem instances individually and collect the results. Note, however, that on a space of 20 axioms with a maximal domain size of eight elements, we already have to deal with a total of $(2^{20} - 1) \cdot 8 \approx 8.400.000$ problem instances. If each of them just requires a running time of one second,[76] the whole job would take roughly 100 days.

Therefore, we then designed a scheduler that makes sure all axiom subsets are treated for all domain sizes in a sensible order. The order in which we check the problem instances has a big effect on the overall running time because one can make use of a combination of four different effects:

(1) if a set of axioms is *inconsistent* at domain size $|X| = n$, then it will also be *inconsistent* for all *larger* domain sizes $|X| > n$ (Corollary 4.16),
(2) if a set of axioms is *inconsistent* at domain size $|X| = n$, then also all its (axiom) *supersets* are *inconsistent* at this domain size $|X| = n$,
(3) if a set of axioms is *consistent* at domain size $|X| = n$, then it will also be *consistent* for all *smaller* domain sizes $|X| < n$. (Corollary 4.15), and
(4) if a set of axioms is *consistent* at domain size $|X| = n$, then also all its (axiom) *subsets* are *consistent* at this domain size $|X| = n$.

Since larger instances require exponentially more time (there are exponentially more variables in the satisfiability problem due to exponentially more subsets in $\mathcal{X}$), we start our search at the smallest domain size and then after completely solving this “level” move on to the next domain size.[77]

On each level, as soon as we find an impossibility, we can, by condition (2), mark all axiom supersets as impossible at the current domain size (if they had not been found to be impossible at a smaller domain size already). In order to use this mechanism as efficiently as possible, we must check small axiom sets first.

But also the dual approach of starting from large axiom sets and marking all axiom subsets as “compatible” as soon as we find a possibility (condition (4)), is an option. In experiments, we found that the best performance can be achieved when combining these two approaches and so we decided to run the search in alternating directions (changing every 15 minutes): from large axiom sets to small ones and the other way around.

On a new level, only problems have to be considered that still have the status “possible” because of condition (1) in the enumeration above. This further increases the speed of our search method.

[75] *PrecoSAT* won the gold medal in the applications track of the *SAT 2009* competition [60], while the SAT solver *zChaff* could only claim the gold medal in the special *certified UNSAT* track of the *SAT 2005* competition [59].

[76] In our tests, especially larger instances required much more time to be solved on average.

[77] It is for this reason that condition (3) is not very helpful in practice.

From a practical side, the theorem search comes with the limitations of only being able to treat at most 21 axioms at the same time[78] and at most a domain size of eight elements.[79] But with better memory management and improved versions of the SAT solvers, these (practical) boundaries should be extendable.

5.3 Implemented Axioms

We ran our theorem search on a space of 20 axioms from the literature concerning *choice under complete uncertainty*. These axioms (or versions of them) can actually all be obtained from the corresponding section of a survey by Barberà, Bossert and Pattanaik [11]. We will proceed by discussing the axioms and presenting them in their usual form as well as in MSLSP (to verify that we can apply our universal step (Corollary 4.16).[80] The axioms' propositional forms, which were needed for the base case verification, are included in Appendix B.4.

The first axioms we want to look at are the *order axioms*. For one, there are the axioms describing the linear order $\dot{\geq}$ on X, for another, the ones describing a weak order $\succeq$ on $\mathcal{X} = 2^X \setminus \{\emptyset\}$. The former will just be denoted by ($\mathtt{LIN}_\varepsilon$), whereas the latter are split up into their three components reflexivity ($\mathtt{REFL}_\sigma$), completeness ($\mathtt{COMPL}_\sigma$) and transitivity ($\mathtt{TRANS}_\sigma$), which are then treated as separate axioms in order to investigate which parts are actually necessary for impossibilities. The axioms in their intuitive form and their formal representations in MSLSP are:

$$(\mathtt{LIN}_\varepsilon) \quad x \mathrel{\dot{\geq}} x \text{ for all } x \in X \qquad \text{(reflexivity)}$$
$$\cong \forall_\varepsilon x \left[x \mathrel{\dot{\geq}} x\right]$$

$$x \mathrel{\dot{\geq}} y \lor x \mathrel{\dot{\leq}} y \text{ for all } x \neq y \in X \qquad \text{(completeness)}$$
$$\cong \forall_\varepsilon x \forall_\varepsilon y \left[x \neq y \to (x \mathrel{\dot{\geq}} y \lor x \mathrel{\dot{\leq}} y)\right]$$

$$x \mathrel{\dot{\geq}} y \land y \mathrel{\dot{\geq}} z \Rightarrow x \mathrel{\dot{\geq}} z \text{ for all } x, y, z \in X \qquad \text{(transitivity)}$$
$$\cong \forall_\varepsilon x \forall_\varepsilon y \forall_\varepsilon z \left[(x \mathrel{\dot{\geq}} y \land y \mathrel{\dot{\geq}} z) \to x \mathrel{\dot{\geq}} z\right]$$

$$x \mathrel{\dot{\geq}} y \land y \mathrel{\dot{\geq}} x \Rightarrow x = y \text{ for all } x, y \in X \qquad \text{(antisymmetry)}$$
$$\cong \forall_\varepsilon x \forall_\varepsilon y \left[x \mathrel{\dot{\geq}} y \land y \mathrel{\dot{\geq}} x \to x = y\right]$$

[78] Stack overflows can occur for larger axiom sets.

[79] This is due to memory limits in the SAT solvers.

[80] Again, we are using shorthand notation like $x \neq y$ for $\neg(x = y)$; see also Section 4.2.

$$(\texttt{REFL}_\sigma) \quad A \succeq A \text{ for all } A \in \mathcal{X}$$
$$\cong \forall_\sigma A\,[A \succeq A]$$

$$(\texttt{COMPL}_\sigma) \quad A \succeq B \vee A \preceq B \text{ for all } A \neq B \in \mathcal{X}$$
$$\cong \forall_\sigma A \forall_\sigma B\,[A \neq B \to (A \succeq B \vee A \preceq B)]$$

$$(\texttt{TRANS}_\sigma) \quad A \succeq B \wedge B \succeq C \Rightarrow A \succeq C \text{ for all } A, B, C \in \mathcal{X}$$
$$\cong \forall_\sigma A \forall_\sigma B \forall_\sigma C\,[(A \succeq B \wedge B \succeq C) \to A \succeq C]$$

Next we have the (very natural) axiom of *extension*, which was introduced and extensively discussed in Section 1.3 already:

$$(\texttt{EXT}) \quad x \mathrel{\dot{\geq}} y \iff \{x\} \succeq \{y\} \text{ for all } x, y \in X$$
$$\cong \forall_\varepsilon x \forall_\varepsilon y \left[x \mathrel{\dot{\geq}} y \leftrightarrow \{x\} \succeq \{y\}\right]$$

A further set of axioms we included in our search is the one dealing with the concept of *dominance*, i.e., the idea that adding an object to a set of prospects that are all dominated by (or dominating) the object produces a better (or worse) set, respectively. We chose for the well-known *Gärdenfors principle* (GF) as well as a weaker version by BARBERÀ [9] called *simple dominance* (SDom), which restricts (GF) to small sets:

$$(\texttt{GF1}) \quad ((\forall a \in A) x \mathrel{\dot{>}} a) \Rightarrow A \cup \{x\} \succ A \text{ for all } x \in X \text{ and } A \in \mathcal{X}$$
$$\cong \forall_\sigma A \forall_\varepsilon x \left[\exists_\varepsilon a (a \in A \wedge x \mathrel{\dot{\not>}} a) \vee A \cup \{x\} \succ A\right]$$

$$(\texttt{GF2}) \quad ((\forall a \in A) x \mathrel{\dot{<}} a) \Rightarrow A \cup \{x\} \prec A \text{ for all } x \in X \text{ and } A \in \mathcal{X}$$
$$\cong \forall_\sigma A \forall_\varepsilon x \left[\exists_\varepsilon a (a \in A \wedge x \mathrel{\dot{\not<}} a) \vee A \cup \{x\} \prec A\right]$$

$$(\texttt{SDom}) \quad x \mathrel{\dot{>}} y \Rightarrow (\{x\} \succ \{x, y\} \wedge \{x, y\} \succ \{y\}) \text{ for all } x, y \in X$$
$$\cong \forall_\varepsilon x \forall_\varepsilon y [x \mathrel{\dot{>}} y \to (\{x\} \succ \{x\} \cup \{y\} \wedge \{x\} \cup \{y\} \succ \{y\})]$$

Independence axioms are also commonly postulated and especially their weaker variants or versions thereof, like bottom, top, disjoint and intermediate independence, frequently play a role in characterization results (see, e.g., PATTANAIK and PELEG [51], and NITZAN and PATTANAIK [49]). We decided to include standard independence (as already introduced in Section 1.3), a stronger version (strictIND), which implies *strict* preferences, and a few weaker versions, viz. bottom (`botIND`), top (`topIND`), disjoint (`disIND`) and intermediate independence (`intIND`), which only apply to certain combinations of sets and elements. Attention, however, needs to be paid with respect to axiom names here. Sometimes the names "bottom independence" etc. are used for versions of strict independence or extended independence[81] instead of standard independence. We use the standard (weaker) versions unless indicated to the contrary.

$$(\texttt{IND}) \quad A \succ B \Rightarrow A \cup \{x\} \succeq B \cup \{x\} \text{ for all } A, B \in \mathcal{X} \text{ and } x \in X \setminus (A \cup B)$$
$$\cong \forall_\sigma A \forall_\sigma B \forall_\varepsilon x\,[(x \notin (A \cup B) \wedge A \succ B) \rightarrow A \cup \{x\} \succeq B \cup \{x\}]$$

$$(\texttt{strictIND}) \quad A \succ B \Rightarrow A \cup \{x\} \succ B \cup \{x\}$$
$$\text{for all } A, B \in \mathcal{X} \text{ and } x \in X \setminus (A \cup B)$$
$$\cong \forall_\sigma A \forall_\sigma B \forall_\varepsilon x\,[(x \notin (A \cup B) \wedge A \succ B) \rightarrow A \cup \{x\} \succ B \cup \{x\}]$$

$$(\texttt{botIND}) \quad A \succ B \Rightarrow A \cup \{x\} \succeq B \cup \{x\} \text{ for all } A, B \in \mathcal{X}$$
$$\text{and } x \in X \setminus (A \cup B) \text{ such that } y \dot{>} x \text{ for all } y \in A \cup B$$
$$\cong \forall_\sigma A \forall_\sigma B \forall_\varepsilon x\,[x \in A \cup B \vee \exists_\varepsilon y (y \in A \cup B \wedge y \dot{\not>} x) \vee$$
$$A \not\succ B \vee A \cup \{x\} \succeq B \cup \{x\}]$$

$$(\texttt{topIND}) \quad A \succ B \Rightarrow A \cup \{x\} \succeq B \cup \{x\} \text{ for all } A, B \in \mathcal{X}$$
$$\text{and } x \in X \setminus (A \cup B) \text{ such that } x \dot{>} y \text{ for all } y \in A \cup B$$
$$\cong \forall_\sigma A \forall_\sigma B \forall_\varepsilon x\,[x \in A \cup B \vee \exists_\varepsilon y (y \in A \cup B \wedge x \dot{\not>} y) \vee$$
$$A \not\succ B \vee A \cup \{x\} \succeq B \cup \{x\}]$$

$$(\texttt{disIND}) \quad A \succ B \Rightarrow A \cup \{x\} \succeq B \cup \{x\} \text{ for all } A, B \in \mathcal{X},$$
$$\text{such that } A \cap B = \emptyset, \text{ and for all } x \in X \setminus (A \cup B)$$
$$\cong \forall_\sigma A \forall_\sigma B \forall_\varepsilon x\,[(x \notin (A \cup B) \wedge \text{disjoint}(A, B) \wedge A \succ B) \rightarrow$$
$$A \cup \{x\} \succeq B \cup \{x\}]$$

[81] Extended independence does only require a weak preference in the antecedent:

$$A \succeq B \Rightarrow A \cup \{x\} \succeq B \cup \{x\} \text{ for all } A, B \in \mathcal{X} \text{ and } x \in X \setminus (A \cup B).$$

$$
\begin{aligned}
&(\texttt{intIND}) \quad A \succ B \Rightarrow A \cup \{x, y\} \succeq B \cup \{x, y\} \\
&\qquad \text{for all } A, B \in \mathcal{X} \text{ and } x, y \in X \setminus (A \cup B) \\
&\qquad \text{such that } x \mathrel{\dot{>}} z \text{ and } z \mathrel{\dot{>}} y \text{ for all } z \in A \cup B \\
&\cong \forall_\sigma A \forall_\sigma B \forall_\varepsilon x \forall_\varepsilon y \, [\exists_\varepsilon z (z \in A \cup B \wedge (x \mathrel{\dot{\not>}} z \vee z \mathrel{\dot{\not>}} y)) \vee \\
&\qquad x \in A \cup B \vee y \in A \cup B \vee A \not\succ B \vee \\
&\qquad A \cup \{x\} \cup \{y\} \succeq B \cup \{x\} \cup \{y\}]
\end{aligned}
$$

Bossert [15] introduced axioms describing the attitude of the decision maker towards uncertainty. We formalize weakenings of these axioms that apply to small sets only, since these turned out to suffice for characterization results like the ones by Arlegi [4].

The axiom of simple *uncertainty aversion* postulates that the decision maker will, for any alternative x, (strictly) prefer this alternative to a set containing both a better and a worse alternative. *Uncertainty appeal*, on the other hand, says that the ranking has to be just the other way around: the set with a better and a worse element is (strictly) preferred to the single element x.

These axioms play a crucial role in the characterization of the *min-max ordering*, which we will say more about in the beginning of Section 5.6.

$$
\begin{aligned}
&(\texttt{SUAv}) \quad (x \mathrel{\dot{>}} y \wedge y \mathrel{\dot{>}} z) \Rightarrow \{y\} \succ \{x, z\} \text{ for all } x, y, z \in X \\
&\cong \forall_\varepsilon x \forall_\varepsilon y \forall_\varepsilon z [(x \mathrel{\dot{>}} y \wedge y \mathrel{\dot{>}} z) \rightarrow \{y\} \succ \{x\} \cup \{z\}]
\end{aligned}
$$

$$
\begin{aligned}
&(\texttt{SUAp}) \quad (x \mathrel{\dot{>}} y \wedge y \mathrel{\dot{>}} z) \Rightarrow \{x, z\} \succ \{y\} \text{ for all } x, y, z \in X \\
&\cong \forall_\varepsilon x \forall_\varepsilon y \forall_\varepsilon z [(x \mathrel{\dot{>}} y \wedge y \mathrel{\dot{>}} z) \rightarrow \{x\} \cup \{z\} \succ \{y\}]
\end{aligned}
$$

In the very same characterization [4] that also employs Bossert's uncertainty principles, also monotonicity axioms called *simple top* and *bottom monotonicity*, respectively, are used.

The underlying idea of these axioms is simple: given two alternatives, it is better to get the better one of the two together with some third element (instead of the worse one with the same third element). The two respective versions of the axiom then only apply to alternatives that are ranked higher (top) than the third alternative, or ranked lower (bottom), respectively.

$$
\begin{aligned}
&(\texttt{STopMon}) \quad x \mathrel{\dot{>}} y \Rightarrow \{x, z\} \succ \{y, z\} \\
&\qquad \text{for all } x, y, z \in X \text{ such that } x \mathrel{\dot{>}} z \text{ and } y \mathrel{\dot{>}} z \\
&\cong \forall_\varepsilon x \forall_\varepsilon y \forall_\varepsilon z [(x \mathrel{\dot{>}} z \wedge y \mathrel{\dot{>}} z \wedge x \mathrel{\dot{>}} y) \rightarrow \{x\} \cup \{z\} \succ \{y\} \cup \{z\}]
\end{aligned}
$$

$$(\texttt{SBotMon}) \quad y \dot{>} z \Rightarrow \{x,y\} \succ \{x,z\}$$
$$\text{for all } x,y,z \in X \text{ such that } x \dot{>} y \text{ and } x \dot{>} z$$
$$\cong \forall_\varepsilon x \forall_\varepsilon y \forall_\varepsilon z[(x \dot{>} y \wedge x \dot{>} z \wedge y \dot{>} z) \rightarrow \{x\} \cup \{y\} \succ \{x\} \cup \{z\}]$$

An intuitively rather odd axiom is the principle of *even-numbered extension of equivalence*. It says that, for all sets with an even number of elements, if the decision maker is indifferent about whether this set is added to each of two distinct singleton sets, then she should also be indifferent about whether it is added to the union of the two singleton sets.

Even though it substantially lacks intuitive support, this axiom is useful because (together with a few other principles) it characterizes a median-based ordering proposed by NITZAN and PATTANAIK [49].

$$(\texttt{evenExt}) \quad (A \cup \{x\} \sim \{x\} \wedge A \cup \{y\} \sim \{y\}) \Rightarrow A \cup \{x,y\} \sim \{x,y\}$$
$$\text{for all } A \in \mathcal{X}, \text{ such that } |A| \text{ is even, and for all } x,y \in X \setminus A$$
$$\cong \forall_\sigma A \forall_\varepsilon x \forall_\varepsilon y[(\text{evencard}(A) \wedge x \notin A \wedge y \notin A) \rightarrow$$
$$((A \cup \{x\} \sim \{x\} \wedge A \cup \{y\} \sim \{y\}) \rightarrow$$
$$A \cup \{x\} \cup \{y\} \sim \{x\} \cup \{y\})]$$

The final axiom in our list is *monotone consistency* (MC), which was designed by ARLEGI [4] to characterize (in connection with other axioms) the *min-max ordering* (see also Section 5.6).

(MC) expresses that if a set of objects A is at least as good as another set B, then the union of the two is at least as good as the latter. This implies – and for complete binary relations is equivalent to – the potentially worse set B not being strictly better than the union of the two. Intuitively, it means that after adding the alternatives of the (weakly preferred) set A to the set B, the decision maker maintains the alternatives she had in B plus the ones that were contained in A, which was weakly preferred to B. Thus, this process should not produce a set that is strictly worse than B.

Although (MC) appears to be similar to the first axiom of the Gärdenfors principle, it is quite different in fact since it does not dictate the existence of any *strict* preferences.

$$(\texttt{MC}) \quad A \succeq B \Rightarrow A \cup B \succeq B \text{ for all } A,B \in \mathcal{X}$$
$$\cong \forall_\sigma A \forall_\sigma B\,[A \succeq B \rightarrow A \cup B \succeq B]$$

5.4 Impossibility Theorems Found

Our theorem search (checking problem instances up to a domain size of eight) yields a total of 84 *minimal* impossibility theorems on the space of the 20 above axioms. The results are minimal in two senses:

- the corresponding axiom set is minimal with respect to set inclusion, i.e., all proper subsets are compatible at the given domain size,
- the domain size is minimal, i.e., for all smaller domain sizes the given axiom set is still compatible.

Counting the total number of incompatible axiom sets (i.e., including all supersets), we find 312.432 inconsistent axiom sets out of about one million possible combinations.

The whole experiment required a running time of roughly one day[82] for handling all of the nearly 8,5 million instances. In order to externally verify as many impossibilities as possible, we used the solver *zChaff* [61] for all instances up to domain size 7, and switched to the faster solver *PrecoSAT* for instances with (exponentially larger) domain size 8. The five impossibilities occurring only on domain size 8 have therefore not been verified externally.[83]

In the following Table 5.1 we list all minimal impossibilities that our search method was able to find (and hence, assuming correctness of our method, all that there are) for domain sizes up to 8. Recall again that, by Corollary 4.16, these all directly correspond to full impossibility results (from the given domain size upwards).

The results are ordered ascendingly by minimal domain size, and descendingly by the number of axioms involved as a second criterion. The idea was to have stronger and easier to grasp impossibilities higher up in table, whereas more complicated results will come later. The numbers are simply an enumeration of the results and come with no further meaning.

No.	Size	LIN_ε	REFL_σ	COMPL_σ	TRANS_σ	EXT	SDom	GF1	GF2	IND	strictIND	SUAv	SUAp	STopMon	SBotMon	topIND	botIND	disIND	intIND	evenExt	MC
1	3	✓	·	·	·	·	✓	·	·	·	✓	·	·	·	·	·	·	·	·	·	·
2	3	✓	·	·	·	·	·	·	·	·	·	✓	✓	·	·	·	·	·	·	·	·
3	3	✓	·	·	·	·	·	✓	✓	·	✓	·	·	·	·	·	·	·	·	·	·
4	3	✓	·	✓	·	·	·	·	✓	·	✓	·	·	·	·	·	·	·	·	·	✓
5	3	✓	·	·	·	✓	·	·	✓	·	✓	·	·	·	·	·	·	·	·	·	✓
6	3	✓	·	·	·	·	·	·	✓	·	✓	✓	·	·	·	·	·	·	·	·	✓

Continued...

Table 5.1: Results of our automated and exhaustive theorem search on a space of 20 axioms (including orders).

[82] The experiment was performed on an Intel Xeon 2,26 GHz octo-core machine using only one core and 5GB of the available 24GB memory. The machine is part of the Dutch national compute cluster *Lisa* [58].

[83] Using only *zChaff* would have also been possible, but slower by a factor of about 10.

No.	Size	LIN$_\varepsilon$	REFL$_\sigma$	COMPL$_\sigma$	TRANS$_\sigma$	EXT	SDom	GF1	GF2	IND	strictIND	SUAv	SUAp	STopMon	SBotMon	topIND	botIND	disIND	intIND	evenExt	MC
7	3	✓	·	·	·	·	·	·	✓	·	✓	·	·	·	✓	·	·	·	·	·	✓
8	4	✓	·	·	·	·	·	·	✓	·	✓	·	·	✓	·	·	·	·	·	·	✓
9	4	✓	·	·	✓	·	·	✓	✓	✓	·	✓	·	·	·	·	·	·	·	·	·
10	4	✓	·	·	✓	·	✓	✓	·	✓	·	✓	·	·	·	·	·	·	·	·	·
11	4	✓	·	·	✓	✓	·	✓	·	·	✓	✓	·	·	·	·	·	·	·	·	·
12	4	✓	·	·	✓	·	·	✓	✓	✓	·	·	✓	·	·	·	·	·	·	·	·
13	4	✓	·	·	✓	·	✓	·	✓	✓	·	·	✓	·	·	·	·	·	·	·	·
14	4	✓	·	·	✓	✓	·	·	✓	·	✓	·	✓	·	·	·	·	·	·	·	·
15	4	✓	·	·	✓	·	·	✓	·	✓	·	✓	·	✓	·	·	·	·	·	·	·
16	4	✓	·	·	✓	·	✓	·	·	✓	·	✓	·	✓	·	·	·	·	·	·	·
17	4	✓	·	·	✓	·	·	✓	·	·	✓	✓	·	✓	·	·	·	·	·	·	·
18	4	✓	·	·	✓	·	·	·	✓	✓	·	·	✓	·	✓	·	·	·	·	·	·
19	4	✓	·	·	✓	·	✓	·	·	✓	·	·	✓	·	✓	·	·	·	·	·	·
20	4	✓	·	·	✓	·	·	·	✓	·	✓	·	✓	·	✓	·	·	·	·	·	·
21	4	✓	·	·	✓	✓	·	·	·	·	✓	✓	·	·	·	·	·	·	·	·	✓
22	4	✓	·	·	✓	·	·	·	✓	✓	·	·	✓	·	·	·	·	·	·	·	✓
23	4	✓	·	·	✓	·	·	·	✓	·	✓	·	✓	·	·	·	·	·	·	·	✓
24	4	✓	·	·	✓	·	·	·	·	✓	·	✓	·	✓	·	·	·	·	·	·	✓
25	4	✓	·	·	✓	·	·	·	·	·	✓	✓	·	✓	·	·	·	·	·	·	✓
26	4	✓	·	·	✓	·	·	·	·	·	·	✓	·	✓	·	·	✓	·	·	·	✓
27	4	✓	·	·	✓	·	·	·	·	·	·	✓	·	✓	·	·	·	✓	·	·	✓
28	4	✓	·	·	✓	·	·	✓	✓	·	·	✓	·	·	·	✓	✓	·	·	·	·
29	4	✓	·	·	✓	·	✓	✓	·	·	·	✓	·	·	·	✓	✓	·	·	·	·
30	4	✓	·	·	✓	·	·	✓	✓	·	·	·	✓	·	·	✓	✓	·	·	·	·
31	4	✓	·	·	✓	·	✓	·	✓	·	·	·	✓	·	·	✓	✓	·	·	·	·
32	4	✓	·	·	✓	·	·	✓	·	·	·	✓	·	✓	·	✓	✓	·	·	·	·
33	4	✓	·	·	✓	·	✓	·	·	·	·	✓	·	✓	·	✓	✓	·	·	·	·
34	4	✓	·	·	✓	·	·	·	✓	·	·	·	✓	·	✓	✓	✓	·	·	·	·
35	4	✓	·	·	✓	·	✓	·	·	·	·	·	✓	·	✓	✓	✓	·	·	·	·
36	4	✓	·	·	✓	·	·	·	✓	·	·	·	✓	·	✓	·	✓	✓	·	·	·
37	4	✓	·	·	✓	·	✓	·	·	·	·	·	✓	·	✓	·	✓	✓	·	·	·
38	4	✓	·	·	✓	·	·	✓	·	·	·	✓	·	✓	·	✓	·	✓	·	·	·
39	4	✓	·	·	✓	·	✓	·	·	·	·	✓	·	✓	·	✓	·	✓	·	·	·
40	4	✓	·	·	✓	·	·	✓	✓	·	·	✓	·	·	·	·	✓	·	·	·	✓
41	4	✓	·	·	✓	·	✓	✓	·	·	·	✓	·	·	·	·	✓	·	·	·	✓
42	4	✓	·	·	✓	·	·	·	✓	·	·	·	✓	·	·	✓	✓	·	·	·	✓
43	4	✓	·	·	✓	·	·	·	✓	·	·	·	✓	·	·	·	✓	✓	·	·	✓
44	5	✓	·	·	✓	·	·	✓	·	✓	·	✓	·	·	·	·	·	·	·	·	·

Continued...

Table 5.1: Results of our automated and exhaustive theorem search on a space of 20 axioms (including orders).

No.	Size	LIN_ε	REFL_σ	COMPL_σ	TRANS_σ	EXT	SDom	GF1	GF2	IND	strictIND	SUAv	SUAp	STopMon	SBotMon	topIND	botIND	disIND	intIND	evenExt	MC
45	5	✓	·	·	✓	·	·	✓	·	·	✓	✓	·	·	·	·	·	·	·	·	·
46	5	✓	·	·	✓	·	·	·	✓	✓	·	·	✓	·	·	·	·	·	·	·	·
47	5	✓	·	·	✓	·	·	·	✓	·	✓	·	✓	·	·	·	·	·	·	·	·
48	5	✓	·	·	✓	·	·	✓	·	·	·	✓	·	·	·	✓	✓	·	·	·	·
49	5	✓	·	·	✓	·	·	·	✓	·	·	·	✓	·	·	✓	✓	·	·	·	·
50	5	✓	·	·	✓	·	·	·	✓	·	·	·	✓	·	·	·	✓	✓	·	·	·
51	5	✓	·	·	✓	·	·	✓	·	·	·	✓	·	·	·	✓	·	✓	·	·	·
52	5	✓	·	✓	✓	·	·	·	·	·	✓	✓	·	·	·	·	·	·	·	·	✓
53	5	✓	·	·	✓	·	·	·	·	·	✓	✓	·	·	✓	·	·	·	·	·	✓
54	5	✓	·	·	✓	·	·	✓	·	·	·	✓	·	·	·	·	✓	·	·	·	✓
55	5	✓	·	·	✓	·	·	✓	·	·	·	✓	·	·	·	·	·	✓	·	·	✓
56	6	✓	·	·	✓	·	·	·	·	·	✓	✓	·	·	·	·	·	·	·	·	✓
57	6	✓	·	✓	✓	·	·	✓	✓	✓	·	·	·	·	·	·	·	·	·	·	·
58	6	✓	·	✓	✓	·	✓	·	✓	✓	·	·	·	✓	·	·	·	·	·	·	·
59	6	✓	·	✓	✓	·	✓	✓	·	✓	·	·	·	·	✓	·	·	·	·	·	·
60	6	✓	·	✓	✓	·	✓	·	·	✓	·	·	·	✓	✓	·	·	·	·	·	·
61	6	✓	·	✓	✓	·	·	✓	✓	·	·	·	·	·	·	✓	✓	·	·	·	·
62	6	✓	·	·	✓	·	·	✓	✓	·	·	✓	·	·	·	·	✓	·	✓	·	·
63	6	✓	·	·	✓	·	✓	✓	·	·	·	✓	·	·	·	·	✓	·	✓	·	·
64	6	✓	·	·	✓	·	·	✓	·	·	·	✓	·	✓	·	·	✓	·	✓	·	·
65	6	✓	·	·	✓	·	·	✓	✓	·	·	·	✓	·	·	✓	·	·	✓	·	·
66	6	✓	·	·	✓	·	✓	·	✓	·	·	·	✓	·	·	✓	·	·	✓	·	·
67	6	✓	·	·	✓	·	·	·	✓	·	·	·	✓	·	✓	✓	·	·	✓	·	·
68	6	✓	·	✓	✓	·	·	·	✓	✓	·	·	·	✓	·	·	·	·	·	·	✓
69	6	✓	·	·	✓	·	·	·	✓	·	·	·	✓	·	·	✓	·	·	✓	·	✓
70	6	✓	·	✓	✓	·	✓	·	✓	·	·	·	·	✓	·	✓	✓	·	·	·	·
71	6	✓	·	✓	✓	·	✓	✓	·	·	·	·	·	·	✓	✓	✓	·	·	·	·
72	6	✓	·	✓	✓	·	✓	·	·	·	·	·	·	✓	✓	✓	✓	·	·	·	·
73	6	✓	·	✓	✓	·	·	·	✓	·	·	·	·	✓	·	✓	✓	·	·	·	✓
74	6	✓	·	✓	✓	·	·	✓	✓	·	·	·	·	·	·	·	✓	✓	·	·	✓
75	6	✓	·	✓	✓	·	·	·	✓	·	·	·	·	✓	·	·	✓	✓	·	·	✓
76	6	✓	·	✓	✓	·	✓	✓	·	·	·	·	·	·	✓	·	✓	✓	·	·	✓
77	6	✓	·	✓	✓	·	✓	·	·	·	·	·	·	✓	✓	·	✓	✓	·	·	✓
78	7	✓	·	·	✓	·	·	✓	·	·	·	✓	·	·	·	·	✓	·	✓	·	·
79	7	✓	·	·	✓	·	·	·	✓	·	·	·	✓	·	·	✓	·	·	✓	·	·

Continued...

Table 5.1: Results of our automated and exhaustive theorem search on a space of 20 axioms (including orders).

No.	Size	LIN_ε	REFL_σ	COMPL_σ	TRANS_σ	EXT	SDom	GF1	GF2	IND	strictIND	SUAv	SUAp	STopMon	SBotMon	topIND	botIND	disIND	intIND	evenExt	MC
80	8	✓	·	✓	✓	·	·	✓	✓	·	·	·	·	·	·	·	✓	✓	✓	·	·
81	8	✓	·	✓	✓	·	·	✓	✓	·	·	·	·	·	·	✓	·	✓	✓	·	·
82	8	✓	·	✓	✓	·	✓	✓	·	·	·	·	·	·	✓	·	✓	✓	✓	·	·
83	8	✓	·	✓	✓	·	✓	·	✓	·	·	·	·	✓	·	✓	·	✓	✓	·	·
84	8	✓	·	✓	✓	·	·	·	✓	·	·	·	·	✓	·	✓	·	✓	✓	·	✓

Table 5.1: Results of our automated and exhaustive theorem search on a space of 20 axioms (including orders).

It would now, of course, be helpful to find some structure in this large table that makes it easier to explore the results.[84] Unfortunately, this is rather difficult and the best we can do is give some general remarks and observations that will help the reader process the results.

We first note that impossibilities can occur from all of the tested domain sizes larger than 2 onwards. This already is something new since so far only impossibilities with $|X| \geq k$, $k \in \{3, 4, 6\}$ were known.

The results themselves differ much in their level of appeal and interestingness. We can find impossibilities of at least the five (potentially overlapping) categories

(a) known results,
(b) variations of known results,
(c) direct consequences of other results,
(d) straightforward results, and most importantly
(e) new results, so far unknown.

Some previously known results we can easily recognize among the ones in our list: the Kannai-Peleg Theorem corresponds to our Impossibility No. 57, an impossibility theorem by Barberà and Pattanaik [12] can be found as Impossibility No. 1. Other than that we are only aware of one more known impossibility under the interpretation of *complete uncertainty*, which we could unfortunately not encode in our framework since it uses the axiom of *neutrality* (see Section 4.2). It is a variant of the Kannai-Peleg Theorem presented by Barberà, Bossert and Pattanaik [11], in which the number of elements has been lowered to four by adding the aforementioned axiom.

Variations of known results are also easy to spot by just keeping some axioms fixed and browsing for results involving these. Impossibilities No. 80 and No. 10, for instance, are variations of the Kannai-Peleg Theorem, where in the former a weakening of the axioms makes the impossibility occur only

[84] On a computer, however, an exploration is quite easy using for instance the filtering tool within a spreadsheet program like Microsoft Excel. One can then switch axioms on and off and inspect the corresponding impossibilities.

at a larger domain size. The latter is a variation in the other direction: the additional axiom (SUAv) causes an impossibility at a domain size of 4 elements already. And many more such variations of known theorems can be found.[85]

As we used a set of axioms in which certain axioms imply others, we had to expect results that are just direct consequences of others. In particular, every result involving some (weak) form of independence will also occur with the standard or strict independence only, and similarly for simple dominance, which is a weaker form of the Gärdenfors principle. Examples of such results are the Impossibilities No. 3 (implied by No. 1) and Impossibility No. 9 (implied by No. 28).

Straightforward results we could only find one: Impossibility No. 2 says that a binary relation cannot fulfill both (SUAv) and (SUAp), i.e., reflect the contradictory principles of uncertainty aversion and uncertainty appeal. This is (especially when examining the exact statement of the axioms) immediate.

What we are left with are the new, previously unknown results. There are quite a few of them, but they differ in how interesting they are. For instance, it is not very reasonable to only postulate (GF1) but not (GF2), which makes the new Impossibility No. 11 not so fascinating after all. But we can also find results like Impossibility No. 52 or No. 56, where the combination of axioms appears to be reasonable and yet leads to an impossibility.

We will return to some of these results in Section 5.6, where we prove a few of them by hand. But let us now for a moment shift our perspective from problems, i.e., combinations of axioms, to the special role of individual axioms with respect to *all* results. On the one hand, the axiom ($\texttt{LIN}_\varepsilon$) of a linear order $\dot{\geq}$ on X occurs in all impossibilities. This means that there is no impossibility without this axiom (on the given axiom space and up to domain size 8). This could have been anticipated: if we use the empty relation on X for $\dot{\geq}$, then most axioms do not say anything about $\succeq$ anymore and can hence not be incompatible.

Also note that the only impossibility without any form of independence is the (straightforward) Impossibility No. 2.

On the other hand, the axioms (evenExt) and ($\texttt{REFL}_\sigma$) do not occur in any impossibility. Therefore, we can conclude that these must be particularly well-compatible with the other axioms. Or put differently, that adding them to a given set of axioms does not cause an impossibility.[86]

The axiom (intIND) of intermediate independence is contained in all discovered impossibilities at domain sizes 7 and 8 (and does not cause any impossibilities at sizes 5 and below). That this axiom is involved in somewhat larger instances makes a great deal of sense intuitively: for each application of the axiom we have to add *two* elements (one above, one below the set we apply

[85] See, e.g., Impossibilities No. 33, 37, 40, etc.

[86] Even though reflexivity ($\texttt{REFL}_\sigma$) will not be needed for any impossibility, we will usually speak of a weak order as soon as completeness ($\texttt{COMPL}_\sigma$) and transitivity ($\texttt{TRANS}_\sigma$) are required.

it to), and so it was to be expected that larger domain sizes are necessary for a contradiction.

5.5 Potential Possibilities Found

Yet another application of our theorem search, which was originally designed to find *impossibilities*, could be the discovery of *possibility* or characterization results. For such theorems, however, we cannot expect the search to return the full result as we lack an (inductive) step going upwards for possibilities.[87] But still we can consider the results of our theorem search a heuristic for possibilities, too: if an axiom set is compatible for large problem domains, then this might be a good indicator for a general possibility.[88]

We have listed a few such *potential possibilities* in Table 5.2. All of these are maximal with respect to set inclusion, i.e., there are no proper axiom supersets of the respective axiom sets that are still consistent at the given domain size.

No.	Size	LIN_ε	REFL_σ	COMPL_σ	TRANS_σ	EXT	SDom	GF1	GF2	IND	strictIND	SUAv	SUAp	STopMon	SBotMon	topIND	botIND	disIND	intIND	evenExt	MC
P1	8	✓	✓	✓	·	✓	✓	✓	✓	✓	·	✓	·	✓	✓	✓	✓	✓	✓	✓	✓
P2	8	✓	✓	✓	✓	✓	·	✓	·	✓	✓	·	✓	✓	✓	✓	✓	✓	✓	✓	✓
P3	8	✓	✓	✓	✓	✓	·	·	✓	✓	✓	✓	·	✓	✓	✓	✓	✓	✓	✓	·
P4	8	✓	✓	·	✓	✓	✓	✓	✓	✓	·	·	·	✓	✓	✓	✓	✓	✓	✓	✓
P5	8	✓	✓	✓	✓	✓	✓	✓	✓	·	·	·	✓	✓	✓	·	✓	·	✓	✓	✓
P6	8	✓	✓	✓	✓	✓	✓	✓	✓	·	·	✓	·	✓	✓	✓	·	·	✓	✓	✓

Table 5.2: A small selection of potential possibilities returned by our automated and exhaustive theorem search on a space of 20 axioms (including orders).

[87] At a general level, such a step even cannot exist because contrapositively it would cause impossibilities to propagate downwards, i.e., from larger to smaller domain sizes. We know that this is not the case since we can, for instance, construct weak orders on small domains ($|X| \leq 5$) satisfying the axioms of the Kannai-Peleg Theorem (see Appendix B.2).

[88] Especially leaving out the axiom of intermediate independence (`intIND`) can support this claim: without (`intIND`) the largest domain size necessary for any of the above impossibilities is $|X| = 6$ and no further impossibilities occur with 7 or 8 elements, respectively.

Potential Possibility P2, for instance, shows that the dominance axioms are indeed somewhat problematic: if we drop them,[89] then (apart from the clear incompatibility of (`SUAv`) and (`SUAp`)) all other axioms are consistent up to a domain size of eight elements. The somewhat dual result P3 surprisingly forces (`STopMon`) out, which shows that the axioms might be less symmetric than intuitively expected.

As an alternative to discarding all dominance principles for producing potential possibilities, it also suffices to just give up simple uncertainty aversion (`SUAv`) and top (`topIND`) as well as disjoint independence (`disIND`) (and therefore also the stronger (`IND`) and (`strictIND`)) to obtain a potential possibility (P5). Potential possibility P6, on the other hand, is the dual result for (`SUAp`) and (`botIND`) instead of (`SUAv`) and (`topIND`), respectively.

But also loosening the restrictions of the order $\succeq$ can be a potential way out of the impossibilities. Giving up transitivity, for instance, enables us to keep all dominance principles and standard independence, if we only drop strict independence (`strictIND`) and simple uncertainty appeal (`SUAp`) (see Potential Possibility P1). Alternatively, we can choose to not postulate completeness ($\texttt{COMPL}_\sigma$) and obtain a similar, but slightly weaker combination of axioms (P4).

For finding the full possibility and characterization results, remember also that (in addition to the pure fact that these axiom sets are still consistent at the given domain size) our program can output examples of weak orders satisfying a given set of axioms. One can then look at these orders manually and try to find common characteristics that might help in finding out which (class of) orders, if any, satisfy the axioms or are even characterized by them. But since, in order to establish a full possibility, one then still has to show that each of the axioms is satisfied by the predicted order for arbitrary domain sizes, this remains a laborious task.

5.6 Some Manual Proofs of Impossibilities

In the following we want to discuss some of the obtained impossibilities and also provide manual proofs for the theorems. The latter should underpin the usefulness of our theorem search as a heuristic, even in case one does not consider the output of such a program a rigorous proof. The knowledge about which axioms to use and what the critical domain size is, can guide the construction of manual proofs significantly. Additionally, one can run the program again with slightly modified axioms, not only to get an even better understanding about where the borderline is between the possible and the impossible, but also to have some assistance in choosing the right steps when proving the results by hand. And there is even one more application of the program during the search for a manual proof: one can run the program with single axioms

[89] We can even keep (`GF1`).

left out and look at the created examples of orders satisfying the remaining axioms in order to understand which structural properties these imply that are then incompatible with the left-out axiom.

5.6.1 Non-intuitive Impossibility

Let us start with the concrete example of our most striking result. In the year 2000 Bossert, Pattanaik and Xu [16] published a result stating that the axioms (SDom), (IND), (SUAv) and (STopMon) characterize the so-called *min-max ordering*, which is defined by

$$A \succeq_{mnx} B \iff [\min(A) \dot{>} \min(B) \vee (\min(A) = \min(B) \wedge \max(A) \dot{\geq} \max(B))].$$

Their paper also contained a dual result for the *max-min ordering* (characterized by the axioms (SDom), (IND), (SUAp), (SBotMon)).

The reader can now check that this contradicts the results of our theorem search since both of these axiom sets are among the impossibility theorems in Table 5.1 (Impossibilities No. 16 and 19). Indeed, it turns out that the proofs of Bossert, Pattanaik and Xu were flawed as Arlegi [4] pointed out three years later.[90] Arlegi, however, only notes that the *min-max* and *max-min orderings* do not satisfy the axiom of independence (IND), i.e., that these orders cannot be characterized by the axioms (SDom), (IND), (SUAv), (STopMon), and (SDom), (IND), (SUAp), (SBotMon), respectively. Thus, this does not only show the non-intuitiveness of our findings (as the contrary was believed for some time), but it also yields more than just a counterexample to the original publication: we additionally get that the four axioms under consideration are even inconsistent (in the presence of transitivity) and hence *no* transitive binary relation whatsoever can satisfy them. We give a manual proof for this result.

Theorem 5.1 (Impossibility No. 16). *Let X be a linearly ordered set with $|X| \geq 4$. Then there exists no transitive binary relation $\succeq$ on $\mathcal{X}$ satisfying simple dominance* (SDom)*, independence* (IND)*, simple uncertainty aversion* (SUAv)*, and simple top monotonicity* (STopMon)*.*

Proof. Let x_i, $i \in \{1, 2, 3, 4\}$ denote four distinct elements of X such that they are ordered descendingly by $\dot{>}$ with respect to their index, i.e., $x_1 \dot{>} x_2 \dot{>} x_3 \dot{>} x_4$. By way of contradiction, suppose there exists a transitive binary relation $\succeq$ on $\mathcal{X}$ satisfying simple dominance (SDom), independence (IND), simple uncertainty aversion (SUAv), and simple top monotonicity (STopMon).

On the one hand, it follows from simple uncertainty aversion applied to $x_1 \dot{>} x_2 \dot{>} x_3$ that $\{x_2\} \succ \{x_1, x_3\}$, and adding x_4 to both sets yields (by independence):

[90] He also presents two new axiom sets to characterize the two orders and proves the characterizations.

$$\{x_2, x_4\} \succeq \{x_1, x_3, x_4\}. \tag{5.1}$$

On the other hand, we can use simple dominance (applied to $x_3 \dot{>} x_4$) to show $\{x_3, x_4\} \succ \{x_4\}$, from which

$$\{x_1, x_3, x_4\} \succeq \{x_1, x_4\} \tag{5.2}$$

follows by independence. Furthermore, simple top monotonicity applied to $x_1 \dot{>} x_2 \dot{>} x_4$ directly gives

$$\{x_1, x_4\} \succ \{x_2, x_4\}, \tag{5.3}$$

which we are able to combine with (5.2) by transitivity.[91] We thus obtain

$$\{x_1, x_3, x_4\} \succ \{x_2, x_4\},$$

which directly contradicts (5.1).

Thus, there cannot be any transitive binary relation $\succeq$ on $\mathcal{X}$ satisfying the principles of simple dominance (`SDom`), independence (`IND`), simple uncertainty aversion (`SUAv`), and simple top monotonicity (`STopMon`). $\square$

Note, that all four axioms are not only actually used in the proof above, but they are necessary for the result and also logically independent from each other as the following examples of weak orders show:[92]
Let $X = \{x_1, x_2, x_3, x_4\}$ and $x_1 \dot{>} x_2 \dot{>} x_3 \dot{>} x_4$.

(a) The weak order $\succeq$ given by
$\{x_1\} \succ \{x_2\} \succ \{x_3\} \succ \{x_4\} \succ \{x_1, x_2\} \succ \{x_1, x_3\} \succ \{x_2, x_3\} \succ \{x_1, x_4\} \succ \{x_2, x_4\} \succ \{x_3, x_4\} \succ \{x_1, x_2, x_3\} \succ \{x_1, x_2, x_4\} \succ \{x_1, x_3, x_4\} \succ \{x_2, x_3, x_4\} \succ \{x_1, x_2, x_3, x_4\}$
satisfies (`IND`), (`SUAv`), (`STopMon`), but not (`SDom`).

(b) The weak order $\succeq$ given by
$\{x_1\} \succ \{x_1, x_2\} \succ \{x_2\} \succ \{x_1, x_3\} \succ \{x_2, x_3\} \succ \{x_3\} \succ \{x_1, x_2, x_3\} \succ \{x_1, x_4\} \succ \{x_2, x_4\} \succ \{x_1, x_2, x_4\} \succ \{x_3, x_4\} \succ \{x_4\} \succ \{x_1, x_3, x_4\} \succ \{x_2, x_3, x_4\} \succ \{x_1, x_2, x_3, x_4\}$
satisfies (`SDom`), (`SUAv`), (`STopMon`), but not (`IND`).

(c) The weak order $\succeq$ given by
$\{x_1\} \succ \{x_1, x_2\} \succ \{x_1, x_3\} \sim \{x_1, x_2, x_3\} \succ \{x_2\} \succ \{x_2, x_3\} \succ \{x_3\} \succ \{x_1, x_4\} \sim \{x_1, x_2, x_4\} \sim \{x_1, x_3, x_4\} \sim \{x_1, x_2, x_3, x_4\} \succ \{x_2, x_4\} \sim \{x_2, x_3, x_4\} \succ \{x_3, x_4\} \succ \{x_4\}$
satisfies (`SDom`), (`IND`), (`STopMon`), but not (`SUAv`).

(d) The weak order $\succeq$ given by
$\{x_1\} \succ \{x_1, x_2\} \succ \{x_2\} \succ \{x_1, x_3\} \sim \{x_1, x_2, x_3\} \succ \{x_2, x_3\} \succ \{x_3\} \succ \{x_1, x_4\} \sim \{x_1, x_3, x_4\} \sim \{x_2, x_4\} \sim \{x_1, x_2, x_4\} \sim \{x_2, x_3, x_4\} \sim \{x_1, x_2, x_3, x_4\} \succ \{x_3, x_4\} \succ \{x_4\}$
satisfies (`SDom`), (`IND`), (`SUAv`), but not (`STopMon`).

[91] See Remark 1.1 for (a proof of) this kind of transitivity.

[92] All of these examples were constructed fully automatically using our program as described in Section 3.4.1.

It can now also be seen that no subset of these four axioms suffices to characterize the *min-max ordering*. Any subset containing (`IND`) can be rejected immediately since (`IND`) is violated by the *min-max ordering* (as we have noted earlier), and any subset not containing it cannot suffice for a characterization either, since example (b) differs from the *min-max ordering* $\succeq_{mnx}$ (in which $\{x_2, x_3, x_4\} \prec_{mnx} \{x_1, x_2, x_3, x_4\}$).

5.6.2 Variations of the Kannai-Peleg Theorem

A different theorem we would like to prove manually is Impossibility No. 9, a variation of the Kannai-Peleg Theorem that trades an additional axioms (simple uncertainty aversion) for the impossibility occurring at a domain size of 4 rather than 6 elements.

The theorems we are presenting in the following are very instructive examples since they do not only shed some more light on different variations of the Kannai-Peleg Theorem, but will also show us how versions of a result can help us strengthen its proof.

Theorem 5.2 (Impossibility No. 9). *Let X be a linearly ordered set with $|X| \geq 4$. Then there exists no transitive binary relation $\succeq$ on $\mathcal{X}$ satisfying the Gärdenfors principle* (`GF`)*, independence* (`IND`) *and simple uncertainty aversion* (`SUAv`)*.*

Proof. Let x_i, $i \in \{1, 2, \ldots, 4\}$ denote four distinct elements of X such that they are ordered descendingly by $\dot{>}$ with respect to their index, i.e., $x_1 \dot{>} x_2 \dot{>} x_3 \dot{>} x_4$. By way of contradiction, suppose there exists a transitive binary relation $\succeq$ on $\mathcal{X}$ satisfying the Gärdenfors principle (`GF`), independence (`IND`) and simple uncertainty aversion (`SUAv`). Simple uncertainty aversion (`SUAv`) dictates that $\{x_2\} \succ \{x_1, x_3\}$, from which we can infer

$$\{x_2, x_4\} \succeq \{x_1, x_3, x_4\} \tag{5.4}$$

by independence (`IND`). Now it is just a matter of using Lemma 1.4 for both sets in order to obtain the following chain:

$$\{x_2, x_3, x_4\} \sim \{x_2, x_4\} \succeq \{x_1, x_3, x_4\} \sim \{x_1, x_2, x_3, x_4\}.$$

This, then, stands in direct contradiction to the Gärdenfors principle (`GF`), which demands that $\{x_2, x_3, x_4\} \prec \{x_1, x_2, x_3, x_4\}$. □

Remark 5.3. Instead of using Lemma 1.4, we can also directly apply the given axioms.

On the one hand, $\{x_1\} \succ \{x_1, x_2\}$ holds by (`GF2`), and (`IND`) yields $\{x_1, x_3\} \succeq \{x_1, x_2, x_3\}$, from which we can obtain $\{x_1\} \succ \{x_1, x_2, x_3\}$ by $\{x_1\} \succ \{x_1, x_3\}$ ((`GF1`)) and transitivity. Then again applying (`IND`) gives $\{x_1, x_4\} \succeq \{x_1, x_2, x_3, x_4\}$. The missing link (in order to get $\{x_1, x_3, x_4\} \succeq$

$\{x_1, x_2, x_3, x_4\}$ by transitivity) $\{x_1, x_3, x_4\} \succeq \{x_1, x_4\}$ can be obtained from $\{x_3, x_4\} \succ \{x_4\}$ ((GF1)) by (IND).

On the other hand, $\{x_3, x_4\} \succ \{x_4\}$ holds by (GF1) as observed before and, thus, (IND) gives $\{x_2, x_3, x_4\} \succeq \{x_2, x_4\}$.

Together with (5.4) we can piece things together by transitivity and obtain

$$\{x_2, x_3, x_4\} \succeq \{x_2, x_4\} \overset{(5.4)}{\succeq} \{x_1, x_3, x_4\} \succeq \{x_1, x_2, x_3, x_4\},$$

which gives the desired contradiction with (GF1).

Note that the same impossibility result also holds with simple uncertainty appeal (SUAp) (Impossibility No. 12) in place of simple uncertainty aversion (SUAv) and the proof works analogously (starting from $\{x_3\} \prec \{x_2, x_4\}$ and then adding x_1 by independence). Also one should remark here that neither reflexivity nor completeness of $\succeq$ are used in the proof above (as already indicated by our results, see Table 5.1). Thus, the impossibility holds for arbitrary transitive binary relations instead of just weak orders.

When we have an even closer look at Table 5.1, which was obtained using our program, then we can see that there is an even stronger form of Theorem 5.2: Impossibility No. 28 corresponds to the the axioms (GF), (SUAv), (botIND), (topIND) and is impossible from domain size 4 on. In contrast to (IND), the axioms (botIND) and (topIND) allow the principle of independence in certain situations only: the element to be added has to be ranked below or above all the elements in both sets, respectively. Actually, one can see at closer inspection of the proof given above for Theorem 5.2 (and Remark 5.3 in particular) that only these weaker forms of (IND) are used.[93] Therefore, we have already proved the following stronger version of Theorem 5.2.

Theorem 5.4 (Impossibility No. 28). *Let X be a linearly ordered set with $|X| \geq 4$. Then there exists no transitive binary relation $\succeq$ on $\mathcal{X}$ satisfying the Gärdenfors principle* (GF)*, bottom* (botIND) *as well as top independence* (topIND)*, and simple uncertainty aversion* (SUAv)*.*

An interesting insight can be obtained from comparing Impossibility No. 48 to the previous result. It shows us that we can drop the second Gärdenfors axiom if we add just one element to the domain, i.e., we have $|X| \geq 5$. The exact result as well as a manual proof is shown in the following.

Theorem 5.5 (Impossibility No. 48). *Let X be a linearly ordered set with $|X| \geq 5$. Then there exists no transitive binary relation $\succeq$ on $\mathcal{X}$ satisfying the first axiom of the Gärdenfors principle* (GF1)*, bottom* (botIND) *as well as top independence* (topIND)*, and simple uncertainty aversion* (SUAv)*.*

[93] The same is true for the Kannai-Peleg Theorem and its original proof [37]. Confer also Impossibility No. 61.

Proof. Let x_i, $i \in \{1, 2, \ldots, 5\}$ denote five distinct elements of X such that they are ordered descendingly by $\dot{>}$ with respect to their index, i.e., $x_1 \dot{>} x_2 \dot{>} x_3 \dot{>} x_4 \dot{>} x_5$. By way of contradiction, suppose there exists a transitive binary relation $\succeq$ on $\mathcal{X}$ satisfying the first axiom of the Gärdenfors principle (GF1), bottom (botIND) as well as top independence (topIND), and simple uncertainty aversion (SUAv).

Simple uncertainty aversion tells us that $\{x_2\} \succ \{x_1, x_3\}$ and so

$$\{x_2, x_5\} \succeq \{x_1, x_3, x_5\} \tag{5.5}$$

follows by bottom independence (botIND). In the exact same way, we can obtain

$$\{x_3, x_5\} \succeq \{x_2, x_4, x_5\} \tag{5.6}$$

by (botIND) from $\{x_3\} \succ \{x_2, x_4\}$ ((SUAv)), and analogously we get

$$\{x_2, x_4, x_5\} \succeq \{x_2, x_5\} \tag{5.7}$$

by top independence (topIND) from $\{x_4, x_5\} \succeq \{x_5\}$, which is the case by (GF1). We can now put these three statements together:

$$\{x_3, x_5\} \overset{(5.6)}{\succeq} \{x_2, x_4, x_5\} \overset{(5.7)}{\succeq} \{x_2, x_5\} \overset{(5.5)}{\succeq} \{x_1, x_3, x_5\}, \tag{5.8}$$

which implies $\{x_3, x_5\} \succeq \{x_1, x_3, x_5\}$ by transitivity and hence contradicts the axiom (GF1). □

Alternatively, we could have replaced (botIND) by (disIND) (No. 51), or (topIND) by (intIND), then, however, requiring at least seven elements in the domain (No. 78).

Knowing which axioms exactly cause the impossibility turned out to be extremely helpful in the process of constructing and improving the proof of Theorem 5.5. Not only did we know what forms of independence we had to use, but also did an early version of the proof still contain applications of (disIND), (botIND) and (topIND), which showed us that a better proof must still be possible.

Two further variants of the Kannai-Peleg Theorem can be found in Impossibilities No. 80 and 81, which can be considered strengthenings of the original Kannai-Peleg Theorem as they contain weaker versions of independence only. The strengthening, however, comes at the cost of the impossibility starting from a domain size of eight elements instead of six. The form of independence that remains is a combination of intermediate, disjoint, and bottom or top independence, respectively, which are (even together) strictly weaker than standard independence.

5.6.3 Impossibilities without Dominance

All existing impossibilities in the literature we are aware of involve the Gärdenfors principle or at least simple dominance. So let us now consider what kinds of results we can obtain without any dominance principle.

A striking impossibility without any principle of dominance – i.e., without (`GF`) or (`SDom`) – is Impossibility No. 52: the axioms strict independence (`strictIND`), simple uncertainty aversion (`SUAv`) and monotone consistency (`MC`) are incompatible in presence of completeness and transitivity from domain size 5 on.[94] One might be tempted to think that this impossibility is mostly due to problems between (`SUAv`) and (`MC`) since they seem to express contrary ideas: whereas (`SUAv`) favours small sets over large ones, (`MC`) tells us that unions of two sets should be preferred to at least one of the sets. But actually there is even a characterization result of the *min-max ordering* by Arlegi [4] involving both axioms (`SUAv`) and (`MC`), and thus showing us that this sensible and intuitive ordering fulfills these two axioms. Therefore, we can see that it should not at all be considered unreasonable to have both axioms act together.

The proof of Impossibility No. 52 needs to make use of completeness (and transitivity) of $\succeq$, which also makes it different from the previous proofs in this chapter.

Theorem 5.6 (Impossibility No. 52). *Let X be a linearly ordered set with $|X| \geq 5$. Then there exists no weak order $\succeq$ on $\mathcal{X}$ satisfying strict independence* (`strictIND`)*, simple uncertainty aversion* (`SUAv`) *and monotone consistency* (`MC`)*.*

Proof. Let x_i, $i \in \{1, 2, \ldots, 5\}$ denote five distinct elements of X such that they are ordered descendingly by $\dot{>}$ with respect to their index, i.e., $x_1 \dot{>} x_2 \dot{>} x_3 \dot{>} x_4 \dot{>} x_5$. By way of contradiction, suppose there exists a weak order $\succeq$ on $\mathcal{X}$ satisfying (`strictIND`), (`SUAv`) and (`MC`). We will proceed by assuming a weak preference between the sets $\{x_1, x_4\}$ and $\{x_2, x_5\}$ and show that either way this leads to a contradiction, hence violating the completeness of $\succeq$.
Case 1: Suppose $\{x_2, x_5\} \succeq \{x_1, x_4\}$. Then (`MC`) yields

$$\{x_1, x_2, x_4, x_5\} = \{x_2, x_5\} \cup \{x_1, x_4\} \succeq \{x_1, x_4\}, \tag{5.9}$$

which we want to contradict. This can be done starting from

$$\{x_4\} \succ \{x_3, x_5\} \text{ and } \{x_3\} \succ \{x_2, x_4\},$$

which are both direct applications of (`SUAv`). The latter can be extended to match the righthand side of the former by adding x_5 using (`strictIND`), which then produces $\{x_3, x_5\} \succ \{x_2, x_4, x_5\}$. Hence, transitivity[95] yields

$$\{x_4\} \succ \{x_2, x_4, x_5\},$$

to which we can add the further element x_1 by strict independence. This leaves us with the desired $\{x_1, x_4\} \succ \{x_1, x_2, x_4, x_5\}$ contradicting (5.9).
Case 2: The case of assuming $\{x_1, x_4\} \succeq \{x_2, x_5\}$ is analogous. We obtain

[94] If we want to drop completeness of the relation $\succeq$, the impossibility only occurs for domains with at least 6 elements (see Impossibility No. 56).
[95] Like before, cf. Remark 1.1 for (a proof of) transitivity of the strict relation.

$$\{x_1, x_2, x_4, x_5\} = \{x_1, x_4\} \cup \{x_2, x_5\} \succeq \{x_2, x_5\} \tag{5.10}$$

by (MC), and can again start from (SUAv), but with (partially) different elements this time:

$$\{x_2\} \succ \{x_1, x_3\} \text{ and } \{x_3\} \succ \{x_2, x_4\}.$$

We add x_1 to the latter by (strictIND) in order to apply transitivity:

$$\{x_2\} \succ \{x_1, x_3\} \succ \{x_1, x_2, x_4\}.$$

To this we then just add x_5 and get the desired contradiction of $\{x_2, x_5\} \succ \{x_1, x_2, x_4, x_5\}$ and (5.10). □

Note that we have found quite a few variants of this impossibility. According to our results, completeness could be replaced by simple bottom monotonicity (Impossibility No. 53) or even be dropped at the price of having one more element in the domain (Impossibility No. 56). Alternatively, one can weaken strict independence to either bottom or disjoint independence at the price of adding the axiom of simple top monotonicity (Impossibilities No. 26 and 27, respectively).[96] A seemingly further variant can be obtained from trading the axiom of extension (EXT) for a smaller domain. It is, however, a direct consequence of Impossibilities No. 26 and 27, respectively, since (EXT) and (strictIND) together imply (STopMon).

Since strict independence can be considered a relatively strong axiom and is therefore criticizable as a natural axiom, especially the variants in No. 26 and 27 are worth considering again, as they do only postulate a very weak form of independence. We have the following formal statement and its proof:

Theorem 5.7 (Impossibility No. 26). *Let X be a linearly ordered set with $|X| \geq 4$. Then there exists no transitive binary relation $\succeq$ on $\mathcal{X}$ satisfying bottom independence* (botIND)*, simple uncertainty aversion* (SUAv)*, simple top monotonicity* (STopMon) *and monotone consistency* (MC)*.*

Proof. Let x_i, $i \in \{1, 2, \ldots, 4\}$ denote four distinct elements of X such that they are ordered descendingly by $\dot{>}$ with respect to their index, i.e., $x_1 \dot{>} x_2 \dot{>} x_3 \dot{>} x_4$. By way of contradiction, suppose there exists a transitive binary relation $\succeq$ on $\mathcal{X}$ satisfying the axioms (botIND), (SUAv), (STopMon), and (MC).

Simple uncertainty aversion tells us that $\{x_2\} \succ \{x_1, x_3\}$ and so

$$\{x_2, x_4\} \succeq \{x_1, x_3, x_4\} \tag{5.11}$$

follows by bottom independence (botIND).[97]. Similarly, we have $\{x_3\} \succ \{x_1, x_4\}$ by (SUAv), and can use this to obtain

[96] These impossibilities even need one element less in the domain X.

[97] The same statement follows from (disIND), which could hence have been used instead (making this a proof for Impossibility No. 27).

$$\{x_1, x_3, x_4\} \succeq \{x_1, x_4\} \tag{5.12}$$

by (`MC`). We can now put the two statements (5.11) and (5.12) together and transitivity yields

$$\{x_2, x_4\} \succeq \{x_1, x_4\},$$

which directly contradicts (`STopMon`). □

This result comes as quite a surprise since Arlegi [4] characterizes the *min-max ordering* by an axiom set including (`SUAv`), (`STopMon`), and (`MC`) (as well as two further axioms). It follows that adding just a tiny bit of independence to these three axioms turns their possibility into a general impossibility.

5.7 Summary

In this chapter we presented our automated, exhaustive theorem search and its results on a space of 20 axioms from the literature.

The search method and its implementation were described and all involved axioms explained. In Table 5.1 we presented an overview of all 84 minimal impossibility results that had been found by our algorithm. We indicated how these can be classified from "known" and "variations of known" theorems to completely "new, so far unknown" results. Additionally, we exhibited a few satisfiable instances since these can potentially cater as heuristics for general possibilities or characterizations, i.e., compatible combinations of axioms.

Impossibility theorems of very different importance were discovered and some instructive examples were selected for manual proofs, which we showed and described in Section 5.6. The highlight was a highly unintuitive impossibility manifested in a peer-reviewed publication of a false possibility result involving exactly the axioms that were now found to be generally incompatible. But also many variations and partial strengthenings of the Kannai-Peleg Theorem were obtained as well as two new impossibilities not containing any form of dominance.

Conclusion and Future Work

Summary and Conclusion

This thesis has been concerned with the development of an automated search for impossibility theorems in the field of *ranking sets of objects*.

The initial attempt to realize theorem verification in first-order logic, which we favored due to its supposedly high readability compared to less expressive logics, had to be discarded because of very weak performance of automated provers. We conjectured that this is caused by the large overhead created when implementing sets as additional objects in the universe in order to be able to quantify over them.

Therefore, the next natural choice fell onto propositional logic, which Tang and Lin had already used successfully to automatically prove major impossibility theorems in social choice theory. We were able to extend their method to *ranking sets of objects* and successfully verified its main impossibility, the Kannai-Peleg Theorem, using a combination of a manual inductive step and an automated verification of the corresponding base case using a SAT solver.

A generalized version of this inductive step for a large class of axioms (not only, but also covering the Kannai-Peleg Theorem) was subsequently established with the help of tools from model theory. This result thus reduces impossibility theorems to small instances and therefore allows establishing full impossibility results by just proving that axioms are inconsistent for small instances of the problem. The checking of these base cases can be very difficult and tedious by hand, but can, as seen for the Kannai-Peleg Theorem, be processed on a computer within seconds.

This insight motivated us to implement an automated and exhaustive theorem search that, given a set of axioms, can check all its subsets for inconsistent combinations at different domain sizes. A few experiments with this method proved its fruitfulness and we reported a total of 84 impossibility results (with minimal domain sizes of eight or less elements) on a space of 20 axioms, which corresponds to about one million axiom combinations. We classified these theorems and discussed a selection in detail, also giving manual proofs, which were relatively easy to construct knowing exactly which axioms and domain size to use. Among the results we found quite a few variations of the Kannai-Peleg Theorem, but also surprising results, like an impossibility previously believed to be a possibility as well as an impossibility without any form of dominance principle.

We have thus created a method that can not only be used for formalizing and automatically verifying known impossibility theorems, but also for the discovery of completely new or modified impossibilities and also as a guiding tool for finding general possibilities.

Synoptically, we see our main contribution in two different fields. On the one hand, quite clearly, the discovered impossibility theorems are valuable additions to the field of *ranking sets of objects*. On the other hand, and even more importantly, the method of our theorem search does contribute both to *ranking sets of objects* but also to *automated reasoning*, where it stands for an extension of the technique developed by Tang and Lin, offering a tool that could well be applied to other axioms, areas and disciplines, too. It also underlines once more that satisfiability checking is a task with many application, even with more sophisticated mechanisms like first- and higher-order provers available.

Considering the particular impossibilities found by our program, the question remains whether the Kannai-Peleg Theorem is now just one of many results, or whether it still stands out in some way. The answer to this question is involved, but we think that the Kannai-Peleg Theorem still holds a special position among all impossibility results in the field of *ranking sets of objects* due to the naturalness of its axioms. Even though also some of our impossibility results comprise natural and intuitive combinations of axioms, they do not quite reach the level of the Kannai-Peleg Theorem. It is largely open, however, whether there are striking results in other parts of *ranking sets of objects* that are still unknown, but could easily be discovered using our theorem search. We see a lot of potential in the application of our search, which, amongst other future perspectives, we are going to discuss in the next section.

Future Work

On a general level and motivated by our own results as well as the increasing capabilities of automated provers, we expect more contributions regarding the application of automated reasoning to (social) choice theory, game theory, and related disciplines in the future. These could differ in the logic used for the formalization, but at least for fully automated approaches we see a particular potential in logics with little expressivity but fast algorithms. The universality of higher-order logic comes at the cost of not being efficiently checkable on a computer, and so this part will probably have to be reserved for interactive proof assistants until more powerful systems are available.

More specifically, we see at least three different ways of extending our work that we believe are of interest:

- by simply applying our method to other axiom sets (potentially in other areas),
- by refining our implementation of the theorem search, and
- by modifying this presented technique to be applicable to other settings or further disciplines within or outside of (social) choice theory.

The first idea most likely offers good chances for immediate results. Implementing further axioms can be done quickly, and as long as they are covered by our universal step, results can be read off after a short computation time. Especially in the realm of *opportunity sets*,[98] where to our knowledge no impossibility results are known to date, the potential for success is very high. But also other models of *ranking sets of objects*, like the interpretation of sets as final outcomes,[99] offer a variety of axioms to be found in the literature, the compatibility of which can now be explored easily.

Possibility results can similarly be searched for, but will mostly require a manual proof as described in Section 5.5. In our opinion, the key to applying the method successfully lies in combining the feature of certain SAT solvers to generate multiple satisfying models with a manual search for similarities among these.

Both of the above naturally lead us to the second idea of extending our work further: refining the implementation. For additional axioms it would make sense to integrate a parser or a similar input module that can read our language MSLSP and therefore axioms do no longer have to be transformed and coded by hand.[100] Possibility searches, on the other hand, would benefit from a module reading different satisfying assignments from the solver and automatically generating a "minimal" model consisting of the overlap of all assignments. This would then present researchers with a description of what the entered axioms imply with respect to the ranking of sets.

Two further (and minor) ideas are to parallelize the computation part (for instance using a clever scheduler that hands problems to an array of SAT solvers), or to implement dependencies between the entered axioms. The latter would make sure that only absolutely minimal results are returned, whereas now the reader will remember that some of our results are trivial consequences of others (since some of our axioms implied others).

The third idea of extending our work by adjusting the technique to other disciplines is probably the hardest task. A starting point could be our Preservation Theorem, which can potentially still be strengthened to a larger class

[98] See Section 1.2 for a brief introduction and references.
[99] See again Section 1.2.
[100] TCHALTSEV's TPTP parser in Java [74] could be a good basis for such a module.

of axioms. One could try to find out where the exact borderline lies between formulas that are preserved in certain substructures and those that are not. For arbitrary first-order models this has been done in the famous Łoś-Tarski Theorem, but for our class of *structures for set preferences* it is still an open question.

In theory, the presented method appears to be applicable to a variety of settings and disciplines. *Ranking sets of objects* is a first successful example of such a setting, but we are curious which others will follow.

A

First-Order Logic

In this appendix we include materials related to the approach pursued in Chapter 2 of using a theorem prover on a first-order formalization of the problem under consideration. In particular, we present excerpts of an automatically constructed first-order proof, which unfortunately turned out be be much more complex than we had expected.

A.1 "Trivial" First-Order Proof

The following proof in the PCL2 (Proof Communication Language 2) format[1] was created by *Prover E*. It is the proof of the (very simple) problem *lemma2* (Lemma 1.4 for the case of two elements only) as described in Section 2.3.

Despite the extremely simple nature of the problem, we could not even include the whole proof as produced by *Prover E* here (as it is about 40 pages long), but had to skip 600 (!) lines to reduce its presentation to six pages instead. The proof's length and its complex and difficult to read structure show how far automated theorem provers are from producing human readable proof objects in our specific formalization.

```
# Preprocessing time       : 0.012 s
# Problem is unsatisfiable (or provable), constructing proof object
# SZS status Theorem
# SZS output start CNFRefutation.
     1 :  : ![X1]:![X2]:(w(X1,X2)=>(S(X1)&S(X2))) :
initial("lemma_2.tptp", sos)
     2 :  : ![X3]:![X4]:(l(X3,X4)=>(E(X3)&E(X4))) :
initial("lemma_2.tptp", sos)
```

[1] PCL2 is a proof output protocol language under development by Schulz [63] as a successor to PCL [20, 21].

```
     3 :  : ![X5]:![X6]:(el(X5,X6)=>(E(X5)&S(X6))) :
initial("lemma_2.tptp", sos)
     4 :  : ![X7]:![X8]:![X9]:(((S(X7)&S(X8))&equal(union(X7,X8),
X9))=>S(X9)) : initial("lemma_2.tptp", sos)
     6 :  : ![X12]:![X13]:((E(X12)&equal(singleton(X12), X13))=>S(X13))
: initial("lemma_2.tptp", sos)
     7 :  : ![X14]:(~(E(X14))=>equal(singleton(X14), tptp2)) :
initial("lemma_2.tptp", sos)
     8 :  : ![X15]:![X16]:((S(X15)&equal(min(X15), X16))=>E(X16)) :
initial("lemma_2.tptp", sos)
     9 :  : ![X17]:(~(S(X17))=>equal(min(X17), tptp2)) :
initial("lemma_2.tptp", sos)
    10 :  : ![X18]:![X19]:((S(X18)&equal(max(X18), X19))=>E(X19)) :
initial("lemma_2.tptp", sos)
    11 :  : ![X20]:(~(S(X20))=>equal(max(X20), tptp2)) :
initial("lemma_2.tptp", sos)
    15 :  : ![X30]:(S(X30)=>w(X30,X30)) : initial("lemma_2.tptp", sos)
    16 :  : ![X31]:![X32]:(((S(X31)&S(X32))&~(equal(X31,
X32)))=>(w(X31,X32)|w(X32,X31))) : initial("lemma_2.tptp", sos)
    19 :  : ![X37]:![X38]:(((E(X37)&E(X38))&~(equal(X37,
X38)))=>(l(X37,X38)|l(X38,X37))) : initial("lemma_2.tptp", sos)
    21 :  :
![X42]:![X43]:((((E(X42)&E(X43))&l(X42,X43))&l(X43,X42))=>equal(X42,
X43)) : initial("lemma_2.tptp", sos)
    22 :  : ![X44]:![X45]:((E(X44)&S(X45))=>(equal(singleton(X44),
X45)<=>(el(X44,X45)&![X46]:(~(equal(X46, X44))=>~(el(X46,X45)))))) :
initial("lemma_2.tptp", sos)
    23 :  :
![X47]:![X48]:![X49]:(((S(X48)&S(X49))&S(X47))=>(equal(union(X48,X49),
X47)<=>![X50]:(E(X50)=>(el(X50,X47)<=>(el(X50,X48)|el(X50,X49)))))) :
initial("lemma_2.tptp", sos)
    24 :  : ![X51]:![X52]:((S(X51)&S(X52))=>equal(union(X51,X52),
union(X52,X51))) : initial("lemma_2.tptp", sos)
    25 :  : ![X53]:(E(X53)=>?[X54]:((S(X54)&equal(X54,
singleton(X53)))&![X55]:((S(X55)&~(equal(X55, X54)))=>~(equal(X55,
singleton(X53)))))) : initial("lemma_2.tptp", sos)
    27 :  : ![X60]:![X61]:((S(X60)&S(X61))=>
    (![X62]:(E(X62)=>(el(X62,X60)<=>el(X62,X61)))=>equal(X60, X61))) :
initial("lemma_2.tptp", sos)
    28 :  : (E(a5)&E(a6)) : initial("lemma_2.tptp", sos)
    29 :  : ~(equal(a5, a6)) : initial("lemma_2.tptp", sos)
    30 :  : ![X63]:(E(X63)=>(equal(X63, a6)|equal(X63, a5))) :
initial("lemma_2.tptp", sos)
    31 :  : ![X64]:(E(X64)=>(~(S(X64))&~(equal(X64, tptp2)))) :
initial("lemma_2.tptp", sos)
    32 :  : ![X65]:(S(X65)=>(~(E(X65))&~(equal(X65, tptp2)))) :
initial("lemma_2.tptp", sos)
    34 :  : ![X67]:((E(X67)|S(X67))|equal(X67, tptp2)) :
initial("lemma_2.tptp", sos)
```

```
    35 :  : ![X68]:![X69]:((S(X68)&E(X69))=>(equal(min(X68),
X69)<=>(el(X69,X68)&![X70]:((el(X70,X68)&~(equal(X70,
X69)))=>(l(X70,X69)&~(l(X69,X70))))))) : initial("lemma_2.tptp", sos)
    36 :  : ![X71]:![X72]:((S(X71)&E(X72))=>(equal(max(X71),
X72)<=>(el(X72,X71)&![X73]:((el(X73,X71)&~(equal(X73,
X72)))=>(l(X72,X73)&~(l(X73,X72))))))) : initial("lemma_2.tptp", sos)
    37 : conj :
![X74]:(S(X74)=>(w(X74,union(singleton(min(X74)),singleton(max(X74))))
    &w(union(singleton(min(X74)),singleton(max(X74))),X74))) :
initial("lemma_2.tptp", goals)
    38 : neg :
~(![X74]:(S(X74)=>(w(X74,union(singleton(min(X74)),singleton(max(X74))))
    &w(union(singleton(min(X74)),singleton(max(X74))),X74)))) :
assume_negation(37)
    40 :  : ![X14]:(~(E(X14))=>equal(singleton(X14), tptp2)) :
fof_simplification(7)
    41 :  : ![X17]:(~(S(X17))=>equal(min(X17), tptp2)) :
fof_simplification(9)
    42 :  : ![X20]:(~(S(X20))=>equal(max(X20), tptp2)) :
fof_simplification(11)
    46 :  : ![X44]:![X45]:((E(X44)&S(X45))=>(equal(singleton(X44),
X45)<=>(el(X44,X45)&![X46]:(~(equal(X46, X44))=>~(el(X46,X45)))))) :
fof_simplification(22)
    47 :  : ![X64]:(E(X64)=>(~(S(X64))&~(equal(X64, tptp2)))) :
fof_simplification(31)
    48 :  : ![X65]:(S(X65)=>(~(E(X65))&~(equal(X65, tptp2)))) :
fof_simplification(32)
    50 :  : ![X68]:![X69]:((S(X68)&E(X69))=>(equal(min(X68),
X69)<=>(el(X69,X68)&![X70]:((el(X70,X68)&~(equal(X70,
X69)))=>(l(X70,X69)&~(l(X69,X70))))))) : fof_simplification(35)
    51 :  : ![X71]:![X72]:((S(X71)&E(X72))=>(equal(max(X71),
X72)<=>(el(X72,X71)&![X73]:((el(X73,X71)&~(equal(X73,
X72)))=>(l(X72,X73)&~(l(X73,X72))))))) : fof_simplification(36)
    52 :  : ![X1]:![X2]:(~(w(X1,X2))|(S(X1)&S(X2))) : fof_nnf(1)
    53 :  : ![X3]:![X4]:(~(w(X3,X4))|(S(X3)&S(X4))) :
variable_rename(52)
    54 :  : ![X3]:![X4]:((S(X3)|~(w(X3,X4)))&(S(X4)|~(w(X3,X4)))) :
distribute(53)
    55 :  : /* 0; 0; 7(0)=1(0)*/[++S(X2),--w(X1,X2)] :
split_conjunct(54)
    56 :  : /* 0; 0; 5(0)=1(0)*/[++S(X1),--w(X1,X2)] :
split_conjunct(54)
    57 :  : ![X3]:![X4]:(~(l(X3,X4))|(E(X3)&E(X4))) : fof_nnf(2)
    58 :  : ![X5]:![X6]:(~(l(X5,X6))|(E(X5)&E(X6))) :
variable_rename(57)
    59 :  : ![X5]:![X6]:((E(X5)|~(l(X5,X6)))&(E(X6)|~(l(X5,X6)))) :
distribute(58)
    60 :  : /* 0; 0; 582(0)=1(0)*/[++E(X2),--l(X1,X2)] :
split_conjunct(59)
```

```
    61 :  : /* 0; 0; 584(0)=1(0)*/[++E(X1),--l(X1,X2)] :
split_conjunct(59)
    62 :  : ![X5]:![X6]:(~(el(X5,X6))|(E(X5)&S(X6))) : fof_nnf(3)
    63 :  : ![X7]:![X8]:(~(el(X7,X8))|(E(X7)&S(X8))) :
variable_rename(62)
    64 :  : ![X7]:![X8]:((E(X7)|~(el(X7,X8)))&(S(X8)|~(el(X7,X8)))) :
distribute(63)
    65 :  : /* 0; 0; 7(0)=1(0)*/[++S(X2),--el(X1,X2)] :
split_conjunct(64)
    66 :  : /* 0; 0; 584(0)=1(0)*/[++E(X1),--el(X1,X2)] :
split_conjunct(64)
    67 :  :
![X7]:![X8]:![X9]:(((~(S(X7))|~(S(X8)))|~(equal(union(X7,X8),
X9)))|S(X9)) : fof_nnf(4)
    68 :  :
![X10]:![X11]:![X12]:(((~(S(X10))|~(S(X11)))|~(equal(union(X10,X11),
X12)))|S(X12)) : variable_rename(67)
    69 :  : /* 0; 0; 5(0)=1(0)*/[++S(X1),--equal(union(X2,X3),
X1),--S(X3),--S(X2)] : split_conjunct(68)
    75 :  : ![X12]:![X13]:((~(E(X12))|~(equal(singleton(X12),
X13)))|S(X13)) : fof_nnf(6)
    76 :  : ![X14]:![X15]:((~(E(X14))|~(equal(singleton(X14),
X15)))|S(X15)) : variable_rename(75)
    77 :  : /* 0; 0; 5(0)=1(0)*/[++S(X1),--equal(singleton(X2),
X1),--E(X2)] : split_conjunct(76)
    78 :  : ![X14]:(E(X14)|equal(singleton(X14), tptp2)) : fof_nnf(40)
    79 :  : ![X15]:(E(X15)|equal(singleton(X15), tptp2)) :
variable_rename(78)
    80 :  : /* 0; 0; 660(0)=55(0)*/[++equal(singleton(X1),
tptp2),++E(X1)] : split_conjunct(79)
    81 :  : ![X15]:![X16]:((~(S(X15))|~(equal(min(X15), X16)))|E(X16))
: fof_nnf(8)
    82 :  : ![X17]:![X18]:((~(S(X17))|~(equal(min(X17), X18)))|E(X18))
: variable_rename(81)
    83 :  : /* 0; 0; 584(0)=1(0)*/[++E(X1),--equal(min(X2),
X1),--S(X2)] : split_conjunct(82)
    84 :  : ![X17]:(S(X17)|equal(min(X17), tptp2)) : fof_nnf(41)
    85 :  : ![X18]:(S(X18)|equal(min(X18), tptp2)) :
variable_rename(84)
    86 :  : /* 0; 0; 682(0)=55(0)*/[++equal(min(X1), tptp2),++S(X1)] :
split_conjunct(85)
    87 :  : ![X18]:![X19]:((~(S(X18))|~(equal(max(X18), X19)))|E(X19))
: fof_nnf(10)
    88 :  : ![X20]:![X21]:((~(S(X20))|~(equal(max(X20), X21)))|E(X21))
: variable_rename(87)
    89 :  : /* 0; 0; 584(0)=1(0)*/[++E(X1),--equal(max(X2),
X1),--S(X2)] : split_conjunct(88)
    90 :  : ![X20]:(S(X20)|equal(max(X20), tptp2)) : fof_nnf(42)
    91 :  : ![X21]:(S(X21)|equal(max(X21), tptp2)) :
variable_rename(90)
    92 :  : /* 0; 0; 704(0)=55(0)*/[++equal(max(X1), tptp2),++S(X1)] :
split_conjunct(91)


--- SKIPPED 600 LINES ---


672056 : neg : /* 0; 0; 18446744073709551615(0)=1753(0)*/[++equal(X1,
max(esk10_0)),--l(X1,max(esk10_0))] : csr(671729,61)
672092 : neg : /* 0; 0; 18446744073709551615(0)=424(0)*/[++equal(X1,
a6),--l(X1,a6),--equal(min(esk10_0), a5)] : spm(672056,501616)
705516 : neg : /* 0; 0; 18446744073709551615(0)=424(0)*/[++equal(X1,
a6),++l(a5,a6),--l(X1,a6)] : spm(672092,531918)
705782 : neg : /* 0; 0; 1753(0)=424(0)*/[++equal(max(esk10_0),
a6),++l(a5,a6)] : spm(705516,477145)
961862 :  : /* 0; 0;
638(0)=18446744073709551613(0)*/[++equal(union(X1,X2),
X3),--el(a5,X2),--el(a5,X3),--el(a6,X1),--el(a6,X3),--S(X1)] :
csr(18597,65)
961863 :  : /* 0; 0;
638(0)=18446744073709551613(0)*/[++equal(union(X1,X2),
X3),--el(a6,X1),--el(a6,X3),--el(a5,X2),--el(a5,X3)] : csr(961862,65)
961865 : neg : /* 0; 0; 638(0)=1748(0)*/[++equal(union(X1,X2),
esk10_0),--el(a5,X2),--el(a5,esk10_0),--el(a6,X1)] : spm(961863,473746)
961968 : neg : /* 0; 0; 638(0)=1748(0)*/[++equal(union(X1,X2),
esk10_0),--el(a5,X2),--$true,--el(a6,X1)] : rw(961865,525390)
961969 : neg : /* 0; 0; 638(0)=1748(0)*/[++equal(union(X1,X2),
esk10_0),--el(a5,X2),--el(a6,X1)] : cn(961968)
1210109 : neg : /* 0; 0; 1751(0)=421(0)*/[++equal(min(esk10_0),
a5),--w(esk10_0,esk10_0),--el(a5,singleton(max(esk10_0))),
--el(a6,singleton(a6))] : spm(570,961969)
1212175 : neg : /* 0; 0; 1751(0)=421(0)*/[++equal(min(esk10_0),
a5),--w(esk10_0,esk10_0),--el(a5,singleton(max(esk10_0))),--$true] :
rw(1210109,26722)
1212176 : neg : /* 0; 0; 1751(0)=421(0)*/[++equal(min(esk10_0),
a5),--w(esk10_0,esk10_0),--el(a5,singleton(max(esk10_0)))] :
cn(1212175)
1212955 : neg : /* 0; 0; 1751(0)=421(0)*/[++equal(min(esk10_0),
a5),--el(a5,singleton(a5)),--w(esk10_0,esk10_0)] : spm(1212176,606413)
1213065 : neg : /* 0; 0; 1751(0)=421(0)*/[++equal(min(esk10_0),
a5),--$true,--w(esk10_0,esk10_0)] : rw(1212955,22281)
1213066 : neg : /* 0; 0; 1751(0)=421(0)*/[++equal(min(esk10_0),
a5),--w(esk10_0,esk10_0)] : cn(1213065)
1213432 : neg : /* 0; 0; 421(0)=424(0)*/[++equal(a5,
a6),++l(a6,a5),--w(esk10_0,esk10_0)] : spm(476251,1213066)
1214084 : neg : /* 0; 0;
2236(0)=1(0)*/[++l(a6,a5),--w(esk10_0,esk10_0)] : sr(1213432,186)
```

```
1215510 : neg : /* 0; 0; 2236(0)=1(0)*/[++l(a6,a5),--S(esk10_0)] :
spm(1214084,118)
1215513 : neg : /* 0; 0; 2236(0)=1(0)*/[++l(a6,a5),--$true] :
rw(1215510,233)
1215514 : neg : /* 0; 0; 2236(0)=1(0)*/[++l(a6,a5)] : cn(1215513)
1215548 : neg : /* 0; 0; 1751(0)=421(0)*/[++equal(min(esk10_0),
a5),++equal(a6, a5)] : spm(667082,1215514)
1215587 : neg : /* 0; 0; 2237(0)=1(0)*/[--l(a5,a6),--$true] :
rw(533663,1215514)
1215588 : neg : /* 0; 0; 2237(0)=1(0)*/[--l(a5,a6)] : cn(1215587)
1215635 : neg : /* 0; 0; 1751(0)=421(0)*/[++equal(min(esk10_0), a5)] :
sr(1215548,186)
1215645 : neg : /* 0; 0; 1753(0)=424(0)*/[++equal(max(esk10_0), a6)] :
sr(705782,1215588)
1217059 : neg : /* 0; 0; 1753(0)=421(0)*/[++equal(max(esk10_0),
a5),--w(esk10_0,union(singleton(a6),singleton(a5))),
--w(union(singleton(a6),singleton(min(esk10_0))),esk10_0)] :
rw(134497,1215635)
1217060 : neg : /* 0; 0; 1753(0)=421(0)*/[++equal(max(esk10_0),
a5),--w(esk10_0,union(singleton(a6),singleton(a5))),
--w(union(singleton(a6),singleton(a5)),esk10_0)] : rw(1217059,1215635)
1267074 : neg : /* 0; 0; 424(0)=421(0)*/[++equal(a6,
a5),--w(esk10_0,union(singleton(a6),singleton(a5))),
--w(union(singleton(a6),singleton(a5)),esk10_0)] : rw(1217060,1215645)
1267075 : neg : /* 0; 0;
2370(0)=1(0)*/[--w(esk10_0,union(singleton(a6),singleton(a5))),
--w(union(singleton(a6),singleton(a5)),esk10_0)] : sr(1267074,186)
1267117 : neg : /* 0; 0;
2250(0)=1(0)*/[--w(esk10_0,esk10_0),--el(a5,singleton(a5)),
--el(a6,singleton(a6))] : spm(1267075,961969)
1267193 : neg : /* 0; 0;
2250(0)=1(0)*/[--w(esk10_0,esk10_0),--$true,--el(a6,singleton(a6))] :
rw(1267117,22281)
1267194 : neg : /* 0; 0;
2250(0)=1(0)*/[--w(esk10_0,esk10_0),--$true,--$true] :
rw(1267193,26722)
1267195 : neg : /* 0; 0; 2250(0)=1(0)*/[--w(esk10_0,esk10_0)] :
cn(1267194)
1267202 : neg : /* 0; 0; 1749(0)=1(0)*/[--S(esk10_0)] :
spm(1267195,118)
1267205 : neg : /* 0; 0; 1(0)=1(0)*/[--$true] : rw(1267202,233)
1267206 : neg : [] : cn(1267205)
1267207 : neg : [] : 1267206 : 'proof'
# SZS output end CNFRefutation
```

B

Propositional Logic

In this appendix we include materials related to the approach pursued in Chapters 3 and 5 of applying a *SAT solver* to a formalization in propositional logic. In particular, we give a source code example from our program for the instantiation of axioms, show output of satisfiable as well as unsatisfiable instances, and present a complete list of the axioms we experimented with together with their formalizations in the language MSLSP (see Section 4.2) and propositional logic.

B.1 Source Code Example for Coding Axioms

We present examples of *source code* as it was used to generate the computer-readable form of axioms in the framework of ranking sets of objects. In particular, we want to underline how close (and hence easily checkable) our source code is to the propositional form of the axioms. The examples we picked are (`GF1`) and (`LIN`) since they offer a good overview of the different constructions used. For each of the two axioms, we first show them in their original and also their propositional form again and then present the corresponding source code.

$$\begin{aligned}(\texttt{LIN})\quad &(\forall x \in X)\; x \mathrel{\dot{\geq}} x\\ &\equiv \bigwedge_{x\in X} l_{x,x},\end{aligned}$$

$$\begin{aligned}&(\forall x \in X)(\forall y \in X)\left[x \neq y \to x \mathrel{\dot{\geq}} y \vee x \mathrel{\dot{\leq}} y\right]\\ &\equiv \bigwedge_{x\in X} \bigwedge_{\substack{y\in X\\ y\neq x}} \left[l_{x,y} \vee l_{y,x}\right],\end{aligned}$$

$$\begin{aligned}&(\forall x \in X)(\forall y \in X)(\forall z \in X)\left[x \mathrel{\dot{\geq}} y \wedge y \mathrel{\dot{\geq}} z \to x \mathrel{\dot{\geq}} z\right]\\ &\equiv \bigwedge_{x\in X} \bigwedge_{y\in X} \bigwedge_{z\in X} \left[\neg l_{x,y} \vee \neg l_{y,z} \vee l_{x,z}\right],\end{aligned}$$

$$\begin{aligned}&(\forall x \in X)(\forall y \in X)\left[\left(x \mathrel{\dot{\geq}} y \wedge y \mathrel{\dot{\geq}} x\right) \to x = y\right]\\ &\equiv \bigwedge_{x\in X} \bigwedge_{\substack{y\in X\\ y\neq x}} \left[\neg l_{x,y} \vee \neg l_{y,x}\right].\end{aligned}$$

```
private void linL(NewWriter writer) throws IOException
{
        writer.comment("Linear order of elements");
        for (int x=0;x<E;x++) //reflexivity
        {
                writer.var(l(x,x));
                writer.newClause();
        }
        for (int x=0;x<E;x++) //completeness
        {
                for (int y=0;y<E;y++)
                {
                        if (x!=y)
                        {
                                writer.var(l(x,y));
                                writer.var(l(y,x));
                                writer.newClause();
                        }
                }
        }
        for (int x=0;x<E;x++) //transitivity
        {
                for (int y=0;y<E;y++)
                {
                        for (int z=0;z<E;z++)
                        {
                                {
                                        writer.not(l(x,y));
                                        writer.not(l(y,z));
                                        writer.var(l(x,z));
                                        writer.newClause();
                                }
                        }
                }
        }
        for (int x=0;x<E;x++) //antisymmetry
        {
                for (int y=0;y<E;y++)
                {
                        if (x!=y)
                        {
                                writer.not(l(x,y));
                                writer.not(l(y,x));
                                writer.newClause();
                        }
                }
        }
}
```

$$
\begin{aligned}
\texttt{(GF1)}\quad & (\forall A \in \mathcal{X})(\forall x \in X)[((\forall a \in A) x \dot{>} a) \to A \cup \{x\} \succ A] \\
& \equiv \bigwedge_{A \in \mathcal{X}} \bigwedge_{x \in X} \Bigg[\Bigg(\Bigg(\bigvee_{a \in A} \neg l_{x,a} \vee l_{a,x} \Bigg) \vee w_{A \cup \{x\}, A} \Bigg) \wedge \\
& \qquad \Bigg(\Bigg(\bigvee_{a \in A} \neg l_{x,a} \vee l_{a,x} \Bigg) \vee \neg w_{A, A \cup \{x\}} \Bigg) \Bigg],
\end{aligned}
$$

```
private void gf1(NewWriter writer) throws IOException
{
        writer.comment("Gardenfors 1");
        for (int s=0;s<S;s++)
        {
                for (int x=0;x<E;x++)
                {
                        for (int a=0;a<E;a++)
                        {
                                if (isElement(a,s))
                                {
                                        writer.not(l(x,a));
                                        writer.var(l(a,x));
                                }
                        }
                        writer.var(w(union(s,singleton(x)),s));
                        writer.newClause();
                        for (int a=0;a<E;a++)
                        {
                                if (isElement(a,s))
                                {
                                        writer.not(l(x,a));
                                        writer.var(l(a,x));
                                }
                        }
                        writer.not(w(s,union(s,singleton(x))));
                        writer.newClause();
                }
        }
}
```

B.2 Output on a Satisfiable Instance

Here the reader can find the output as given by our program (wrapping the SAT solver *zChaff*)[2] for a possible problem instance. We used the axioms of the Kannai-Peleg Theorem with a domain size of five elements. The output shows that this is indeed a possible problem instance, in the sense that there exists a weak order on the space $\mathcal{X}$ of non-empty subsets satisfying the given axioms.

The list of numbers represents an assignment for the 986 propositional variables involved (where a "-" indicates that a variable is set to *false*) that satisfies the input formula (problem description). Below the output of the SAT solver our program gives a more human-readable description of the model represented by the assignment: first in its original form, then with renamed elements in favour of a more intuitive description.

```
Problem 8478 (7 axioms): (|X|=5)
Counting clauses...
Creating file...
Coding axioms...
Saving file...
Done.

Cleaning up (removing clauses containing both polarities of a
literal)...
Collected statistics with error code 0.

Z-Chaff Version: zChaff 2007.3.12
Solving kannai_size5_cleaned.cnf ......

c 30356 Clauses are true, Verify Solution successful.
Instance Satisfiable
1 2 3 4 5 -6 7 8 9 10 -11 -12 13 -14 -15 -16 -17 18 19 -20 -21 -22 23
24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
48 49 50 51 52 53 54 55 56 -57 58 -59 60 61 62 63 64 -65 66 -67 68 69
70 71 72 -73 74 -75 76 77 78 79 80 -81 82 -83 84 85 86 87 -88 89 90 91
92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111
112 113 114 115 116 117 118 -119 -120 -121 122 -123 -124 -125 -126 -127
-128 -129 -130 -131 -132 -133 -134 -135 -136 -137 -138 -139 -140 -141
-142 -143 -144 -145 -146 -147 -148 -149 -150 -151 -152 153 154 155 156
-157 -158 -159 -160 161 162 163 164 -165 -166 -167 -168 169 170 171 172
-173 -174 -175 -176 177 178 179 180 -181 -182 -183 184 -185 186 -187
-188 -189 -190 -191 192 -193 194 -195 -196 -197 -198 -199 200 -201 202
-203 -204 -205 -206 -207 208 -209 210 -211 -212 -213 -214 215 216 217
218 -219 -220 -221 -222 223 224 225 226 -227 -228 -229 -230 231 232 233
234 -235 -236 -237 -238 239 240 241 242 -243 -244 -245 246 247 248 249
```

[2] See Section 3.4.1 for details.

```
250 -251 -252 -253 254 255 256 257 -258 -259 -260 -261 262 263 264 265
-266 -267 -268 -269 270 271 272 273 -274 275 -276 277 278 279 280 281
282 283 284 285 286 287 288 289 -290 291 -292 293 294 295 296 297 298
299 300 301 302 303 304 -305 -306 -307 308 309 310 311 312 -313 314
-315 316 317 318 319 320 -321 -322 -323 324 325 326 327 328 -329 330
-331 332 333 334 335 -336 337 -338 339 340 341 342 343 344 345 346 347
348 349 350 351 -352 353 -354 355 356 357 358 359 360 361 362 363 364
365 366 -367 -368 -369 370 -371 -372 -373 -374 -375 -376 -377 378 -379
-380 -381 -382 -383 -384 -385 -386 -387 -388 -389 -390 -391 -392 -393
-394 -395 -396 -397 -398 -399 -400 401 402 403 404 -405 -406 -407 -408
409 410 411 412 -413 -414 -415 -416 417 418 419 420 -421 -422 -423 -424
425 426 427 428 -429 -430 -431 432 -433 434 -435 -436 -437 -438 -439
440 -441 442 -443 -444 -445 -446 -447 448 -449 450 -451 -452 -453 -454
-455 456 -457 458 -459 -460 -461 -462 463 464 465 466 -467 -468 -469
-470 471 472 473 474 -475 -476 -477 -478 479 480 481 482 -483 -484 -485
-486 487 488 489 490 -491 -492 -493 494 495 496 497 498 -499 500 -501
502 503 504 505 506 -507 -508 -509 510 511 512 513 514 -515 516 -517
518 519 520 521 -522 523 -524 525 526 527 528 529 530 531 532 533 534
535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552
-553 -554 -555 556 557 558 559 560 -561 562 -563 564 565 566 567 568
-569 570 -571 572 573 574 575 576 -577 578 -579 580 581 582 583 -584
585 -586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601
602 603 604 605 606 607 608 609 610 611 612 613 614 -615 -616 -617 618
-619 -620 -621 -622 -623 -624 -625 626 -627 -628 -629 -630 -631 -632
-633 634 -635 -636 -637 -638 -639 -640 -641 642 -643 -644 -645 -646
-647 -648 649 650 651 652 -653 -654 -655 -656 657 658 659 660 -661 -662
-663 -664 665 666 667 668 -669 -670 -671 -672 673 674 675 676 -677 -678
-679 680 -681 682 -683 -684 -685 -686 -687 688 -689 690 -691 -692 -693
-694 -695 696 -697 698 -699 -700 -701 -702 -703 704 -705 706 -707 -708
-709 -710 711 712 713 714 -715 -716 -717 -718 719 720 721 722 -723 -724
-725 -726 727 728 729 730 -731 -732 -733 -734 735 736 737 738 -739 -740
-741 742 743 744 745 746 -747 -748 -749 750 751 752 753 -754 -755 -756
-757 758 759 760 761 762 -763 -764 -765 766 767 768 769 -770 771 -772
773 774 775 776 777 778 779 780 781 782 783 784 785 -786 787 -788 789
790 791 792 793 794 795 796 797 798 799 800 -801 -802 -803 804 805 806
807 808 -809 810 -811 812 813 814 815 816 -817 -818 -819 820 821 822
823 824 -825 826 -827 828 829 830 831 -832 833 -834 835 836 837 838 839
840 841 842 843 844 845 846 847 -848 849 -850 851 852 853 854 855 856
857 858 859 860 861 862 -863 -864 -865 866 -867 -868 -869 -870 -871
-872 -873 874 -875 -876 -877 -878 -879 -880 -881 882 -883 -884 -885
-886 -887 -888 -889 890 -891 -892 -893 -894 -895 -896 897 898 899 900
-901 -902 -903 -904 905 906 907 908 -909 -910 -911 -912 913 914 915 916
-917 -918 -919 -920 921 922 923 924 -925 -926 -927 928 -929 930 -931
-932 -933 -934 -935 936 -937 938 -939 -940 -941 -942 -943 944 -945 946
-947 -948 -949 -950 -951 952 -953 954 -955 -956 -957 -958 959 960 961
962 -963 -964 -965 -966 967 968 969 970 -971 -972 -973 -974 975 976 977
978 -979 -980 -981 -982 983 984 985 986
Random Seed Used                           0
Max Decision Level                         115
```

```
Num. of Decisions                           505
( Stack + Vsids + Shrinking Decisions )     9 + 495 + 0
Original Num Variables                      986
Original Num Clauses                        30336
Original Num Literals                       90646
Added Conflict Clauses                      20
Num of Shrinkings                           0
Deleted Conflict Clauses                    0
Deleted Clauses                             0
Added Conflict Literals                     42
Deleted (Total) Literals                    0
Number of Implication                       6785
Total Run Time                              0.014338
RESULT:        SAT
Exited with error code 0
Elements: 1 > 2 > 5 > 4 > 3
Sets: {1} > {1,2} > {1,5}~{1,2,5} > {1,4}~{1,2,4}~{1,4,5}~{1,2,4,5} >
{2} > {2,5} > {2,4}~{5}~{2,4,5} > {4,5} > {4} >
{1,3}~{1,2,3}~{1,3,4}~{1,2,3,4}~{1,3,5}~{1,2,3,5}~{1,3,4,5}~{1,2,3,4,5}
> {2,3}~{2,3,4}~{2,3,5}~{2,3,4,5} > {3,5}~{3,4,5} > {3,4} > {3}

After renaming of elements according to their position:
Elements: 1 > 2 > 3 > 4 > 5
Sets: {1} > {1,2} > {1,3}~{1,2,3} > {1,4}~{1,2,4}~{1,3,4}~{1,2,3,4} >
{2} > {2,3} > {2,4}~{3}~{2,3,4} > {3,4} > {4} >
{1,5}~{1,2,5}~{1,4,5}~{1,2,4,5}~{1,3,5}~{1,2,3,5}~{1,3,4,5}~{1,2,3,4,5}
> {2,5}~{2,4,5}~{2,3,5}~{2,3,4,5} > {3,5}~{3,4,5} > {4,5} > {5}
```

B.3 Output on an Unsatisfiable Instance

Below the reader can find the output of our program (wrapping the SAT solver *zChaff*)[3] for an impossible problem instance. It includes a verification of the unsatisfiability as described in Section 3.4.2. We used the axioms of the Kannai-Peleg Theorem with a domain size of six elements. The output shows that this is indeed an impossible problem instance, in the sense that there exists no weak order on the space $\mathcal{X}$ of non-empty subsets satisfying the axioms.

```
Problem 8478 (7 axioms): (|X|=6)
Counting clauses...
Creating file...
Coding axioms...
Saving file...
Done.

Cleaning up (removing clauses containing both polarities of a
literal)...
Collected statistics with error code 0.

Z-Chaff Version: zChaff 2007.3.12
Solving kannai_size6_cleaned.cnf ......

Instance Unsatisfiable
Random Seed Used                           0
Max Decision Level                         252
Num. of Decisions                          3666
( Stack + Vsids + Shrinking Decisions )    1854 + 1812 + 0
Original Num Variables                     4005
Original Num Clauses                       252681
Original Num Literals                      756291
Added Conflict Clauses                     1252
Num of Shrinkings                          0
Deleted Conflict Clauses                   0
Deleted Clauses                            0
Added Conflict Literals                    15910
Deleted (Total) Literals                   0
Number of Implication                      1214745
Total Run Time                             4.7403
RESULT:        UNSAT
Exited with error code 0

Verifiying unsatisfiability from trace...
ZVerify SAT Solver Verifier
Copyright Princeton University, 2003-2004. All Right Reseverd.
```

[3] See Section 3.3.3 for details.

```
COMMAND LINE: SATSolver/zverify_df kannai_size6_cleaned.cnf
resolve_trace
Read in original clauses ...                  252681 Clauses
Mem Usage After Readin file:                  14248
Begin constructing all involved clauses
Num. Learned Clause:                          1252
Num. Clause Built:                            1160
Constructed all involved clauses
Conflict clause verification finished.
Levelize variables...finished
Begin Resolution... Empty clause generated.
Mem Usage :                                   14248
Original Num. Clauses:                        252681
Needed Clauses to Construct Empty:            7893
Total Variable count:                         4005
Variables involved in Empty:                  2350
CPU Time:                                     1.02806
Peak Mem Usage:                               14248
Verification Successful
Exited with error code 0
Unsatisfiability verified successfully!

Axioms are inconsistent for size |X|=6.
```

B.4 List of Axioms Used in the Theorem Search

This is a complete list of the axioms used in the theorem search of Chapter 5. All axioms are given in their natural formulation, their representation in the language MSLSP (to guarantee that the universal step Corollary 4.16 applies) and their formalization in propositional logic[4] in a succinct as well as conjunctive normal form (for the automatic verification described in Sections 3.3).

$(\texttt{LIN}_\varepsilon)$

$$\begin{aligned}
&x \mathrel{\dot{\geq}} x \text{ for all } x \in X && \text{(reflexivity)}\\
\cong\, &\forall_\varepsilon x \left[x \mathrel{\dot{\geq}} x\right]\\
\cong\, &\bigwedge_{x\in X} l_{x,x}\\
\equiv\, &\bigwedge_{x\in X} l_{x,x}
\end{aligned}$$

$$\begin{aligned}
&x \mathrel{\dot{\geq}} y \lor x \mathrel{\dot{\leq}} y \text{ for all } x \neq y \in X && \text{(completeness)}\\
\cong\, &\forall_\varepsilon x \forall_\varepsilon y \left[x \neq y \to (x \mathrel{\dot{\geq}} y \lor x \mathrel{\dot{\leq}} y)\right]\\
\cong\, &\bigwedge_{x\in X}\bigwedge_{y\in X} [x \neq y \to (l_{x,y} \lor l_{y,x})]\\
\equiv\, &\bigwedge_{x\in X}\bigwedge_{\substack{y\in X\\ y\neq x}} [l_{x,y} \lor l_{y,x}]
\end{aligned}$$

$$\begin{aligned}
&x \mathrel{\dot{\geq}} y \land y \mathrel{\dot{\geq}} z \Rightarrow x \mathrel{\dot{\geq}} z \text{ for all } x, y, z \in X && \text{(transitivity)}\\
\cong\, &\forall_\varepsilon x \forall_\varepsilon y \forall_\varepsilon z \left[(x \mathrel{\dot{\geq}} y \land y \mathrel{\dot{\geq}} z) \to x \mathrel{\dot{\geq}} z\right]\\
\cong\, &\bigwedge_{x\in X}\bigwedge_{y\in X}\bigwedge_{z\in X} [(l_{x,y} \land l_{y,z}) \to l_{x,z}]\\
\equiv\, &\bigwedge_{x\in X}\bigwedge_{y\in X}\bigwedge_{z\in X} [\neg l_{x,y} \lor \neg l_{y,z} \lor l_{x,z}]
\end{aligned}$$

$$\begin{aligned}
&x \mathrel{\dot{\geq}} y \land y \mathrel{\dot{\geq}} x \Rightarrow x = y \text{ for all } x, y \in X && \text{(antisymmetry)}\\
\cong\, &\forall_\varepsilon x \forall_\varepsilon y \left[(x \mathrel{\dot{\geq}} y \land y \mathrel{\dot{\geq}} x) \to x = y\right]\\
\cong\, &\bigwedge_{x\in X}\bigwedge_{y\in X} [(l_{x,y} \land l_{y,x}) \to x = y]\\
\equiv\, &\bigwedge_{x\in X}\bigwedge_{\substack{y\in X\\ y\neq x}} [\neg l_{x,y} \lor \neg l_{y,x}]
\end{aligned}$$

[4] As always, we assume that all domains are finite for this to be possible.

(`REFL`$_\sigma$)

$$\begin{aligned} & A \succeq A \text{ for all } A \in \mathcal{X} \\ \cong & \forall_\sigma A \, [A \succeq A] \\ \cong & \bigwedge_{A \in \mathcal{X}} w_{A,A} \\ \equiv & \bigwedge_{A \in \mathcal{X}} w_{A,A} \end{aligned}$$

(`COMPL`$_\sigma$)

$$\begin{aligned} & A \succeq B \vee A \preceq B \text{ for all } A \neq B \in \mathcal{X} \\ \cong & \forall_\sigma A \forall_\sigma B \, [A \neq B \rightarrow (A \succeq B \vee A \preceq B)] \\ \cong & \bigwedge_{A \in \mathcal{X}} \bigwedge_{B \in \mathcal{X}} [B \neq A \rightarrow (w_{A,B} \vee w_{B,A})] \\ \equiv & \bigwedge_{A \in \mathcal{X}} \bigwedge_{\substack{B \in \mathcal{X} \\ B \neq A}} [w_{A,B} \vee w_{B,A}] \end{aligned}$$

(`TRANS`$_\sigma$)

$$\begin{aligned} & A \succeq B \wedge B \succeq C \Rightarrow A \succeq C \text{ for all } A, B, C \in \mathcal{X} \\ \cong & \forall_\sigma A \forall_\sigma B \forall_\sigma C \, [(A \succeq B \wedge B \succeq C) \rightarrow A \succeq C] \\ \cong & \bigwedge_{A \in \mathcal{X}} \bigwedge_{B \in \mathcal{X}} \bigwedge_{C \in \mathcal{X}} [(w_{A,B} \wedge w_{B,C}) \rightarrow w_{A,C}] \\ \equiv & \bigwedge_{A \in \mathcal{X}} \bigwedge_{B \in \mathcal{X}} \bigwedge_{C \in \mathcal{X}} [\neg w_{A,B} \vee \neg w_{B,C} \vee w_{A,C}] \end{aligned}$$

(`EXT`)

$$\begin{aligned} & x \mathrel{\dot{\geq}} y \iff \{x\} \succeq \{y\} \text{ for all } x, y \in X \\ \cong & \forall_\varepsilon x \forall_\varepsilon y \left[x \mathrel{\dot{\geq}} y \leftrightarrow \{x\} \succeq \{y\}\right] \\ \cong & \bigwedge_{x \in X} \bigwedge_{y \in X} \left[l_{x,y} \leftrightarrow w_{\{x\},\{y\}}\right] \\ \equiv & \bigwedge_{x \in X} \bigwedge_{y \in X} \left[(\neg l_{x,y} \vee w_{\{x\},\{y\}}) \wedge (\neg w_{\{x\},\{y\}} \vee l_{x,y})\right] \end{aligned}$$

(GF1)

$$\begin{aligned}
&((\forall a \in A) x \dot{>} a) \Rightarrow A \cup \{x\} \succ A \text{ for all } x \in X \text{ and } A \in \mathcal{X} \\
\cong\, & \forall_\sigma A \forall_\varepsilon x \left[\exists_\varepsilon a (a \in A \wedge x \dot{\not>} a) \vee A \cup \{x\} \succ A\right] \\
\cong\, & \bigwedge_{A \in \mathcal{X}} \bigwedge_{x \in X} \left[\left(\bigwedge_{a \in A} l_{x,a} \wedge \neg l_{a,x}\right) \rightarrow \left(w_{A \cup \{x\}, A} \wedge \neg w_{A, A \cup \{x\}}\right)\right] \\
\equiv\, & \bigwedge_{A \in \mathcal{X}} \bigwedge_{x \in X} \left[\left(\left(\bigvee_{a \in A} \neg l_{x,a} \vee l_{a,x}\right) \vee w_{A \cup \{x\}, A}\right) \wedge \right. \\
& \left. \left(\left(\bigvee_{a \in A} \neg l_{x,a} \vee l_{a,x}\right) \vee \neg w_{A, A \cup \{x\}}\right)\right]
\end{aligned}$$

(GF2)

$$\begin{aligned}
&((\forall a \in A) x \dot{<} a) \Rightarrow A \cup \{x\} \prec A \text{ for all } x \in X \text{ and } A \in \mathcal{X} \\
\cong\, & \forall_\sigma A \forall_\varepsilon x \left[\exists_\varepsilon a (a \in A \wedge x \dot{\not<} a) \vee A \cup \{x\} \prec A\right] \\
\cong\, & \bigwedge_{A \in \mathcal{X}} \bigwedge_{x \in X} \left[\left(\bigwedge_{a \in A} l_{a,x} \wedge \neg l_{x,a}\right) \rightarrow \left(w_{A, A \cup \{x\}} \wedge \neg w_{A \cup \{x\}, A}\right)\right] \\
\equiv\, & \bigwedge_{A \in \mathcal{X}} \bigwedge_{x \in X} \left[\left(\left(\bigvee_{a \in A} \neg l_{a,x} \vee l_{x,a}\right) \vee w_{A, A \cup \{x\}}\right) \wedge \right. \\
& \left. \left(\left(\bigvee_{a \in A} \neg l_{a,x} \vee l_{x,a}\right) \vee \neg w_{A \cup \{x\}, A}\right)\right]
\end{aligned}$$

(SDom)

$$\begin{aligned}
&x \dot{>} y \Rightarrow (\{x\} \succ \{x,y\} \wedge \{x,y\} \succ \{y\}) \text{ for all } x, y \in X \\
\cong\, & \forall_\varepsilon x \forall_\varepsilon y \left[x \dot{>} y \rightarrow (\{x\} \succ \{x\} \cup \{y\} \wedge \{x\} \cup \{y\} \succ \{y\})\right] \\
\cong\, & \bigwedge_{x \in X} \bigwedge_{y \in X} \left[l_{x,y} \wedge \neg l_{y,x} \rightarrow \right. \\
& \left. \left(w_{\{x\},\{x,y\}} \wedge \neg w_{\{x,y\},\{x\}} \wedge w_{\{x,y\},\{y\}} \wedge \neg w_{\{y\},\{x,y\}}\right)\right] \\
\equiv\, & \bigwedge_{x \in X} \bigwedge_{y \in X} \left[(\neg l_{x,y} \vee l_{y,x} \vee w_{\{x\},\{x,y\}}) \wedge (\neg l_{x,y} \vee l_{y,x} \vee \neg w_{\{x,y\},\{x\}}) \wedge \right. \\
& \left. (\neg l_{x,y} \vee l_{y,x} \vee w_{\{x,y\},\{y\}}) \wedge (\neg l_{x,y} \vee l_{y,x} \vee \neg w_{\{y\},\{x,y\}})\right]
\end{aligned}$$

(IND)

$$\begin{aligned}
&A \succ B \Rightarrow A \cup \{x\} \succeq B \cup \{x\} \text{ for all } A, B \in \mathcal{X} \text{ and } x \in X \setminus (A \cup B) \\
\cong\, & \forall_\sigma A \forall_\sigma B \forall_\varepsilon x \left[(x \notin (A \cup B) \wedge A \succ B) \rightarrow A \cup \{x\} \succeq B \cup \{x\}\right] \\
\cong\, & \bigwedge_{A \in \mathcal{X}} \bigwedge_{B \in \mathcal{X}} \bigwedge_{\substack{x \in X \\ x \notin (A \cup B)}} \left[(w_{A,B} \wedge \neg w_{B,A}) \rightarrow w_{A \cup \{x\}, B \cup \{x\}}\right] \\
\equiv\, & \bigwedge_{A \in \mathcal{X}} \bigwedge_{B \in \mathcal{X}} \bigwedge_{\substack{x \in X \\ x \notin (A \cup B)}} \left[\neg w_{A,B} \vee w_{B,A} \vee w_{A \cup \{x\}, B \cup \{x\}}\right]
\end{aligned}$$

(`strictIND`)

$$
\begin{aligned}
&A \succ B \Rightarrow A \cup \{x\} \succ B \cup \{x\} \text{ for all } A, B \in \mathcal{X} \text{ and } x \in X \setminus (A \cup B)\\
\cong\, &\forall_\sigma A \forall_\sigma B \forall_\varepsilon x \left[(x \notin (A \cup B) \wedge A \succ B) \rightarrow A \cup \{x\} \succ B \cup \{x\}\right]\\
\cong\, &\bigwedge_{A \in \mathcal{X}} \bigwedge_{B \in \mathcal{X}} \bigwedge_{\substack{x \in X\\ x \notin (A \cup B)}} \left[(w_{A,B} \wedge \neg w_{B,A}) \rightarrow (w_{A \cup \{x\}, B \cup \{x\}} \wedge \neg w_{B \cup \{x\}, A \cup \{x\}})\right]\\
\equiv\, &\bigwedge_{A \in \mathcal{X}} \bigwedge_{B \in \mathcal{X}} \bigwedge_{\substack{x \in X\\ x \notin (A \cup B)}} \left[(\neg w_{A,B} \vee w_{B,A} \vee w_{A \cup \{x\}, B \cup \{x\}}) \wedge \right.\\
&\qquad\qquad \left. (\neg w_{A,B} \vee w_{B,A} \vee \neg w_{B \cup \{x\}, A \cup \{x\}})\right]
\end{aligned}
$$

(`botIND`)

$$
\begin{aligned}
&A \succ B \Rightarrow A \cup \{x\} \succeq B \cup \{x\}\\
&\text{for all } A, B \in \mathcal{X} \text{ and } x \in X \setminus (A \cup B) \text{ such that } y \dot{>} x \text{ for all } y \in A \cup B\\
\cong\, &\forall_\sigma A \forall_\sigma B \forall_\varepsilon x \left[x \in A \cup B \vee \exists_\varepsilon y (y \in A \cup B \wedge y \dot{\not>} x) \vee A \not\succ B \vee A \cup \{x\} \succeq B \cup \{x\}\right]\\
\cong\, &\bigwedge_{A \in \mathcal{X}} \bigwedge_{B \in \mathcal{X}} \bigwedge_{\substack{x \in X\\ x \notin (A \cup B)}} \left[\left(\bigwedge_{y \in (A \cup B)} l_{y,x} \wedge \neg l_{x,y} \wedge w_{A,B} \wedge \neg w_{B,A}\right) \rightarrow w_{A \cup \{x\}, B \cup \{x\}}\right]\\
\equiv\, &\bigwedge_{A \in \mathcal{X}} \bigwedge_{B \in \mathcal{X}} \bigwedge_{\substack{x \in X\\ x \notin (A \cup B)}} \left[\left(\bigvee_{y \in (A \cup B)} \neg l_{y,x} \vee l_{x,y}\right) \vee \neg w_{A,B} \vee w_{B,A} \vee w_{A \cup \{x\}, B \cup \{x\}}\right]
\end{aligned}
$$

(`topIND`)

$$
\begin{aligned}
&A \succ B \Rightarrow A \cup \{x\} \succeq B \cup \{x\}\\
&\text{for all } A, B \in \mathcal{X} \text{ and } x \in X \setminus (A \cup B) \text{ such that } x \dot{>} y \text{ for all } y \in A \cup B\\
\cong\, &\forall_\sigma A \forall_\sigma B \forall_\varepsilon x \left[x \in A \cup B \vee \exists_\varepsilon y (y \in A \cup B \wedge x \dot{\not>} y) \vee A \not\succ B \vee A \cup \{x\} \succeq B \cup \{x\}\right]\\
\cong\, &\bigwedge_{A \in \mathcal{X}} \bigwedge_{B \in \mathcal{X}} \bigwedge_{\substack{x \in X\\ x \notin (A \cup B)}} \left[\left(\bigwedge_{y \in (A \cup B)} l_{x,y} \wedge \neg l_{y,x} \wedge w_{A,B} \wedge \neg w_{B,A}\right) \rightarrow w_{A \cup \{x\}, B \cup \{x\}}\right]\\
\equiv\, &\bigwedge_{A \in \mathcal{X}} \bigwedge_{B \in \mathcal{X}} \bigwedge_{\substack{x \in X\\ x \notin (A \cup B)}} \left[\left(\bigvee_{y \in (A \cup B)} \neg l_{x,y} \vee l_{y,x}\right) \vee \neg w_{A,B} \vee w_{B,A} \vee w_{A \cup \{x\}, B \cup \{x\}}\right]
\end{aligned}
$$

(`disIND`)

$A \succ B \Rightarrow A \cup \{x\} \succeq B \cup \{x\}$

for all $A, B \in \mathcal{X}$, such that $A \cap B = \emptyset$, and for all $x \in X \setminus (A \cup B)$

$$\cong \forall_\sigma A \forall_\sigma B \forall_\varepsilon x \left[(x \notin (A \cup B) \wedge \text{disjoint}(A,B) \wedge A \succ B) \rightarrow A \cup \{x\} \succeq B \cup \{x\}\right]$$

$$\cong \bigwedge_{A \in \mathcal{X}} \bigwedge_{\substack{B \in \mathcal{X} \\ A \cap B = \emptyset}} \bigwedge_{\substack{x \in X \\ x \notin (A \cup B)}} \left[(w_{A,B} \wedge \neg w_{B,A}) \rightarrow w_{A \cup \{x\}, B \cup \{x\}}\right]$$

$$\equiv \bigwedge_{A \in \mathcal{X}} \bigwedge_{\substack{B \in \mathcal{X} \\ A \cap B = \emptyset}} \bigwedge_{\substack{x \in X \\ x \notin (A \cup B)}} \left[\neg w_{A,B} \vee w_{B,A} \vee w_{A \cup \{x\}, B \cup \{x\}}\right]$$

(`intIND`)

$A \succ B \Rightarrow A \cup \{x,y\} \succeq B \cup \{x,y\}$ for all $A, B \in \mathcal{X}$ and $x, y \in X \setminus (A \cup B)$

such that $x \mathrel{\dot{>}} z$ and $z \mathrel{\dot{>}} y$ for all $z \in A \cup B$

$$\cong \forall_\sigma A \forall_\sigma B \forall_\varepsilon x \forall_\varepsilon y [x \in A \cup B \vee y \in A \cup B \vee \exists_\varepsilon z (z \in A \cup B \wedge (x \mathrel{\dot{\not>}} z \vee z \mathrel{\dot{\not>}} y)) \vee$$
$$A \not\succ B \vee A \cup \{x\} \cup \{y\} \succeq B \cup \{x\} \cup \{y\}]$$

$$\cong \bigwedge_{A \in \mathcal{X}} \bigwedge_{B \in \mathcal{X}} \bigwedge_{\substack{x \in X \\ x \notin (A \cup B)}} \bigwedge_{\substack{y \in X \\ y \notin (A \cup B)}}$$
$$\left[\left(\bigwedge_{z \in (A \cup B)} l_{x,z} \wedge \neg l_{z,x} \wedge l_{z,y} \wedge \neg l_{y,z} \wedge w_{A,B} \wedge \neg w_{B,A}\right) \rightarrow w_{A \cup \{x,y\}, B \cup \{x,y\}}\right]$$

$$\equiv \bigwedge_{A \in \mathcal{X}} \bigwedge_{B \in \mathcal{X}} \bigwedge_{\substack{x \in X \\ x \notin (A \cup B)}} \bigwedge_{\substack{y \in X \\ y \notin (A \cup B)}}$$
$$\left[\left(\bigvee_{z \in (A \cup B)} \neg l_{x,z} \vee l_{z,x} \vee \neg l_{z,y} \vee l_{y,z}\right) \vee \neg w_{A,B} \vee w_{B,A} \vee w_{A \cup \{x,y\}, B \cup \{x,y\}}\right]$$

(`SUAv`)

$(x \mathrel{\dot{>}} y \wedge y \mathrel{\dot{>}} z) \Rightarrow \{y\} \succ \{x,z\}$ for all $x, y, z \in X$

$$\cong \forall_\varepsilon x \forall_\varepsilon y \forall_\varepsilon z \left[(x \mathrel{\dot{>}} y \wedge y \mathrel{\dot{>}} z) \rightarrow \{y\} \succ \{x\} \cup \{z\}\right]$$

$$\cong \bigwedge_{x \in X} \bigwedge_{y \in X} \bigwedge_{z \in X} \left[(l_{x,y} \wedge \neg l_{y,x} \wedge l_{y,z} \wedge \neg l_{z,y}) \rightarrow (w_{\{y\},\{x,z\}} \wedge \neg w_{\{x,z\},\{y\}})\right]$$

$$\equiv \bigwedge_{x \in X} \bigwedge_{y \in X} \bigwedge_{z \in X} [(\neg l_{x,y} \vee l_{y,x} \vee \neg l_{y,z} \vee l_{z,y} \vee w_{\{y\},\{x,z\}}) \wedge$$
$$(\neg l_{x,y} \vee l_{y,x} \vee \neg l_{y,z} \vee l_{z,y} \vee \neg w_{\{x,z\},\{y\}})]$$

(`SUAp`)

$$
\begin{aligned}
&(x \mathrel{\dot{>}} y \wedge y \mathrel{\dot{>}} z) \Rightarrow \{x,z\} \succ \{y\} \text{ for all } x,y,z \in X \\
\cong\, & \forall_\varepsilon x \forall_\varepsilon y \forall_\varepsilon z \left[(x \mathrel{\dot{>}} y \wedge y \mathrel{\dot{>}} z) \to \{x\} \cup \{z\} \succ \{y\}\right] \\
\cong\, & \bigwedge_{x\in X} \bigwedge_{y\in X} \bigwedge_{z\in X} \big[(l_{x,y} \wedge \neg l_{y,x} \wedge l_{y,z} \wedge \neg l_{z,y}) \to (w_{\{x,z\},\{y\}} \wedge \neg w_{\{y\},\{x,z\}})\big] \\
\equiv\, & \bigwedge_{x\in X} \bigwedge_{y\in X} \bigwedge_{z\in X} \big[(\neg l_{x,y} \vee l_{y,x} \vee \neg l_{y,z} \vee l_{z,y} \vee w_{\{x,z\},\{y\}}) \wedge \\
& \qquad (\neg l_{x,y} \vee l_{y,x} \vee \neg l_{y,z} \vee l_{z,y} \vee \neg w_{\{y\},\{x,z\}})\big]
\end{aligned}
$$

(`STopMon`)

$$
\begin{aligned}
&x \mathrel{\dot{>}} y \Rightarrow \{x,z\} \succ \{y,z\} \text{ for all } x,y,z \in X \text{ such that } x \mathrel{\dot{>}} z \text{ and } y \mathrel{\dot{>}} z \\
\cong\, & \forall_\varepsilon x \forall_\varepsilon y \forall_\varepsilon z \left[(x \mathrel{\dot{>}} z \wedge y \mathrel{\dot{>}} z \wedge x \mathrel{\dot{>}} y) \to \{x\} \cup \{z\} \succ \{y\} \cup \{z\}\right] \\
\cong\, & \bigwedge_{x\in X} \bigwedge_{y\in X} \bigwedge_{z\in X} \big[(l_{x,z} \wedge \neg l_{z,x} \wedge l_{y,z} \wedge \neg l_{z,y} \wedge l_{x,y} \wedge \neg l_{y,x}) \to \\
& \qquad (w_{\{x,z\},\{y,z\}} \wedge \neg w_{\{y,z\},\{x,z\}})\big] \\
\equiv\, & \bigwedge_{x\in X} \bigwedge_{y\in X} \bigwedge_{z\in X} \big[(\neg l_{x,z} \vee l_{z,x} \vee \neg l_{y,z} \vee l_{z,y} \vee \neg l_{x,y} \vee l_{y,x} \vee w_{\{x,z\},\{y,z\}}) \wedge \\
& \qquad (\neg l_{x,y} \vee l_{y,x} \vee \neg l_{x,z} \vee l_{z,x} \vee \neg l_{y,z} \vee l_{z,y} \vee \neg w_{\{y,z\},\{x,z\}})\big]
\end{aligned}
$$

(`SBotMon`)

$$
\begin{aligned}
&y \mathrel{\dot{>}} z \Rightarrow \{x,y\} \succ \{x,z\} \text{ for all } x,y,z \in X \text{ such that } x \mathrel{\dot{>}} y \text{ and } x \mathrel{\dot{>}} z \\
\cong\, & \forall_\varepsilon x \forall_\varepsilon y \forall_\varepsilon z \left[(x \mathrel{\dot{>}} y \wedge x \mathrel{\dot{>}} z \wedge y \mathrel{\dot{>}} z) \to \{x\} \cup \{y\} \succ \{x\} \cup \{z\}\right] \\
\cong\, & \bigwedge_{x\in X} \bigwedge_{y\in X} \bigwedge_{z\in X} \big[(l_{x,y} \wedge \neg l_{y,x} \wedge l_{x,z} \wedge \neg l_{z,x} \wedge l_{y,z} \wedge \neg l_{z,y}) \to \\
& \qquad (w_{\{x,y\},\{x,z\}} \wedge \neg w_{\{x,z\},\{x,y\}})\big] \\
\equiv\, & \bigwedge_{x\in X} \bigwedge_{y\in X} \bigwedge_{z\in X} \big[(\neg l_{x,y} \vee l_{y,x} \vee \neg l_{x,z} \vee l_{z,x} \vee \neg l_{y,z} \vee l_{z,y} \vee w_{\{x,y\},\{x,z\}}) \wedge \\
& \qquad (\neg l_{x,y} \vee l_{y,x} \vee \neg l_{x,z} \vee l_{z,x} \vee \neg l_{y,z} \vee l_{z,y} \vee \neg w_{\{x,z\},\{x,y\}})\big]
\end{aligned}
$$

(`evenExt`)

$$
\begin{aligned}
& (A \cup \{x\} \sim \{x\} \wedge A \cup \{y\} \sim \{y\}) \Rightarrow A \cup \{x,y\} \sim \{x,y\} \\
& \quad \text{for all } A \in \mathcal{X}, \text{ such that } |A| \text{ is even, and for all } x, y \in X \setminus A \\
\cong\, & \forall_\sigma A \forall_\varepsilon x \forall_\varepsilon y [(\text{evencard}(A) \wedge x \notin A \wedge y \notin A) \rightarrow \\
& ((A \cup \{x\} \sim \{x\} \wedge A \cup \{y\} \sim \{y\}) \rightarrow A \cup \{x\} \cup \{y\} \sim \{x\} \cup \{y\})] \\
\cong\, & \bigwedge_{\substack{A \in \mathcal{X} \\ |A| \text{ even}}} \bigwedge_{\substack{x \in X \\ x \notin A}} \bigwedge_{\substack{y \in X \\ y \notin A}} \big[(w_{A\cup\{x\},\{x\}} \wedge w_{\{x\},A\cup\{x\}} \wedge w_{A\cup\{y\},\{y\}} \wedge w_{\{y\},A\cup\{y\}}) \rightarrow \\
& \qquad (w_{A\cup\{x,y\},\{x,y\}} \wedge w_{\{x,y\},A\cup\{x,y\}})\big] \\
\equiv\, & \bigwedge_{\substack{A \in \mathcal{X} \\ |A| \text{ even}}} \bigwedge_{\substack{x \in X \\ x \notin A}} \bigwedge_{\substack{y \in X \\ y \notin A}} \\
& \big[(\neg w_{A\cup\{x\},\{x\}} \vee \neg w_{\{x\},A\cup\{x\}} \vee \neg w_{A\cup\{y\},\{y\}} \vee \neg w_{\{y\},A\cup\{y\}} \vee w_{A\cup\{x,y\},\{x,y\}}) \wedge \\
& (\neg w_{A\cup\{x\},\{x\}} \vee \neg w_{\{x\},A\cup\{x\}} \vee \neg w_{A\cup\{y\},\{y\}} \vee \neg w_{\{y\},A\cup\{y\}} \vee w_{\{x,y\},A\cup\{x,y\}})\big]
\end{aligned}
$$

(`MC`)

$$
\begin{aligned}
& A \succeq B \Rightarrow A \cup B \succeq B \;\text{ for all } A, B \in \mathcal{X} \\
\cong\, & \forall_\sigma A \forall_\sigma B\, [A \succeq B \rightarrow A \cup B \succeq B] \\
\cong\, & \bigwedge_{A \in \mathcal{X}} \bigwedge_{B \in \mathcal{X}} [w_{A,B} \rightarrow w_{A\cup B,B}] \\
\equiv\, & \bigwedge_{A \in \mathcal{X}} \bigwedge_{B \in \mathcal{X}} [\neg w_{A,B} \vee w_{A\cup B,B}]
\end{aligned}
$$

References

1. T. Ågotnes, W. van der Hoek, and M. Wooldridge. Reasoning about judgment and preference aggregation. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 07)*, pages 566–573. ACM, 2007.
2. M. Allingham. *Choice theory: A very short introduction*. Oxford University Press, 2002.
3. K. Appel and W. Haken. Every planar map is four colorable. *American Mathematical Society*, 82(5):711–712, 1976.
4. R. Arlegi. A note on Bossert, Pattanaik and Xu's "Choice under complete uncertainty: axiomatic characterization of some decision rules". *Economic Theory*, 22(1):219–225, 2003.
5. K. J. Arrow. A difficulty in the concept of social welfare. *The Journal of Political Economy*, 58(4):328–346, 1950.
6. K. J. Arrow. *Social Choice and Individual Values*. Yale University Press, New Haven, 1963.
7. K. J. Arrow, A. K. Sen, and K. Suzumura. *Handbook of social choice and welfare*. North-Holland, 2002.
8. R. W. Bailey. The number of weak orderings of a finite set. *Social Choice and Welfare*, 15(4):559–562, 1998.
9. S. Barberà. The manipulation of social choice mechanisms that do not leave "too much" to chance. *Econometrica*, 45(7):1573–1588, 1977.
10. S. Barberà, C. R. Barrett, and P. K. Pattanaik. On some axioms for ranking sets of alternatives. *Journal of Economic Theory*, 33(2):301–308, 1984.
11. S. Barberà, W. Bossert, and P. K. Pattanaik. Ranking sets of objects. In S. Barberà, P. J. Hammond, and C. Seidl, editors, *Handbook of Utility Theory*, volume II: Extensions, pages 893–977. Kluwer Academic Publishers, Dordrecht, 2004.
12. S. Barberà and P. K. Pattanaik. Extending an order on a set to the power set: Some remarks on Kannai and Peleg's approach. *Journal of Economic Theory*, 32(1):185–191, 1984.
13. A. Biere. PrecoSAT. Available from `http://fmv.jku.at/precosat/`, May 2010.
14. J. H. Blau. The existence of social welfare functions. *Econometrica*, 25(2):302–313, 1957.

15. W. Bossert. Uncertainty aversion in nonprobabilistic decision models. *Mathematical Social Sciences*, 34(3):191–203, 1997.
16. W. Bossert, P. K. Pattanaik, and Y. Xu. Choice under complete uncertainty: Axiomatic characterizations of some decision rules. *Economic Theory*, 16(2):295–312, 2000.
17. B. Can, B. Erdamar, and M. R. Sanver. Expected utility consistent extensions of preferences. *Theory and Decision*, 67(2):123–144, 2009.
18. Center for Discrete Mathematics & Theoretical Computer Science. DIMACS satisfiability suggested format. Available from ftp://dimacs.rutgers.edu/pub/challenge/satisfiability/doc/satformat.dvi, May 1993.
19. Y. Chevaleyre, U. Endriss, J. Lang, and N. Maudet. A short introduction to computational social choice. In *Proceedings of the 33rd Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM-2007)*, volume 4362 of *LNCS*, pages 51–69. Springer-Verlag, 2007.
20. J. Denzinger and S. Schulz. Analysis and representation of equational proofs generated by a distributed completion based proof system. *SEKI-Report SR-94-05, University of Kaiserslautern*, 1994.
21. J. Denzinger and S. Schulz. Recording and analysing knowledge-based distributed deduction processes. *Journal of Symbolic Computation*, 21(4-6):523–541, 1996.
22. H. B. Enderton. *A mathematical introduction to logic*. Academic Press, 1972.
23. U. Endriss. Sincerity and manipulation under approval voting. Working paper, available from http://staff.science.uva.nl/~ulle/pubs/working-papers/sincere-approval.pdf, October 2009.
24. P. C. Fishburn. Arrow's impossibility theorem: Concise proof and infinite voters. *Journal of Economic Theory*, 2(1):103–106, 1970.
25. P. C. Fishburn. Even-chance lotteries in social choice theory. *Theory and Decision*, 3(1):18–40, 1972.
26. P. C. Fishburn. Comment on the Kannai-Peleg impossibility theorem for extending orders. *Journal of Economic Theory*, 32(1):176–179, 1984.
27. W. Gaertner. *A Primer in Social Choice Theory: Revised Edition*. Oxford University Press, USA, 2009.
28. P. Gammie. Some classical results in social choice theory. In G. Klein, T. Nipkow, and L. Paulson, editors, *The Archive of Formal Proofs*. http://afp.sf.net/entries/SenSocialChoice.shtml, November 2008.
29. P. Gärdenfors. Manipulation of social choice functions. *Journal of Economic Theory*, 13(2):217–228, 1976.
30. P. Gärdenfors. On definitions of manipulation of social choice functions. In J. J. Laffont, editor, *Aggregation and Revelation of Preferences*, pages 29–36. North-Holland, 1979.
31. A. Gibbard. Manipulation of voting schemes: A general result. *Econometrica*, 41(4):587–601, 1973.
32. U. Grandi and U. Endriss. First-order logic formalisation of Arrow's Theorem. In *Proceedings of the 2nd International Workshop on Logic, Rationality and Interaction (LORI-2009)*, volume 5834 of *LNAI*, pages 133–146. Springer-Verlag, 2009.
33. N. Gravel. Ranking opportunity sets on the basis of their freedom of choice and their ability to satisfy preferences: A difficulty. *Social Choice and Welfare*, 15(3):371–382, 1998.

34. W. Hodges. *A shorter model theory*. Cambridge University Press, 1997.
35. R. Holzman. A note on the redundancy of an axiom in the Pattanaik-Peleg characterization of the lexicographic maximin extension. *Social Choice and Welfare*, 1(2):123–125, 1984.
36. P. Jones and R. Sugden. Evaluating choice. *International Review of Law and Economics*, 2(1):47–65, 1982.
37. Y. Kannai and B. Peleg. A note on the extension of an order on a set to the power set. *Journal of Economic Theory*, 32(1):172–175, 1984.
38. K. Korovin. iProver v0.7. `http://www.cs.man.ac.uk/~korovink/iprover/`, December 2009.
39. F. Lin and P. Tang. Computer-aided proofs of Arrow's and other impossibility theorems. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 114–119. AAAI Press, 2008.
40. M. Manzano. *Extensions of first-order logic*. Cambridge University Press, 1996.
41. E. Maskin. Nash equilibrium and welfare optimality. *Review of Economic Studies*, 66(1):23–38, 1999.
42. K. O. May. A set of independent necessary and sufficient conditions for simple majority decision. *Econometrica*, 20(4):680–684, 1952.
43. W. McCune. Otter 3.3 reference manual. Technical Memorandum No. 263, Argonne National Laboratory, 2003.
44. W. McCune. Prover9 v2009-11A. `http://www.cs.unm.edu/~mccune/prover9/`, November 2009.
45. M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th conference on Design automation*, pages 530–535. ACM New York, NY, USA, 2001.
46. E. Muller and M. A. Satterthwaite. The equivalence of strong positive association and strategy-proofness. *Journal of Economic Theory*, 14(2):412–418, 1977.
47. T. Nipkow. Social choice theory in HOL. *Journal of Automated Reasoning*, 43(3):289–304, 2009.
48. T. Nipkow, L.C. Paulson, and M. Wenzel. *Isabelle/HOL: a proof assistant for higher-order logic*. Springer Verlag, 2002.
49. S. I. Nitzan and P. K. Pattanaik. Median-based extensions of an ordering over a set to the power set: An axiomatic characterization. *Journal of Economic Theory*, 34(2):252–261, 1984.
50. D. J. Packard. Preference relations. *Journal of Mathematical Psychology*, 19(3):295–306, 1979.
51. P. K. Pattanaik and B. Peleg. An axiomatic characterization of the lexicographic maximin extension of an ordering over a set to the power set. *Social Choice and Welfare*, 1(2):113–122, 1984.
52. P. K. Pattanaik and Y. Xu. On ranking opportunity sets in terms of freedom of choice. *Recherches économiques de Louvain*, 56(3-4):383–390, 1990.
53. M. Peterson. *An Introduction to Decision Theory*. Cambridge University Press, 2009.
54. C. Puppe. Freedom of choice and rational decisions. *Social Choice and Welfare*, 12(2):137–153, 1995.
55. J. A. Robinson and A. Voronkov. *Handbook of automated reasoning*. North-Holland, 2001.
56. A. Rubinstein. The single profile analogues to multi profile theorems: Mathematical logic's approach. *International Economic Review*, 25(3):719–730, 1984.

57. S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edition, 2010.
58. SARA Reken- en Netwerkdiensten. Lisa national compute cluster. https://subtrac.sara.nl/userdoc/wiki/lisa64/description/.
59. SAT 2005 Competition. http://www.satcompetition.org, July 2005.
60. SAT 2009 Competition. http://www.satcompetition.org, July 2009.
61. SAT Research Group, Princeton University. zChaff. Available from http://www.princeton.edu/~chaff/zchaff.html, March 2007.
62. M. A. Satterthwaite. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10(2):187–217, 1975.
63. S. Schulz. E 1.1x user manual. Available from http://www4.informatik.tu-muenchen.de/~schulz/WORK/eprover.ps, September 2009.
64. A. Sen. The impossibility of a Paretian liberal. *The Journal of Political Economy*, 78(1):152–157, 1970.
65. A. Sen. Welfare, preference and freedom. *Journal of Econometrics*, 50(1-2):15–29, 1991.
66. N. J. A. Sloane, editor. The on-line encyclopedia of integer sequences (OEIS). Published electronically at http://www.research.att.com/~njas/sequences/, June 2010.
67. Sun Microsystems, Inc. Java 2 platform standard edition development kit 5.0. Available from http://java.sun.com/javase/downloads/index_jdk5.jsp, November 2009.
68. G. Sutcliffe. System before TPTP. http://www.cs.miami.edu/~tptp/cgi-bin/SystemB4TPTP/.
69. G. Sutcliffe. The CADE ATP system competition. In *The 22nd International Conference on Automated Deduction*, 2009.
70. G. Sutcliffe. The TPTP problem library and associated infrastructure: The FOF and CNF parts, v3.5.0. *Journal of Automated Reasoning*, 43(4):337–362, 2009.
71. P. Tang. *Computer-Aided Theorem Discovery — A New Adventure and its Application to Economic Theory*. PhD thesis, Hong Kong University of Science and Technology, 2010.
72. P. Tang and F. Lin. Computer-aided proofs of theorems in implementation theory. Manuscript.
73. P. Tang and F. Lin. Computer-aided proofs of Arrow's and other impossibility theorems. *Artificial Intelligence*, 173(11):1041–1053, 2009.
74. A. Tchaltsev. TPTP parser in Java. http://www.freewebs.com/andrei_ch/, January 2007.
75. A. Trybulec and H. Blair. Computer assisted reasoning with Mizar. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 26–28, 1985.
76. A. Voronkov. The anatomy of Vampire. *Journal of Automated Reasoning*, 15(2):237–265, 1995.
77. T. Weber and H. Amjad. Efficiently checking propositional refutations in HOL theorem provers. *Journal of Applied Logic*, 7(1):26–40, 2009.
78. F. Wiedijk. Arrow's Impossibility Theorem. *Formalized Mathematics*, 15(4):171–174, 2007.
79. Wolfram Research, Inc. Mathematica. Version 5.1, Wolfram Research, Inc., Champaign, Illinois, 2004.

80. L. Wos, F. Pereira, R. Hong, R. S. Boyer, J. S. Moore, W. W. Bledsoe, L. J. Henschen, B. G. Buchanan, G. Wrightson, and C. Green. An overview of automated reasoning and related fields. *Journal of Automated Reasoning*, 1(1):5–48, 1985.

Index