FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Advances in Schematic Cut Elimination

DISSERTATION

zur Erlangung des akademischen Grades

Doktor der technischen Wissenschaften

eingereicht von

David Michael Cerna
Matrikelnummer 1048402

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Univ. Prof. Dr. Phil. Alexander Leitsch

Diese Dissertation haben begutachtet:

(Dr. Nicolas Peltier) (Assoc. Prof. Dr. Georg Moser)

Wien, 06.02.2015

(David Michael Cerna)

FAKULTÄT
FÜR !NFORMATIK

Faculty of Informatics

Advances in Schematic Cut Elimination

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor der technischen Wissenschaften

by

David Michael Cerna
Registration Number 1048402

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Univ. Prof. Dr. Phil. Alexander Leitsch

The dissertation has been reviewed by:

(Dr. Nicolas Peltier) (Assoc. Prof. Dr. Georg Moser)

Wien, 06.02.2015

(David Michael Cerna)

Erklärung zur Verfassung der Arbeit

David Michael Cerna
Vorgartenstrasse 129-143/1/6 1020 Vienna,Austia

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift David Michael Cerna)

Acknowledgements

First and foremost, I would like to thank my advisor Alexander Leitsch for all the advice concerning the direction of my thesis and the development of the contained ideas. Especially for dealing with my sometimes *ad hoc* and less than organized style. I would also like to thank the reviewers of my dissertation, Georg Moser and Nicolas Peltier, those of whom both helped in improving this work, and whose advice will probably be applied to future work as well. Other than those, of whom are part of the formal structure required for completion of a dissertation, I would like to thank the Theory and Logic Group at the Technical University for existing and providing a great environment for research. More specifically, the inhabitants of HA402, of whom have been, over the years of my PhD, been great co-workers as well as friends. To name a few, Cvetan Dunchev, Paolo Baldi, Giselle Reis, Tomer Libal, Martin Riener, etc. In terms of financial support, I would like to thank the Vienna PhD School for the scholarship which supported the first three years of my studies and the FWF who supported the last three months. Last but not least, I would like to thank my extended family Nadezhda Vassilyeva, the person who I made this adventurous move to Europe with, and without whom, I would not be completing this work.

Abstract

In this dissertation we study schematic formulae and proofs from both the propositional and first-order perspective. By schematic we mean that the formulae have a free numeric parameter indexing the formulae. In the propositional case we study an extension of a decision procedure for satisfiability of *regular propositional schematic formulae*, defined in [4], allowing for multiple free parameters within a single formula. Our work concerning first-order schematic formulae makes up the major of the dissertation and is mainly concerned with cut-elimination for *proof schemata* using the CERES method [31]. By proof schemata, we are referring to a restricted form of cyclic proofs with a free indexing parameter. The CERES method of cut elimination introduced in [14] extracts what is referred to as the *characteristic clause set* from a sequent calculus proof and uses resolution to refute the clause set. The characteristic clause set represents the cut structure of the sequent calculus proof it is derived from and is always unsatisfiable. Unlike the work of [14], which introduces a complete method for cut elimination in first order logic, [31] only introduces a framework which can be used to define proof schemata and possibly eliminate the cuts within a given proof schemata. The problem with proof schemata is that they essentially define an infinite sequence of proofs and many theoretic results used in [14] no longer hold for proof schemata. For example, reductive cut elimination is not as straight forward as it was for first-order sequent calculus proofs and it is unclear if it is even possible to define a reductive method for proof schemata. Also, most importantly, resolution is on longer a complete method for unsatisfiable clause sets, specifically, it is not complete for unsatisfiable schematic clause sets. This issue with resolution being one of the main issues addressed in this thesis. We approach this problem of the lack of a theoretic backbone for cut-elimination on proof schemata by constructing several formal proofs in a schematic version of the standard sequent calculus, each of which is based on a well studied problem (the infinitary pigeonhole principle) often used in the analysis of proof complexity within a given proof system. Using these formal proofs, we apply the schematic version of the CERES method to each of the proofs to see if the method is expressive enough to handle cut elimination for the given proof schema, and if it is not, we ask if we can generalize certain parts of the method to handle the given proof schema and allow for cut elimination in this more general framework. In the process, we develop a generalized version of the resolution calculus introduced in [31], a possible method for Herbrand sequent extraction, though the method is quite restrictive and relies a lot on the structure of the proof schema, and combinatorial results pertaining to the structure of the characteristic clause sets of the given proof schemata. As a final remark, theorem provers where heavily used in the application of the schematic CERES method to the formalized proof schemata used in this dissertation. We also provide an outline as to how this was done and what results where attained with the aid

of automated theorem provers, as well as, idea on how automated theorem provers might be beneficial to future research in this subject.

Kurzfassung

In dieser Dissertation untersuchen wir Formel- und Beweisschemata in der Aussagen- und in der Prädikatenlogik. Mit “schematischmmeinen wir dass Formeln und Beweise durch einen freien numerischen Parameter indiziert sind. Im propositionalen Fall untersuchen wir eine Erweiterung eines Entscheidungsverfahrens für die Erfüllbarkeit regulärer aussagenlogischer Formelschemata (definiert in [4]) auf mehrere freie Parameter. Der Hauptteil der Arbeit besteht in der Untersuchung von Schnittelimination in prädikatenlogischen Beweisschemata mit Hilfe der CERES-Methode für Beweisschemata defininiert in [31]. Diese Beweisschemata haben eine zyklische Form mit einem freien numerischen Parameter. Die Schnitteliminationsmethode CERES definiert in [14] extrahiert eine sogenannte charakteristische Klauselmenge aus einem Beweis im Sequentialkalkül und widerlegt diese mit Resolution. Die charakteristische Klauselmenge repräsentiert die Schnittstruktur des Beweises und ist immer unerfüllbar. Im Gegensatz zur Methode in [14] welche eine vollständige Schnitteliminationsmethode für die Logik erster Stufe liefert ist die Methode in [31] inhärent unvollständig. Das Hauptproblem besteht darin dass Beweischemata unendliche Folgen von Beweis-Schemata repräsentieren und viele theoretische Resultate in [14] nicht für Schemata gelten. Insbesondere ist auch unklar wie und ob eine reduktive Schnittelimination für Schemata definiert werden kann. Auch die schematische Resolution ist nicht mehr vollständig für unerfüllbare Klausel-Schemata. Das Problem der Spezifikation schematischer Resolutionsbeweise zur Widerlegung der schematischen charakteristischen Klauselmengen nimmt einen zentralen Platz in der Untersuchung ein. Da es dazu keinen theoretischen Hintergrund gibt untersuchen wir das Problem anhand typischer Fallstudien der Beweistheorie (wie das unendliche Schubfachprinzip). An Hand dieser Fallstudien untersuchen wir die Expressivität des Formalismus in [31] und zeigen seine Grenzen auf. Im Zuge dieser Untersuchung verallgemeinern wir den schematischen Resolutionskalkül und entwickeln eine Methode der Herbrand-Extraktion (welche aber relativ eng auf spezifische Schemata beschränkt ist). Abschließend ist zu sagen, dass automatische Beweiser in der Auffindung der Resolutions-Widerlegungs-Schemata eine wichtige Rolle gespielt haben und geben einen Hinweis auf potentielle künftige Verwendung von automatischen Beweisern.

Everything we know is but a small glow,
ourselves a mere spark in this flame,
O, but a spark can be an ember
that through the wind blows,
and alone finds a place to reign.

Contents

CHAPTER 1

Introduction

The concept of schema has been pervasive throughout the history of logic [28]. First-order Peano arithmetic's usage of an induction schema is a well known example of schema in mathematical logic [54]. There are many other less known examples where schemata were used in both propositional and first-order logic to attain proof theoretic results. For example, results pertaining to proof length, unification, construction of 'proof skeletons', and first-order schematic Hilbert-type systems [7, 16, 41, 48, 49]. Also, and most importantly regarding this work, schematic proof structure played a crucial role in the analysis of the Fürstenberg's proof of the infinitude of primes [11].

The formal version of Fürstenberg's proof, written in second-order arithmetic, was converted into a proof schema in the theory of first order arithmetic using one *schematic variable*; a single meta-level free variable used to iterate over object level constructions. The schematic variable indicated the number of prime numbers assumed to exist in the given instance of the formal proof. Cut elimination was performed on the schematic version of the formalised Fürstenberg's proof by instantiation of the schematic variable and analysing, independently, the resulting instances of the proof schema. Due to the complexity of this proof schema, it is still an open problem to perform cut elimination without first instantiating the schematic variable. This amounts to identifying invariants which exists in every instance and construction of a language which can express these invariants in a methodological manner. Thus, one can say, it was only shown up to a certain degree how cut-elimination can be performed on such a large and complex proof. As an end result, the Atomic-Cut Normal Form (ACNF) for a given instance was shown equivalent to instances of Euclid's proof of the infinitude of primes. Therefore, as one can see from this outline of the foundational work on the usage of proof schema in proof analysis, construction of a language and a procedure for discovering these invariants are crucial problems needing to be worked on. In the propositional case, this work will be discussed in Ch. 2, methods for automatic invariant discovery given a quite restricted language have been discovered. However, another result by Aravantinos et al. [4] illustrated how sensitive these languages are to changes, and in turn how easy it is to go from decidable invariant discovery to undecidable invariant discovery. Even though there is this slight relationship between Ch. 2 and the rest of the thesis, one ought to consider Ch. 2 independently from the work concerning schematic cut elimination. To elucidate the reader about the relationship between Ch. 2, consider that the structure of proofs of predicate logic formulae can be studied using propositional logic by just ignoring the term language, or by grounding the variables in the formula. Thus, it is possible to use some of the construction found in our propositional work to better understand the structure of predicate logic proofs. In the case of Ch. 2, we see that given a formula, first order or propositional, if there is only a "weak" relationship between two arithmetic variables (free parameters), then it is possible to have multiple parameters in the proof without adding much complexity to the proof.

The work on Fürstenberg's proof opened the doors to considering not only proofs written in a logical calculus, but proofs written as a sequence of proofs in a logical calculus, where the final theorem is the fix point of the series of end sequents. Proof theoretically, these "Proof Schemata" were easier to handle than their higher-order counter parts (less clause complexity in the characteristic clause set for example) and take a much more natural form expressible in language commonplace in mathematical text, namely, they are recursively definable at the meta-level rather than at the object level. While proof schemata are nothing more than sets of proofs of one parameter definable first-order formulae, the consideration of studying such objects, as proof theory has studied various other mathematical constructions, was a novel idea at the time. Though, since the publication of [11] the concept has not gained much traction. The justification for the lack of interest can be found in the difficulty of implementing cut-elimination and other elementary proof theoretic concepts on proof schemata. In a sense the concept is general enough that finding overarching properties is very difficult. As it was shown in [30], Gentzen's reductive cut elimination method fails when an induction rule is added to the calculus. A simple explanation for this failure is that the induction rule introduces a new eigenvariable which is not present above the main sequent of the induction rule. The work of [30] gets around this problem by replacing the introduced eigenvariable by a free parameter over the natural numbers which indexes enumerated formulae and terms. It is not clear how Gentzen's reductive cut elimination method can be generalized to such proofs, or if it is even possible to do so. Also in the same work, it was shown that many important procedures (such as unification) cannot be automated over primitive recursively defined proof schemata. As many properties of recursive functions were shown undeciable in recursion theory, analogously, using recursion as the foundation of our proof schema leads to these results carrying over to our proof schema as well.

Other interesting issues regarding the recursive nature of proof schemata show up when one considers the resolution refutation of a proof schema's characteristic clause set. Namely, the problems are that the refutation will also be recursive, and primitive recursion happens not to be enough for expressing both the computational properties and the proof theoretic structure simultaneous for complex proofs. To be more precise, it is possible to use primitive recursion for defining the various parts of the schematic CERES method, though for more complex proofs, for example the NiA-schema of Ch. 6, the complexity of the necessary encoding of all the required information out weighs any benefit of sticking with simple primitive recursion. Thus, though it is possible that primitive recursion can express every function we will need for proof analysis of primitive recursive proof schemata, figuring out how to write primitive recursive descriptions such that the computation tree of these primitive recursive functions follow, precisely that is, the structure of the refutation of the clause set is highly non-trivial. This issue is addressed in

Ch. 6 & 7. We have not found a way to express the refutation of the proof schema of Ch. 6 primitive recursively, however, we provide a slightly generalized syntactic structure for expressing it. Through out this dissertation, we will see that the schematic resolution refutation is at the heart of almost every important problem pertaining to schematic cut elimination, mainly because schematic cut elimination requires, almost literally, injecting a recursive proof into a purely computational method, resolution.

Though, the work of [11] provided a procedure for using the CERES method to perform cut-elimination on a mathematically significant formal proof, a generalized method had yet been devised, specifically a CERES method which could handle a very general version of schema of proofs. Very recently, a schematic cut-elimination method for first-order logic, such that an arbitrary number of cuts can be eliminated without instantiating the free parameter of the proof [30, 31] was developed. In Ch. 4, we provide most of the results as they are found in [30].

In the work of [30] and later [31] an incomplete, sound and semi-consistent (one has to prove that the resolution refutation used is a refutation) was discovered. The method was based on the CERES method [14]. Though, using this method is more of art than automated theorem proving in that many of the most important parts are left to the user to verify correctness. But, the method provides a framework where one can consider questions about cut-elimination, Herbrand sequent extraction , and other proof theoretic concepts on proof schemata. Also, it is one of the first investigations into this non-trivial and important subject.

In this dissertation there are two main focus of investigation. Firstly, we investigate decidable extensions of regular schemata [4] which allow for multiple parameters (This chapter provides all the results of [25]) and secondly, we investigate cut-elimination on proof schemata. Our work pertaining to extensions of regular schemata is self contained and only makes up a small portion of the bulk of our work. Our investigation into cut-elimination on proof schemata can be broken into three parts. Initially, we worked with the existing schematic CERES method attempting to test the limits of the method by finding proof schemata which can be analysed using the schematic CERES method and proof schemata which cannot be analysed using the schematic CERES method.

As mentioned earlier, the boundary between decidability and undecidability for these problems is easy to cross and in making the huge leap to first-order schemata we invariably must cross it. To make matters worse the concept of invariant discovery sits at the heart of the schematic CERES method, and thus, the essential parts, the formal proofs and the resolution refutation of the clause set, require proofs that, within the languages they are expressed in, they construct what they are meant to construct. To be more specific, being that the initial formal proof needs to be recursively defined, the invariants of the arguments need to be found by the one who constructs the proof. This is not too much to ask for from the user, however, the same process must be repeated after extraction of the characteristic clause set of the formal proof schema. The invariants from the original proof schema with cuts cannot be reused in the process of finding a resolution refutation of the characteristic clause set. It is quite possible that they can help aid the process of finding new invariants, but not much else can be expected.

In terms of analysis of proofs using the current method for cut-elimination for proof schemata we mainly focused on proof schema with a single free parameter. Though, in Ch. 9 we consider a two parameter proof schema, we only work with one of the free parameters instantiated.

Our motivation for ignoring the possibility of more than one free parameter is the difficulties encountered in the single parameter case will be exacerbated greatly in the multiple free parameter case. Thus, without a hard foundation, we did not see the point of pushing things further to a much harder case, but in future work, if multiple parameters will be considered, there is at least a formal proof schema constructed which can be used as a test case.

We first found a proof schema (Non-injectivity assertion schema (NiA schema)) which could not be analysed using the schematic CERES method (the method discussed in Ch. 4 from the paper [31]). From now on, when we state schematic CERES method, we are referring specifically to the method of Ch. 4). It was from the analysis of the NiA-schema that a proof schema (Eventually constant statement schema (ECS-schema)) which could be analysed by the schematic CERES method was found. The ECS-schema is actually a weakened version of the assertion made by the Non-injectivity assertion schema. We add additional assumptions about the considered function.

To provide a little more background pertaining to the motivation behind these two proof schemata, the NiA schema is based on the infinitary pigeonhole principle , which, for the propositional case, has been heavily studied in literature [17, 22–24, 44, 50, 51]. The ECS schema is a monotonic version of the infinitary pigeon hole principle . Many of our results are reflected in literature, but never from the proof analysis direction.

Though the ideas behind the NiA schema are based on the pigeonhole principle, it is, more specifically, a generalization of the *Tape Proof* found in [8, 9, 55]. The basic idea behind this proof is that an infinite tape Turing machine with a two letter alphabet will have at least one letter repeated infinitely often. In the case of the NiA schema, the size of the alphabet can be an arbitrary value.

Once we found out that the NiA schema could not be analysed completely using the schematic CERES method, we attempted to extend the schematic CERES method in order to accommodate the additional structure needed for NiA schema. In the process of generalizing the schematic CERES method we found a mathematical proof of the refutability (we use "mathematical" to imply that the proof is not a fully formal logical proof) of the clause set of the NiA schema as well as results pertaining to the combinatorial structure of the set of derivable clauses for the clause set of the NiA schema. This combinatorial structure plays an integral part in the construction of a more expressive language for the resolution refutation calculus of the schematic CERES method. Interesting enough, and almost expected, the set of all permutations of size n, where n is the free parameter of the proof schema, was the necessary combinatorial structure needed for the refutation as well as construction of the language capable of expressing the refutation. This result will be explained in greater detail in Ch. 6.

Many of these results were attained with the help of the SPASS Theorem Prover [56]. It is not possible to use theorem proves to construct a schematic refutation, however, if one is attempting to show that instances of a clause set are refutable it is possible. From these refutations of instances of the clause set found using SPASS, we found patterns in the derived clauses used by SPASS which helped us construct the induction invariants needed for the proof of refutability. These empirical results as well as analysis will be explained to some extent in Ch. 5 and in more detail in Ch. 10. There is no special reason why we chose to use SPASS almost exclusively rather than other theorem provers, other then the output of this prover is one of the most readable output formats. Though, prover9 [45], being another prominent theorem prover, could not refute many clause sets refutable by SPASS, especially the clause set for larger instantiations of the free parameter. We ended up using VAMPIRE [40] for a few of the hardest refutations, but barely did any analysis of VAMPIRE's output due to the difficulty in reading its output language, especially when the output is a thousand lines long. We did not use any other provers for the work in this thesis.

A problem with the more expressive language which resulted from the analysis of the NiA-schema is that we only have one example which needs this language and this example happens to be the example used to design the language. Thus, we are not completely sure whether this generalization is actually a special case or not. Thus, in an attempt to get a slightly weaker example which might be more expressive than the original language, but within the bounds of the new language, we kept a very similar proof structure to the NiA-schema, but changed the cut formulae by switching the quantifier order. The cuts of the NiA-schema are $\forall\exists$ and the cuts of the new proof are $\exists\forall$. It is known that this type of quantifier switch greatly reduces the complexity of the formula [32]. The new schema which was the result of this switch is the ECS-schema (the same proof schema mentioned earlier in this chapter). As we mentioned earlier, it turned out to fit into the expressive power of the original language for schematic resolution refutation, but it nonetheless gave us insight into the boundary between the two languages. These results as well as a complete resolution refutation can be found Ch. 5

SPASS was even more useful in the case of the ECS-schema in that the induction invariant needed for the resolution refutation was found in the SPASS output. Also, the second order unifier was easily derived from the unifiers used by SPASS.

The combinatorial structure and its role in the language of the resolution calculus lead to the next part of the investigation into cut-elimination on proof schemata. A result of the proof constructing the refutation of the NiA-schema was that an upper bound on the size of our ACNF for instances of the cut-eliminated NiA-schema is the factorial function, essentially the proofs are huge. Thus, it is not sensible to generate these ACNFs but rather attempt to get some information out of them, the essential part of the proof. Therefore, we worked on extraction of a Herbrand sequent from the schematic resolution refutation of the NiA schema. This problem was exacerbated by the fact that the shape of the resolution refutations is outside the expressible power of the schematic resolution calculus of the schematic CERES method [31] and our new language does not provide an easy way to extract a schematic Herbrand sequent. We extracted the Herbrand sequent by first generalizing the term language of the proof schema and showing that we can get a Herbrand sequent out of this generalized language and that the original Herbrand sequent is contained in the generalized language. These results can be found in Ch. 8.

The final part of our investigation into cut-elimination on proof schemata deals with general patterns that seem to be present within proof schemata. Namely, underlying combinatorial structure in the set of derivable clauses of the clause sets. These patterns can be seen in the ECS-schema, however, as the schema become more complex (as is the case for the NiA-schema) this combinatorial structure becomes more apparent. Though the importance of this combinatorial structure is still nothing more than a working hypothesis, we believe investigating even more complex and mathematically significant schema can provide insight into the proof theoretic necessity of this structure. Thus, we constructed a more complex schema based on the NiA-schema, namely, the *generalized* NiA-schema, or *g*NiA-schema . This is a two parameter version of the NiA schema. Though the schema has two parameters we will only consider one parameter versions of it, where one of the two parameters is set to a constant. We focus primarily on the combinatorial structure of the set of derivable clauses of the clause sets and show how some combinatorial significant results can arise out of the combinatorial structure of the clause set. We provide a bijection between a combinatorial problem which arises from the clause set and a recently discovered set of permutations. The main conclusion of Ch. 9 is that the combinatorial structure of the set of derivable clauses for the clause set of proof schemata plays an integral role in the proof analysis of the said proof schemata. This was shown so far through the construction of a language for the schematic refutation and extraction of combinatorial significant results from the set of derivable clauses. We leave as an open problem for future work in the subject to find a formal way to characterize this combinatorial structure. We believe that the method used to characterize permutations in the language for the resolution refutation calculus can be generalized to allow for proof analysis of a general set of proof schema, specifically, proof schema with a certain type of recursion. This recursion can be based on iteration of combinatorial structures rather than the natural numbers, as we did for the NiA-schema. There is a lot of work comprised in this dissertation, much of which seems to be going in different directions. In the conclusion we will give a summary of each section and explain what the intention behind the work was. Also, we will link together the different parts of this dissertation in order to give an overview of the ideas behind this work and there relation to research into cut elimination on proof schema. Hence, we will provide in the conclusion an idea of where this work belongs in the body of research into this topic.

CHAPTER 2

Propositional Schemata

2.1 Introduction

The usage of schemata that we will focus on for the majority of this chapter is schemata as an object level construction iterating propositional formulae. The results presented in this chapter where published in the paper [25]. Work on propositional schemata of this form was pioneered by Aravantinos et al. [4] with application to the field of repeated circuit verification. This construction has also resulted in discoveries in the field of inductive theorem proving, namely, construction of classes of inductively defined formulae which have a decidable satisfiability problem and are more expressive than those currently known in the field [4, 36, 39]. Namely, theses classes are *bounded-linear schemata*, *regular schemata*, *nested regular schemata*, and a multiple free parameter generalization of regular schemata through normalized clause set representation, [6]. Decidable classes of inductive theorems discovered by Kapur et al. [36, 39], the most prominent work in this area, were mainly universally quantified. Propositional schemata can express both bounded existential and universal quantification through $\wedge$ and $\vee$ iterations.

Our main results are formalizations of the class of linked schemata and pure overlap schemata, both being multiple free parameter extension of regular schema, together with a decision procedure for both classes. This decision procedure is a simple extension of the ST procedure for STAB [4]. Our decision procedure allows one to avoid the conversion of propositional schemata to normalized clause sets [6]. Our work is of a similar vein as Gentzen's work [35] in which he provided a method to fuse multiple inductions together in Peano arithmetic.

Though both classes of schemata we introduce are subclasses of the schemata representable by normalized clause sets and benefit from the satisfiability procedure of normalized clauses sets [6], the existence of a tableaux-based decision procedure for satisfiability testing for these classes remained an open problem. The benefit of a tableaux-based decision procedure is that one does not need to convert the propositional schemata into CNF form to test satisfiability. Note that, if one wants to keep logical equivalence between the original formula and the CNF form, the conversion can result in an exponential increase in formula size.

In this section, we consider multiple regular schemata (each with its own parameter) such that the propositional symbols of one schema are not found in the other regular schemata. When this property holds, we can use the parts (i.e. the iterations and propositional variables not found in the iterations) to construct a formula with multiple free parameters–we refer to this class as the class of linked schemata. Essentially, we build formulae using the pieces of several regular schemata. Although, this idea is quite simple, it provides a class of schemata extending regular schemata which still has a tableaux-based decision procedure for satisfiability.

Next we investigate when it is possible for the propositional symbols to occur in two or more *linked* regular schemata, i.e. the same propositional symbol has occurrences indexed by two different parameters. To answer this question, we develop the concept of relative pure literals, literals which are pure when considering occurrences indexed by another parameter. This concept is used to construct the class of pure overlap schemata.

Both linked and pure overlap schemata are extensions of regular schemata, but after applying several tableaux extension rules to the constructed tableau, It is possible to reduce the branches of the constructed tableau to tableaux branches which are decidable using the decision procedure for regular schemata. Essentially, they are both propositional extensions of the class. It is not completely clear if these classes of schemata are the most expressive classes such that their satisfiability problem can be reduced to the satisfiability problem for regular schemata. An open problem regarding this point is whether the purity constraint can be relaxed and retain the reduction– results of Aravantinos et al. [4] (Thm. 6.2) suggests that this is not going to be the case.

Overall, our results provide a simpler and more natural alternative to normalized clause set representation when deciding satisfiability for certain classes of multiple-parameter schemata. To provide a little more information concerning the connection this work has to the rest of the dissertation, in particular, the work of [6], one ought to think of this work as providing a detailed study of the structure of propositional schemata from the direction of multiple, but non-overlapping, parameters. Part of this study concerns itself with the structure of proofs (as well as refutations) of such objects. The propositional structure, in contrast to the first-order term structure, plays a large part in the complexity of the proof schemata defined later in this dissertation. This can be seen in both Ch. 6 & 7. Our work in this chapter (the work published in [25]), when put in the context of [6], can also be considered as providing a more detail analysis of the structure of certain propositional schema and their resolution refutations. However, the problem with the resolution calculus provided in [6] is that it is much weaker than what is required for the proof schema contained in this dissertation. This can be gathered from the lack of expressive power the propositional schemata found in [25] have when compared to the work in later chapters of this dissertation. The propositional schemata of [25] are close to the limits of expressivity of the decidable part of the language found in [6]. Thus, in some sense the connection between the work of [6, 25] and this dissertation is quite strong, but the results do not benefit many of the results found in this dissertation in that they where designed for much less expressive languages than what is used in this work.

The rest of this chapter is structured as follows, Sec. 2.2 will be necessary background material from Aravantinos et al. [4], in Sec. 2.3 we formalize the construction of linked schemata, in Sec. 2.4 we formalize the construction of pure overlap schemata , in Sec. 2.5 we provide a decision procedure for the satisfiability problem of pure overlap schemata. Finally, in Sec. 2.6 we conclude the chapter and shortly discuss the open problems. As a final point, the work contained in this chapter is self-contained and has little to do with the chapters following it. Also, this work was published in [25].

2.2 Background

Syntax and Semantics of Propositional Schemata

The indexing language for standard schematic propositional logic as considered in Aravantinos et al. [4] is the set of *linear arithmetic terms* (denoted by $\mathcal{Z}$) built using the language $\{0, s(\cdot), +, -\}$ and a countably infinite set of variables $\mathcal{V}$. Multiplication is considered as a shorthand for terms of the form $x + x + x + x = 4 \cdot x$ and is not a real operator in the language, nor is it a necessary one. To stick to the framework of Aravantinos et al. [4] $\mathbb{Z}$ is considered as the standard model of the terms in $\mathcal{Z}$.

Definition 2.2.1 (Indexed Proposition [4]). *Let $\mathcal{P}$ be a fixed and countably infinite set of propositional symbols. An* indexed proposition *is an expression of the form $p_{\mathbf{a}}$ where $p \in \mathcal{P}$ and $\mathbf{a} \in \mathcal{Z}$. An indexed proposition $p_{\mathbf{a}}$ s.t. $\mathbf{a} \in \mathbb{Z}$ is called a propositional variable.*

Definition 2.2.2 (Formula Schemata [4]). *The set of* formula schemata *is the smallest set satisfying the following properties.*

- $\bot, \top$ *are formula schemata.*
- *If $\mathbf{a}, \mathbf{b} \in \mathcal{Z}$ then $\mathbf{a} < \mathbf{b}$ is a formula schema.*
- *Each indexed proposition is a formula schema.*
- *If ϕ_1, ϕ_2 are formula schemata then $\phi_1 \wedge \phi_2$, $\phi_1 \vee \phi_2$, $\neg\phi_1$ are formula schemata.*
- *If ϕ is a formula schema not containing $<$, and if $\mathbf{a}, \mathbf{b} \in \mathcal{Z}$, where i is an arithmetic variable, then $\bigwedge_{i=\mathbf{a}}^{\mathbf{b}} \phi$, $\bigvee_{i=\mathbf{a}}^{\mathbf{b}} \phi$ are formula schemata.*

Example 2.2.1. *Consider the formula:*

$$\varphi = q_1 \wedge \bigwedge_{i=0}^{n} \left(p_{i+2n} \wedge \left(\bigvee_{j=n}^{2n+1} \neg q_{n-j} \vee q_{j+1} \right) \right) \wedge 0 \leq n$$

φ is a formula schema.

Formula schemata are inherently finite. We will label the indexed propositions, $\top$, $\bot$ and statements of the form $a < b$, as *atoms* . Formula schemata of the form $\bigwedge_{i=\mathbf{a}}^{\mathbf{b}} \phi$ and $\bigvee_{i=\mathbf{a}}^{\mathbf{b}} \phi$ will be called *iterations* . A formula schema whose constituents are any of the following: $\top$, $\bot$, and $a < b$, is an *arithmetic formula* . Also, it is taken as a standard that arithmetic formulae of the form $a < b$ can only occur outside of iterations. This constraint is necessary being that $a < b$ is interpreted as an iteration, i.e.

$$a < b \equiv \bigvee_{i=a+1}^{b} \top \tag{2.1}$$

Also, we use $a = b$ as an abbreviation for $\neg(b < a) \wedge \neg(a < b)$ and $a \leq b$ as an abbreviation for $\neg(b < a)$. Iterations have both *free* and *bound* variables , where free variable and *parameter* are synonymous. A bound variable i is a variable in the scope of an iteration $\Pi_{i=a}^{b}\phi_i$ where $\Pi = \{\bigvee, \bigwedge\}$. A *substitution* is a function mapping all the free variables to linear expressions. If a substitution σ is applied to a schema φ, i.e $\varphi\sigma$ such that the domain of σ is every free variable in φ, then the linear expressions of φ are integer terms, i.e. all indices in φ are variable free.

Definition 2.2.3 (Interpretation [4])**.** *An* interpretation of the schematic language *is a function mapping every parameter to an integer and every propositional variable to a truth value **T** or **F**. The substitution and interpretation will be denoted as σ and $\mathcal{I}$, respectively.*

Example 2.2.2. *An Interpretation $\mathcal{I}$ such that φ from Ex. 2.2.1 is modelled by $\mathcal{I}$ would be $\sigma \equiv \{n \leftarrow 0\}$ and $q_1 = \mathbf{T}, p_0 = \mathbf{T}, q_0 = \mathbf{T}, q_{-1} = \mathbf{T}, q_1 = \mathbf{F}, q_2 = \mathbf{T}$*

Definition 2.2.4 (Semantics of Schematic Formulae [4])**.** *The semantics of a schematic formula φ in a given interpretation I, denoted by $[\![\varphi]\!]_{\mathcal{I}}$, is defined as follows:*

- $[\![\top]\!]_{\mathcal{I}} = \mathbf{T}$ *and* $[\![\bot]\!]_{\mathcal{I}} = \mathbf{F}$
- $[\![\mathbf{a} < \mathbf{b}]\!]_{\mathcal{I}} = \mathbf{T} \Leftrightarrow [\![\mathbf{a}]\!]_{\mathcal{I}} <_{\mathbb{Z}} [\![\mathbf{b}]\!]_{\mathcal{I}}$
- $[\![P_{\mathbf{a}}]\!]_{\mathcal{I}} = \mathcal{I}(P_{[\![\mathbf{a}]\!]_{\mathcal{I}}})$ *for* $P \in \mathcal{P}$
- $[\![\neg\varphi]\!]_{\mathcal{I}} = \mathbf{T} \Leftrightarrow [\![\varphi]\!]_{\mathcal{I}} = \mathbf{F}$
- $[\![\varphi \vee \psi]\!]_{\mathcal{I}} = \mathbf{T} \Leftrightarrow [\![\varphi]\!]_{\mathcal{I}} = \mathbf{T}$ *or* $[\![\psi]\!]_{\mathcal{I}} = \mathbf{T}$
- $[\![\varphi \wedge \psi]\!]_{\mathcal{I}} = \mathbf{T} \Leftrightarrow [\![\varphi]\!]_{\mathcal{I}} = \mathbf{T}$ *and* $[\![\psi]\!]_{\mathcal{I}} = \mathbf{T}$
- $[\![\bigvee_{i=\mathbf{a}}^{\mathbf{b}} \varphi_i]\!]_{\mathcal{I}} = \mathbf{T} \Leftrightarrow \exists\alpha \in \mathbb{Z}$ *such that* $[\![\mathbf{a}]\!]_{\mathcal{I}} \leq_{\mathbb{Z}} \alpha \leq_{\mathbb{Z}} [\![\mathbf{b}]\!]_{\mathcal{I}}$ *and* $[\![\varphi_i]\!]_{\mathcal{I}[\alpha/i]} = \mathbf{T}$
- $[\![\bigwedge_{i=\mathbf{a}}^{\mathbf{b}} \varphi_i]\!]_{\mathcal{I}} = \mathbf{T} \Leftrightarrow \forall\alpha \in \mathbb{Z}\,,\, [\![\mathbf{a}]\!]_{\mathcal{I}} \leq_{\mathbb{Z}} \alpha \leq_{\mathbb{Z}} [\![\mathbf{b}]\!]_{\mathcal{I}}$ *implies* $[\![\varphi_i]\!]_{\mathcal{I}[\alpha/i]} = \mathbf{T}$

In the above definition, by $[\![\varphi_i]\!]_{\mathcal{I}[\alpha/i]}$ we mean every occurrence of i in φ_i is replaced by α. A propositional schema φ is *valid* (respectively *satisfiable*) iff for all (exists an interpretation) interpretations $\mathcal{I}$ s.t. $[\![\varphi]\!]_{\mathcal{I}} = T$. $\mathcal{I}$ is called a *model* of φ, written as $\mathcal{I} \models \varphi$. Two schemata φ, ψ are equivalent (written $\varphi \equiv \psi$) iff $\mathcal{I} \models \varphi \Leftrightarrow \mathcal{I} \models \psi$. φ and ψ are sat-equivalent (written $\varphi \equiv_S \psi$) iff φ and ψ are both satisfiable or both unsatisfiable (not necessarily by the same model).

Definition 2.2.5 (Unrolling Iterations [4]). *The following set S of rewrite rules is used to unroll the iterations of a given schematic formula φ:*

$$S = \begin{cases} \bigvee_{i=\mathbf{a}}^{\mathbf{b}} \psi \longrightarrow \bot & \mathbf{a}, \mathbf{b} \in \mathbb{Z} \text{ and } \mathbf{b} <_{\mathbb{Z}} \mathbf{a} \\ \bigwedge_{i=\mathbf{a}}^{\mathbf{b}} \psi \longrightarrow \top & \mathbf{a}, \mathbf{b} \in \mathbb{Z} \text{ and } \mathbf{b} <_{\mathbb{Z}} \mathbf{a} \\ \bigvee_{i=\mathbf{a}}^{\mathbf{b}} \psi \longrightarrow \left(\bigvee_{i=\mathbf{a}}^{\mathbf{b} \dot{-} 1} \psi \right) \vee \psi \left[\mathbf{b} / i \right] & \mathbf{a}, \mathbf{b} \in \mathbb{Z} \text{ and } \mathbf{a} \leq_{\mathbb{Z}} \mathbf{b} \\ \bigwedge_{i=\mathbf{a}}^{\mathbf{b}} \psi \longrightarrow \left(\bigwedge_{i=\mathbf{a}}^{\mathbf{b} \dot{-} 1} \psi \right) \wedge \psi \left[\mathbf{b} / i \right] & \mathbf{a}, \mathbf{b} \in \mathbb{Z} \text{ and } \mathbf{a} \leq_{\mathbb{Z}} \mathbf{b} \end{cases} \tag{2.2}$$

By $\leq_{\mathbb{Z}}$ we are referring to the standard ordering over the integers.

Definition 2.2.6 (Regular Schemata (as written in [4])). *A propositional schema ϕ is* regular *if it has a unique parameter n and if it is* flat *, of* bounded propagation *and* aligned *on $[\alpha, n - \beta]$:*

1) *A schema is* flat *if every $\Pi_{i=a}^{b}\psi$ occurring in the schema ψ does not contain an iteration, , where $\Pi \in \{\bigvee, \bigwedge\}$.*
2) *A schema is of* bounded propagation *if every atom that occurs in an iteration $\Pi_{i=a}^{b}\psi$ is of the form $P_{i+\gamma}$ for some $\gamma \in \mathbb{Z}$, where $\Pi \in \{\bigvee, \bigwedge\}$.*
3) *A schema is* aligned *on $[c, d]$ if all iterations occurring in the schema are of the form $\Pi_{i=c}^{d}\psi$, where $\Pi \in \{\bigvee, \bigwedge\}$.*

Example 2.2.3. *Consider the following schema:*

$$\varphi = p_0 \wedge \left(\bigwedge_{i=0}^{n} \neg p_i \vee p_{i+1} \right) \wedge \neg p_n \wedge 0 \leq n \tag{2.3}$$

φ is a regular schemata.

Basics of STAB and the ST procedure

We now overview the main ingredients of the ST decision procedure of the STAB framework introduced in Aravantinos et al. [4]. In this chapter, we only rely on the existence of the ST procedure and the propositional tableaux extension rules to define an extended decision procedure for our newly defined classes of schemata.

Definition 2.2.7 (Tableau). *A* tableau *is a tree T s.t. each node N occurring in T is labelled by a set of schemata written $\Phi_T(N)$.*

Definition 2.2.8 (Extension Rules). *The* extension rules *of the STAB procedure are as follows:*

Propositional Rules

- $\varphi \wedge \psi \Rightarrow \varphi, \psi$
- $\varphi \vee \psi \Rightarrow \varphi \mid \psi$

<u>*Iteration Rules*</u>

- $\bigwedge_{i=a}^{b} \varphi \Rightarrow a \leq b, \varphi\,[b/i] \wedge \bigwedge_{i=a}^{b-1} \varphi \;\Big|\; b < a$
- $\bigvee_{i=a}^{b} \varphi \Rightarrow a \leq b, \varphi\,[b/i] \vee \bigvee_{i=a}^{b-1} \varphi$

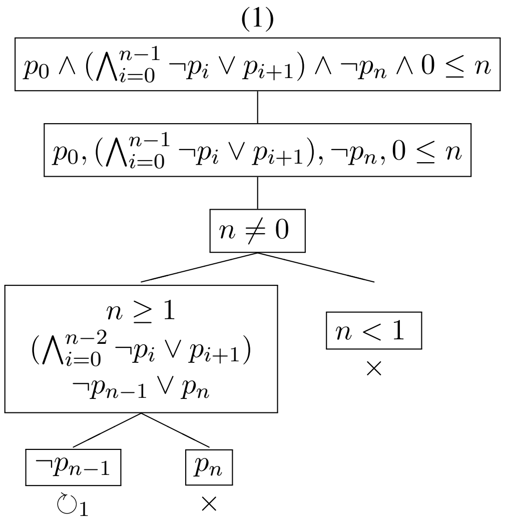
<u>*Closure Rule*</u>

- $p_a, \neg p_b \Rightarrow p_a, \neg p_b, a \neq b$

The way the STAB extension rules work is by extending currently constructed tableau with new leaves containing all the formulae of the prior node minus the formula φ on which the extension rule was applied. The parts of φ will be added to the leaves in accordance with the extension rule definitions. The symbol $|$ in the extension rules means that the constructed tableau branches when this rule is applied. The closure rule, rather than extending the constructed tableau, tells us that there is no need to extend the considered branch because it contains an unsatisfiable sub-branch .

Theorem 2.2.1. *There is a decision procedure for satisfiability testing of regular schemata (ST procedure) based on the STAB extension rules (Def. 2.2.8) and an additional rule to deal with looping , which terminates on every regular schema. The procedure is sound and complete for regular schemata.*

Example 2.2.4. *We provide an example of the ST procedure producing a closed tableau for the regular schema of example 2.2.3. Note that not every available formula is passed down the constructed tableau in the diagram.*


(1)
$p_0 \wedge (\bigwedge_{i=0}^{n-1} \neg p_i \vee p_{i+1}) \wedge \neg p_n \wedge 0 \leq n$
$p_0, (\bigwedge_{i=0}^{n-1} \neg p_i \vee p_{i+1}), \neg p_n, 0 \leq n$
$n \neq 0$
$n \geq 1$
$(\bigwedge_{i=0}^{n-2} \neg p_i \vee p_{i+1})$
$\neg p_{n-1} \vee p_n$
$n < 1$
$\times$
$\neg p_{n-1}$
$\circlearrowleft_1$
p_n
$\times$

The symbol $\circlearrowleft_1$ at the bottom of the left-most branch represents the looping rule. Essentially it means that the branch at the denoted point is the same as the branch at (1) (the top of the tableau), but for $n-1$ instead of n. We will not delve deeper into the theory behind the looping *rule as we only rely on the existence of such a rule for our procedure to work– for more details on the looping we refer to [4].*

Finally, we recall the following two concepts over a set Φ of schemata: the interval constraints ($IC(\Phi)$) and the conjunction of arithmetic formulae in Φ ($\Phi_{\mathcal{Z}}$). The formula $IC(\Phi)$ is the conjunction of the arithmetic formulae $min_{\phi}(i) \leq i \wedge i \leq max_{\phi}(i)$ for each $\phi \in \Phi$ and for each bound variable in ϕ. We assume that all bound variables are distinct in Φ and $min_{\phi}(i)$ is defined as the minimal value that can be assigned to the bound variable i, whereas $max_{\phi}(i)$ is the maximum value that can be assigned to the bound variable i.

Definition 2.2.9 (Pure Literal)**.** *A literal p_a (respectively $\neg p_a$) is* pure *in a set of schemata Φ iff for every occurrence of a literal $\neg p_b$ (respectively p_b) in Φ, the arithmetic formula $\Phi_{\mathcal{Z}} \wedge IC(\Phi) \wedge a = b$ is unsatisfiable.*

This definition will be modified to formalize the class of pure overlap schemata .

2.3 Linked Schemata

The class of linked schemata is an extension of regular schemata based on the following observation:

$$\left(\bigwedge_{i=1}^{n} p_i\right) \wedge \left(\bigvee_{j=n+1}^{m} \neg p_i\right) \equiv_S \left(\bigwedge_{i=1}^{n} p_i\right) \wedge \left(\bigvee_{j=1}^{m} \neg q_i\right). \tag{2.4}$$

Simply, we choose the interpretations such that $[\![p_{n+k}]\!]_{\mathcal{I}} = [\![q_k]\!]_{\mathcal{I}}$ for $k \in [1, m]$. By the finiteness of the language, we can separate the integers into two distinct parts, those greater than n and those less than n. Thus, the propositional variable p in the interval $[1, n]$ is invariant to the labelling of the propositional variable in the interval $[n+1, m]$. They can share the same name or not, the assignment will not influence the interpretations which model the schema. This observation is similar to the reduction from monadic predicate logic with monadic function symbols to monadic predicate logic without monadic function symbols, as outlined in Sec. 6.2 of "The Classical Decision Problem" [18].

Construction

The simplest way to understand the construction of the class of linked schemata is that any regular schema consists of atoms (specifically, ones not contained in iterations) and iterations. We will refer to these "parts" as the *principal objects* , denoted by $\mathcal{P}(\phi)$ of a schema ϕ. We consider sets Φ of regular schemata, such that the propositional symbols are distinct with regards to the regular schemata in the set, i.e. if $\phi, \psi \in \Phi$ and ϕ contains a propositional variable using the symbol p then ψ cannot contain propositional variables using this symbol. We can compute $\bigcup_{\phi \in \Phi} \mathcal{P}(\phi)$ without any propositional symbols occuring in two iterations indexed by different free parameters. Using this set of "parts" and the propositional connectives $\neg$, $\vee$, and $\wedge$ we can construct new formulae. The rest of this section will be focused on the formalization of this concept.

Definition 2.3.1. *Let $p \in \mathcal{P}$ be a propositional symbol and φ a formula schema, then $occ(p, \varphi) = 1$ iff p occurs in φ, otherwise it is $occ(p, \varphi) = 0$*

Definition 2.3.2 (principal Objects)**.** *Given a schema φ we can construct the set of* principal objects *$\mathcal{P}(\varphi)$ using the following inductive definition:*

- $\mathcal{P}(P_a) \Rightarrow \{P_a\}$
- $\mathcal{P}(\bigvee_{i=a}^{b} \psi) \Rightarrow \left\{\bigvee_{i=a}^{b} \psi\right\}$
- $\mathcal{P}(\bigwedge_{i=a}^{b} \psi) \Rightarrow \left\{\bigwedge_{i=a}^{b} \psi\right\}$
- $\mathcal{P}(\neg\psi) \Rightarrow \mathcal{P}(\psi)$
- $\mathcal{P}(\phi \vee \psi) \Rightarrow \mathcal{P}(\phi) \cup \mathcal{P}(\psi)$
- $\mathcal{P}(\phi \wedge \psi) \Rightarrow \mathcal{P}(\phi) \cup \mathcal{P}(\psi)$

One can consider $\mathcal{P}(\varphi)$ as a specially constructed set of formula schema.

Example 2.3.1. *Let use compute the set of principal objects of the following regular schema:*

$$\varphi \equiv (0 \leq n) \wedge P_0 \wedge \bigwedge_{i=1}^{n} (\neg P_{i-1} \vee P_i) \wedge \neg P_n \tag{2.5}$$

We get $\mathcal{P}(\varphi) = \{(0 \leq n), P_0, \bigwedge_{i=1}^{n} (\neg P_{i-1} \vee P_i), P_n\}$

We will abbreviate the set of propositional connectives used as $\mathcal{O} = \{\wedge, \vee, \neg\}$. By $\psi \in cl_{\mathcal{O}}(\Phi)$, we mean that ψ can be constructed using the set of formula schema Φ and the logical connective set $\mathcal{O}$.

Example 2.3.2. *Using the principal object set from Ex. 2.3.1 and the set of operators $\mathcal{O} = \{\wedge, \vee, \neg\}$, some of the formulae we can construct are:*

$$\psi_1 = (0 \leq n) \wedge P_0 \wedge \bigwedge_{i=1}^{n} (\neg P_{i-1} \vee P_i) \wedge \neg P_n \tag{2.6}$$

$$\psi_2 = \big((0{\leq}n){\wedge}P_0{\wedge}\textstyle\bigwedge_{i=1}^{n}(\neg P_{i-1}{\vee}P_i){\wedge}\neg P_n\big){\vee}\big((0{\leq}n){\wedge}\neg P_0{\wedge}\textstyle\bigwedge_{i=1}^{n}(\neg P_{i-1}{\vee}P_i){\wedge}P_n\big) \tag{2.7}$$

It is not necessary that the constructed formulae are valid, satisfiable, or unsatisfiable. One can check that both $\psi_1, \psi_2 \in cl_{\mathcal{O}}(\mathcal{P}(\varphi))$.

Lemma 2.3.1. *If φ is a regular schema, then all $\psi \in cl_{\mathcal{O}}(\mathcal{P}(\varphi))$ have the same aligned interval as φ.*

Proof. Assuming that φ has an aligned interval $[\alpha, n-\beta]$, then any, of its parts must have an aligned interval of at most $[\alpha, n-\beta]$ and are themselves regular schema. Thus, ψ is a boolean combination with the same aligned interval, implying that its aligned interval must be the same.
□ □

Using this simple result we will define the class of linked schemata, as follows.

Definition 2.3.3 (The class of Linked Schemata)**.** *Let us consider the class Λ of all finite sets Φ of regular schemata such that for all propositional symbols p, we have that $\left(\sum_{\phi\in\Phi} occ(p,\phi)\right)$ is either 1 or 0, we define the class* $\mathbf{LS}$ *of* linked schemata *as*

$$\mathbf{LS} = \bigcup_{\Phi\in\Lambda} cl_{\mathcal{O}}\left(\bigcup_{\phi\in\Phi} \mathcal{P}(\phi)\right)$$

Lemma 2.3.2. *If φ is a regular schema, then it is a linked schema.*

Proof. By definition 2.3.3, we can consider the set $\Phi = \{\varphi\}$, also, $\varphi \in cl_{\mathcal{O}}(\mathcal{P}(\varphi))$, and thus, $\varphi \in \mathbf{LS}$.
□ □

Theorem 2.3.1. *The class of regular schemata is contained but not equal to the class of linked schemata.*

Proof. We prove this by providing an example, see Ex. 2.3.3, of a linked schema which is not a regular schema.
□ □

Example 2.3.3. *Let us consider Φ containing the following three regular schemata. In what follows, we write $A \leftrightarrow B$ as an abbreviation for $(\neg A \vee B) \wedge (\neg B \vee A)$:*

$$\varphi_1 = \bigvee_{i=1}^{k} \neg P_i \wedge \neg \bigvee_{i=1}^{k} \neg P_i \tag{2.8a}$$

$$\varphi_2 = \bigvee_{i=1}^{m} Q_i \wedge \bigvee_{i=1}^{m} R_i \wedge \bigwedge_{i=1}^{m} Q_i \leftrightarrow R_i \tag{2.8b}$$

$$\varphi_3 = \bigwedge_{i=1}^{n} M_i \tag{2.8c}$$

We can construct the following $\mathbf{LS}$ *formula using Φ:*

$$\begin{gathered}\left(\left(\bigvee_{i=1}^{k} \neg P_i \rightarrow \bigvee_{i=1}^{m} Q_i\right) \wedge \left(\bigvee_{i=1}^{m} R_i \rightarrow \bigwedge_{i=1}^{n} M_i\right) \wedge \bigwedge_{i=1}^{m} (Q_i \leftrightarrow R_i)\right) \rightarrow \\ \left(\bigvee_{i=1}^{k} \neg P_i \rightarrow \bigwedge_{i=1}^{n} M_i\right).\end{gathered} \tag{2.9}$$

Formula 2.9 gives a formalization of the composition of certain boolean functions when one function's range has the same number of bits as another function's domain. This formula is obviously not regular, but it is linked. This concludes the proof of Thm. 2.3.1.

2.4 Pure Overlap Schemata

In this section we show how one can weaken the restriction that propositional symbols occur indexed by only one parameter. Consider the following formula schema ψ:

$$0 \leq n \wedge \left(\bigwedge_{i=0}^{n} p_i \right) \vee \left(\bigwedge_{i=0}^{m} \neg p_i \right) \wedge 0 \leq m \tag{2.10}$$

It is not a linked schema because p occurs indexed by two different parameters, however, using the tableaux extension rule for propositional $\vee$ we see that the occurrences are handled by two different branches, thus each parameter can be handled separately. It is also important to note that

$$0 \leq n \wedge \left(\bigwedge_{i=0}^{n} p_i \right) \vee \left(\bigwedge_{i=0}^{m} \neg q_i \right) \wedge 0 \leq m \tag{2.11}$$

Replacing p with q in Eqn. 2.10 results in Eqn. 2.11, which changes the formula from valid to satisfiable (only when $0 \leq n, m$). Thus, we cannot reduce this formula to linked schemata without changing its semantic properties. To deal with this problem we introduce relatively pure literals, based on the observation that if the negation of a literal occurs in the same branch indexed by a different parameter then the literal must not be of arithmetic importance. We then show that relatively pure literals can be dropped without effecting satisfiability of the considered pure overlap schemata.

Construction

We first introduce the notion of relatively pure literals and detail the construction of pure overlap schemata.

Definition 2.4.1 (Iteration Invariant DNF (IIDNF))**.** *The* Iteration Invariant disjunctive normal *form of a linked schema is a schema of the form:*

$$(\varphi_{1,1} \wedge \cdots \wedge \varphi_{1,n_1}) \vee \cdots \vee (\varphi_{m,1} \wedge \cdots \wedge \varphi_{m,n_m})$$

where $m, n_1, \cdots, n_m \in \mathbb{N}$ (note they are not free parameters, but rather meta variables) and $\varphi_{i,j}$ is either an iteration, an atom, or negated atom. We will refer to the formula $(\varphi_{i,1} \wedge \cdots \wedge \varphi_{i,n_i})$ as clauses C_i for $i \in [1, m]$, That is, given a formula φ in IIDNF we will write $C_i \in \varphi$ as the i^{th} clause of φ.

Lemma 2.4.1. *Given a set of regular schemata Φ, for all $\psi \in cl_{\mathcal{O}}\left(\bigcup_{\phi \in \Phi} \mathcal{P}(\phi)\right)$ there exists an IIDNF of ψ.*

Proof. Since, iterations are not unfolded in the creation of an IIDNF form of ψ, the problem reduces to showing that all propositional formulae have a DNF form, which is a well known result. Also, it is possible to put a regular schemata into *Negation Normal Form* (NNF) because negation can be passed over iterations, i.e $\neg \bigwedge_{i=a}^{b} \phi_i \equiv_{\models} \bigvee_{i=a}^{b} \neg\phi_i$ and $\neg \bigvee_{i=a}^{b} \phi_i \equiv_{\models} \bigwedge_{i=a}^{b} \neg\phi_i$. □

Definition 2.4.2 (Relatively Pure Literal)**.** *Given a set of regular schemata Φ, let $\psi \in cl_{\mathcal{O}}\left(\bigcup_{\phi\in\Phi} \mathcal{P}(\phi)\right)$ and ψ' be the IIDNF of ψ. A literal p_a ($\neg p_a$) is* relatively pure *in ψ iff for every clause $C \in \psi'$ and for any two distinct regular schemata $\varphi_1, \varphi_2 \in \Phi$ used to construct ψ, where $p_a \in C$ ($\neg p_a \in C$), $\neg p_b \in C$ ($p_b \in C$), $p_a \in \varphi_1$ ($\neg p_a \in \varphi_1$) and $\neg p_b \in \varphi_2$ ($p_b \in \varphi_2$), the arithmetic formula $\Xi_{\mathcal{Z}} \wedge \left(\bigwedge_{i\in\mathcal{V}_b^C} min_{\Xi}(i) \leq i \leq \max_{\Xi}(i)\right) \wedge a = b$ is unsatisfiable, where $\Xi = \mathcal{P}(C)$, $\mathcal{V}_b^C$ is the set of bound variables in C and either a or b or both must contain at least one of the bound variables or free parameter.*

Example 2.4.1. *Consider the schemata:*

$$\neg(5 < n) \wedge \left(\bigwedge_{i=0}^{n} p_i\right) \wedge \left(\bigwedge_{j=6}^{m} \neg p_i\right) \wedge 0 \leq m \tag{2.12}$$

The literal p_i ($\neg p_i$) is relatively pure in this example.

We will refer to a schema as *relatively pure* if all the literals in the schema are either relatively pure or in the IIDNF of the schema they only occur in clauses being indexed by a single parameter. The non-IIDNF form of a relatively pure schema is also relatively pure. Given a set of regular schemata Φ, let $cl_{\mathcal{O}}^{rp}(\Phi)$ be the set of all schema which can be constructed using the logical connectives $\mathcal{O}$ such that they are relatively pure.

Definition 2.4.3 (The class of Pure Overlap Schemata)**.** *Let us consider the class Λ of all finite sets Φ of regular schemata. We define the class of* pure overlap schemata *as*

$$\mathbf{POS} = \bigcup_{\Phi\in\Lambda} cl_{\mathcal{O}}^{rp}\left(\bigcup_{\phi\in\Phi} \mathcal{P}(\phi)\right)$$

It should be noted that even though the definition of relatively pure literals uses the IIDNF of a positional schema it is not the case that members of $\mathbf{POS}$ must be in IIDNF.

Example 2.4.2. *Both Ex. 2.4.1 and Eqn. 2.10 are in the class of pure overlap schemata.*

Lemma 2.4.2. *If φ is a linked schema, then it is a pure overlap schema.*

Proof. A linked schema is a pure overlap schema where each propositional variable is indexed by only one parameter.
□ □

Theorem 2.4.1. *The class of linked schemata is contained but not equal to the class of pure overlap schemata.*

Proof. Eqn. 2.10 is a pure overlap schema but not a linked schema.
□ □

2.5 A Decision procedure for POS

We now introduce a decision procedure for the class POS of schemata, by using and extending results of [4], as follows.

Algorithm 2.5.1 (ST$^{\mathbf{POS}}$ Procedure)**.** *Given a schema $\varphi \in \mathbf{POS}$ in negation normal form . The following algorithm, called the $ST^{\mathbf{POS}}$ procedure, decides the satisfiability of φ:*

1) *Apply STAB propositional extension rules with highest priority until no more can be applied. This results in m sets of atoms and iterations referred to as $B_1, \ldots, B_m$.*

2) *For each B_i, we separate B_i into n (the number of parameters in B_i) sub-branches $B_{(i,1)}, \cdots B_{(i,n)}$, where each $B_{(i,j)}$ contains iterations and atoms indexed by a single parameter. Atoms without a free parameter in the indices can be added to every $B_{(i,j)}$. We will mark such a sub-branching with $\otimes_n$ where n is the number of parameters on the branch.*

3) *Run the ST procedure on the sub-branch $B_{(i,j)}$.*

4) *For any branch B_i, if one of its sub-branches $B_{(i,j)}$ has a closed tableau after following the ST procedure, then the branch B_i is closed.*

Let us make the following observation about the $ST^{\mathbf{POS}}$ decision procedure. When it comes to constructing the interpretation for a formula in $\mathbf{POS}$ we specifically defined the class such that the procedure to construct the model would be precisely the procedure used for regular schemata, except the number of possible models would increase. For linked schemata this is obvious, the propositional symbols are distinct in every sub-branch. However, for pure overlap schemata two distinct sub-branches (of the same branch) can contain the same propositional symbol, but by Def. 2.4.2 the occurrences are distinct from each other arithmetically if one occurrence is negated and the other occurrence is not. Thus, when a propositional symbol occurs on two distinct sub-branches and the two occurrences are not arithmetically distinct, the two occurrences must be of the same polarity. In this case when one sub-branch forces the propositional variable using the positional symbol to be true (false in the case of a negated literal), the other sub-branches will also interpret this literal as true. In some sense one can consider it as a local tautology which can be removed from consideration when constructing the model.

Theorem 2.5.1. *The $ST^{\mathbf{POS}}$ procedure terminates for* $\mathbf{POS}$.

Proof. The key to the termination is that we only need to decompose the members of $\mathbf{POS}$ using the procedure outlined above. This decomposition process always terminates being that we are, up to this point, only applying propositional tableaux extension rules. When the formulae are completely decomposed we use the ST procedure on each sub-branch. The procedure is known to terminate for regular schemata [4] and each of the sub-branches is regular. □ □

In regards to the soundness and completeness of $ST^{\mathbf{POS}}$ the procedure, it was shown that STAB is sound and complete for all propositional schemata [4] (Sec. 5.4). The propositional schemata we introduce in this chapter are constructed using exactly the same language as in the work by Aravantinos et al. [4] Our extension of STAB with the sub-branching rule does not change the soundness and completeness results being that the sub-branching rule, rather than being an additional tableaux rule, is more a method to enforce termination. It essentially states that instead of considering the given branch as a whole we consider it in parts using the same tableaux rules introduced for STAB in prior work.

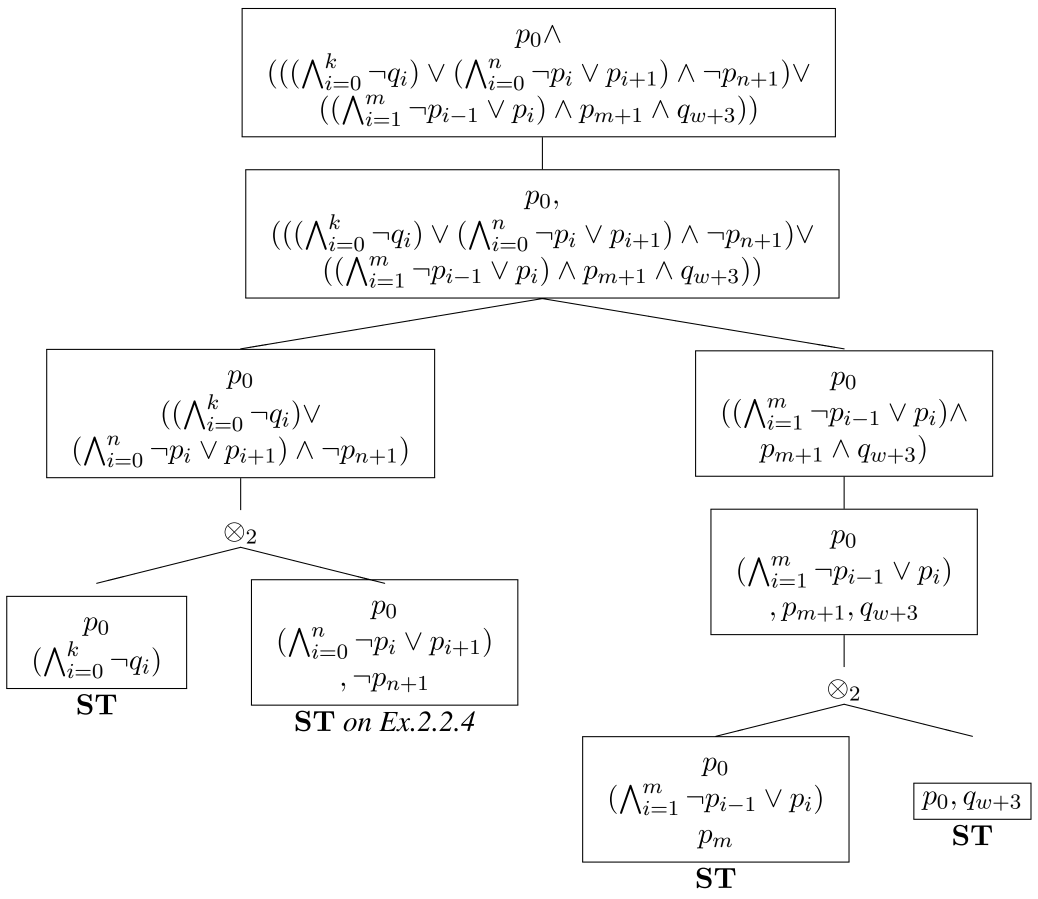
Theorem 2.5.2. *The $ST^{\mathbf{POS}}$ decision procedure is sound and complete for all propositional schemata $\psi \in \mathbf{POS}$.*

Proof. It is quite obvious that all the standard tableaux rules are sound and if we use the sub-branching tableaux rule such that $n = 1$, i.e. $\otimes_1$, Then the above procedure is the same as the procedure from Aravantinos et al. [4] which was shown to be sound and complete. Thus we can prove that this method is sound and complete by induction on the variable n is the definition of the sub-branching rule $\otimes_n$. We assuming for the induction hypothesis that the method is sound and complete for sub-branching $\otimes_m$, where $m \leq n$ and show that it holds for $\otimes_{n+1}$. When a formula in $\mathbf{POS}$ requires the $ST^{\mathbf{POS}}$ procedure to use a sub-branching operator $\otimes_{n+1}$ this implies that that there is a branch which is no longer propositionally reducible and has $n + 1$ free parameters. By the definition of $\mathbf{POS}$ and relatively pure literals, Def. 2.4.3 & 2.4.2, we know that we don't need to use literals from two different parameters mutually to decide the satisfiability of the given formula, thus, instead of considering $\otimes_{n+1}$ we could consider two sub-branching rule applications, first $\otimes_1$ separating one parameter from the rest and then $\otimes_n$ separating the rest. These two steps hold by the induction hypothesis and prove the theorem. $\square$

Example 2.5.1. *We conclude this section by illustrating our $ST^{\mathbf{POS}}$ decision procedure on the following formula ψ:*

$$\begin{gathered} p_0 \wedge \Big(\Big(\Big(\bigwedge_{i=0}^{k} \neg q_i \Big) \vee \Big(\bigwedge_{i=0}^{n} \neg p_i \vee p_{i+1} \Big) \wedge \neg p_{n+1} \Big) \vee \\ \Big(\Big(\bigwedge_{i=0}^{m} \neg p_{i-1} \vee p_i \Big) \wedge p_{m+1} \wedge \neg q_{w+3} \Big) \Big) \end{gathered}$$

Applying $ST^{\mathbf{POS}}$ on the above formula, we obtain the following branching tree (corresponding to the run of $ST^{\mathbf{POS}}$):


$p_0 \wedge$
$(((\bigwedge_{i=0}^{k} \neg q_i) \vee (\bigwedge_{i=0}^{n} \neg p_i \vee p_{i+1}) \wedge \neg p_{n+1}) \vee$
$((\bigwedge_{i=1}^{m} \neg p_{i-1} \vee p_i) \wedge p_{m+1} \wedge q_{w+3}))$
$p_0,$
$(((\bigwedge_{i=0}^{k} \neg q_i) \vee (\bigwedge_{i=0}^{n} \neg p_i \vee p_{i+1}) \wedge \neg p_{n+1}) \vee$
$((\bigwedge_{i=1}^{m} \neg p_{i-1} \vee p_i) \wedge p_{m+1} \wedge q_{w+3}))$
p_0
$((\bigwedge_{i=0}^{k} \neg q_i) \vee$
$(\bigwedge_{i=0}^{n} \neg p_i \vee p_{i+1}) \wedge \neg p_{n+1})$
p_0
$((\bigwedge_{i=1}^{m} \neg p_{i-1} \vee p_i) \wedge$
$p_{m+1} \wedge q_{w+3})$
$\otimes_2$
p_0
$(\bigwedge_{i=1}^{m} \neg p_{i-1} \vee p_i)$
$, p_{m+1}, q_{w+3}$
p_0
$(\bigwedge_{i=0}^{k} \neg q_i)$
ST
p_0
$(\bigwedge_{i=0}^{n} \neg p_i \vee p_{i+1})$
$, \neg p_{n+1}$
ST *on Ex.2.2.4*
$\otimes_2$
p_0
$(\bigwedge_{i=1}^{m} \neg p_{i-1} \vee p_i)$
p_m
ST
p_0, q_{w+3}
ST

Interesting result of this derivation is that the assignment to w influences the interpretation modelling the formula. If an interpretation $\mathcal{I}$ assigns $q_{-1} = T$,$q_0 = F$,$p_0 = T$,$p_{-1} = F$, $p_5 = F$,$w \leftarrow -4$, $n \leftarrow -2$, $k \leftarrow 0$, and $m \leftarrow 5$ then $\mathcal{I} \models \psi$. But if $\mathcal{I}$ assigns to $n \leftarrow 0$ keeping the same propositional variable assignments, then $\mathcal{I} \not\models \psi$.

2.6 Conclusion and future work

In this work we have shown that the ST procedure of Aravantinos et al. [4] can be extended to handle more expressive classes of schemata which allow for restricted use of multiple free parameters. The two classes shown, though their construction is awkward, are simple to conceptually understand and work with. Also, neither requires the heavy machinery of normalized clause sets, nor do the classes require a conversion of the schemata into a clausal normal form. Also, the introduced decision procedure $ST^{\mathbf{POS}}$ is sound, complete, and terminates for all propositional schemata in the class $\mathbf{POS}$. Though an advantage normalized clause sets have over both of the introduced classes of schemata is that they can handle propositional variables being indexed by multiple free parameters without restriction. This is one of the significant advantages to separating the propositional part from the equational part, and using a levelled resolution calculus. When it is not required to have unrestricted usage of the propositional variables it suffices to use STAB.

This also has the added value of compression being that it is possible for clausal form to result in an exponential increase in the size of the formula.

As for future work, further increase in expressivity by relaxing the purity constraint does not seem feasible as this would require two parameters to be active in the same branch. This is when the undecidability result for propositional schemata [4] stops us in our tracks. However, investigating how the new classes outlined here can interact with the class of *regular nested schemata* [3] could lead to new expressivity results. In particular, we are interested in the relationship between alternation-free μ-calculus [19] and such a class of schemata. Also, Aravantinos et al. [5] investigated the relationship between LTL and regular schemata. Being that pure overlap schemata are a super class of regular schemata it is quite possible that a more expressive temporal logic is related to pure overlap schemata or linked schemata. In either case this work has broaden the scope of application of propositional schemata.

CHAPTER 3

Introduction to the CERES method of cut Elimination

Since the publication of Gentzen's *Hauptsatz* [34], the principal method of cut elimination in sequent calculi has been reductive cut-elimination, i.e. reducing the complexity of the cuts in the sequent calculus proof and pushing the cuts closer to the axioms. In the book, [15] Gentzen's method of cut elimination is outlined and discussed as well as the more recent method developed by Schütte and Tait . However, the method of cut-elimination which is of central focus in the book "Methods of Cut Elimination" is cut-elimination by resolution introduced by M. Baaz and A. Leitsch [14], the CERES method of cut-elimination . Unlike the previously mentioned methods of cut elimination, the CERES method changes the problem of cut-elimination from a proof transformation procedure to the computational problem of showing that a clause set defining the cut structure of a sequent calculus proof is unsatisfiable using resolution. The resolution refutation which is produced can then be used as a proof skeleton and a sequent calculus proof in what is called an Atomic Cut Normal Form (ACNF) can be produced. A step in a standard resolution refutation can be considered an atomic cut in a sequent calculus proof given that the resolution refutation is ground. This transition from cut-elimination being a purely proof transformation method, to cut-elimination as a computational problem, allowed investigation of computational complexity of the problem of cut-elimination through optimizations and refinements originally developed for resolution theorem proving. In a sense, it opened the doors to considering more of the spectrum of cut-free versions of a sequent calculus proof proving a specific end-sequent, more specifically the cut-free versions which can be constructed from resolution refutations of the characteristic clause set. It is shown in the book "Methods of Cut Elimination", specifically in section 6.10, what are the benefits of using the proof skeletons derived from the refutations of the characteristic clause set, namely, shrinking the size of the cut-free proof in some cases. Essentially, there are sequences of proofs whose length grows non-elementarily when using the Gentzen method, but grows elementarily when considering a specific sequence of refutations using the CERES method.

The rest of this chapter will be devoted to a description of the CERES method as well as an example of how the method works. Though, the rest of this dissertation concerns itself with an extension of the CERES method to be discussed in Ch. 4. One should consider this chapter as necessary background needed for understanding Ch. 4.

3.1 The First Order LK Sequent Calculus

In this section we introduce the calculus used in Gentzen's *Hauptsatz* [34] as well as the calculus referred to in literature concerning cut-elimination [54]. As a side note, Gentzen proved that the calculus is consistent by showing that every instance of the cut rule can be removed (cut-elimination) and thus falsum ($\bot$) cannot be derived (of course the proof is much more complex than that). The original usage of the concept of cut-elimination, as one can see, was as a step in a proof rather than independent procedure in and of itself. The following outline of the first-order sequent calculus is based on the description provided in [54]. The calculus used for the majority of this thesis is based on the sequent calculus which will be outlined in this section, with the addition of some specific equational theories. There are a few extensions which will be discussed in Ch. 4 which allow schematic definitions of proofs, terms, and predicates. The core structure will remain the same. We will first define first order constructions and then use these constructions to define the calculus.

Definition 3.1.1 (From [54]). *A first order (formal) language consist of the following symbols:*

1) ***Constants:***

   1.1) *Individual constant symbols ,* $c_0, c_1, \cdots c_j$ *where* $j \in \mathbb{N}$.

   1.2) *Function symbols with* $i \in \mathbb{N}$ *arguments,* $f^i_0, f^i_1, \cdots f^i_j$ *where* $j \in \mathbb{N}$.

   1.3) *Predicate symbols with* $i \in \mathbb{N}$ *arguments,* $P^i_0, P^i_1, \cdots P^i_j$ *where* $j \in \mathbb{N}$.

2) ***Variables:***

   2.1) *Free variable symbols,* $\alpha_0, \alpha_1, \cdots \alpha_j$ *where* $j \in \mathbb{N}$.

   2.2) *Bound variable symbols,* $x_0, x_1, \cdots x_j$ *where* $j \in \mathbb{N}$.

3) ***Logical symbols:***

   3.1) $\vee$ *(OR) ,* $\wedge$ *(AND) ,* $\neg$ *(NEGATION),* $\rightarrow$ *(IMPLICATION),* $\forall$ *(UNIVERSAL), and* $\exists$ *(EXISTENTIAL)*

4) ***Auxiliary symbols:***

   4.1) *(,) (parentheses) and , (comma)*

For the entirety of this dissertation we will assume the each of these sets, i.e. free variable, constant symbols, etc. , is countably large. Also, that there is an ordering on the members of these sets which is the canonical ordering of the natural numbers. Now we will discuss term and formula construction.

Definition 3.1.2 (From [54]). *Terms are defined inductively as follows:*

1) Every individual constant is a term.

2) Every free variable is a term.

3) If f^i is a function symbol with arity i and $t_1, \cdots, t_i$ are terms, then $f^i(t_1, \cdots, t_i)$ is a term.

Definition 3.1.3 (From [54]). *Let P^i be a predicate symbol with arity i and $t_1, \cdots, t_i$ are terms, then $P^i(t_1, \cdots, t_i)$ is an atomic formula. Formula are defined inductively as follows:*

1) Every atomic formula is a term.

2) If A and B are formulae, then $A \vee B$, $A \wedge B$, $\neg A$, and $A \rightarrow B$ are formulae.

3) If A is a formula, α a free variable, and x a bound variable not occurring in A, Then $\forall x A'$ and $\exists A'$ are formulae, where A' is the same as A except every occurrence of α in A is replaced by x.

Definition 3.1.4 (From [54]). *Let Γ and Δ represent finite sets of formulae, then we define a* sequent *as $\Gamma \vdash \Delta$. The antecedent of the sequent $\Gamma \vdash \Delta$ is Γ and the consequent is Δ.*

Definition 3.1.5 (LK-calculus [54]).

1) Structural Rules:

1.1) Weakening:

$$\frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta}\, w:l \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A}\, w:r$$

1.2) Contraction:

$$\frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta}\, c:l \qquad \frac{\Gamma \vdash \Delta, A, A}{\Gamma \vdash \Delta, A}\, c:r$$

1.3) Cut:

$$\frac{\Gamma, A \vdash \Delta \qquad \Gamma' \vdash \Delta', A}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\, cut$$

2) Logical Rules:

2.1) Negation:

$$\frac{\Gamma \vdash \Delta, A}{\Gamma, \neg A \vdash \Delta}\, \neg:l \qquad \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash \Delta, \neg A}\, \neg:r$$

2.2) Conjunction:

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}\, \wedge:l \qquad \frac{\Gamma, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}\, \wedge:l \qquad \frac{\Gamma \vdash \Delta, A \qquad \Gamma' \vdash \Delta', B}{\Gamma, \Gamma' \vdash \Delta, \Delta', A \wedge B}\, \wedge:r$$

2.3) Disjunction:

$$\frac{\Gamma, A \vdash \Delta \qquad \Gamma', B \vdash \Delta'}{\Gamma, \Gamma', A \vee B \vdash \Delta, \Delta'} \vee : l \quad \frac{\Gamma \vdash \Delta, A}{\Gamma \vdash \Delta, A \vee B} \vee : r \qquad \frac{\Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \vee B} \vee : r$$

2.4) Implication:

$$\frac{\Gamma, B \vdash \Delta \qquad \Gamma' \vdash \Delta', A}{\Gamma, \Gamma', A \rightarrow B \vdash \Delta, \Delta'} \rightarrow: l \qquad \frac{\Gamma, A \vdash \Delta, B}{\Gamma \vdash \Delta, A \rightarrow B} \rightarrow: r$$

3) Quantifier Rules:

3.1) Universal Quantification:

$$\frac{\Gamma, A(t) \vdash \Delta}{\Gamma, \forall x A(x) \vdash \Delta} \forall : l \qquad \frac{\Gamma, \vdash \Delta, A(\alpha)}{\Gamma \vdash \Delta, \forall x A(x)} \forall : r$$

3.2) Existential Quantification:

$$\frac{\Gamma, A(\alpha) \vdash \Delta}{\Gamma, \exists x A(x) \vdash \Delta} \forall : l \qquad \frac{\Gamma, \vdash \Delta, A(t)}{\Gamma \vdash \Delta, \exists x A(x)} \forall : r$$

In the quantification rules (3.1 and 3.2) t is an arbitrary term. In the $\forall : r$ and $\exists : l$ rules α is an eigenvariable, i.e. it does not occur in any formula in the sequent below the inference rule bounding it. The variable used to bound the eigenvariable or the term cannot occur in the auxiliary formula. The auxiliary formulae are the formulae which are used by the inference rule. The main formula is the formula resulting from the inference rule.

Definition 3.1.6. *An LK-proof is a tree of inference rules with the following two properties:*

- *The topmost sequents are of the form $A \vdash A$.*
- *Every sequent in an LK-proof is an upper sequent of a sequent which is also part of the same LK-proof except the lowest sequent.*

Definition 3.1.7. *An LK-proof is called* regular *if at any arbitrary point in the proof tree eigenvariables used by quantifier rules on different branches are different.*

Later on in this dissertation we will break the first point by allowing LK-proofs modulo a theory. This is all we will need to know about the LK calculus in order to explain the CERES method. We end this section with an example of an LK proof which is derived from the ECS-schema when the free parameter is instantiated by 1. This proof is modulo the theory found at the top of the proof tree (the leaves). More is explained in Ch. 5 and the schema can be found in Appendix A.

Example 3.1.1. *In the following LK-proof we will use $(1), (2), (3), (4)$ to represent places where two proof parts connect. This notation will be used throughout this dissertation.*

$$\dfrac{\dfrac{\begin{array}{c}0 \sim f(\alpha),\\ 0 \sim f(g(\alpha)) \vdash \\ f(\alpha) \sim f(g(\alpha))\end{array} \qquad \alpha \leq g(\alpha) \vdash \alpha \leq g(\alpha)}{\begin{array}{c}0 \sim f(\alpha),\\ ((\alpha \leq g(\alpha)) \rightarrow 0 \sim f(g(\alpha))),\\ \alpha \leq g(\alpha) \vdash \\ f(\alpha) \sim f(g(\alpha))\end{array}} \rightarrow: l \qquad \vdash \alpha \leq \alpha}{\begin{array}{c}(\alpha \leq \alpha) \rightarrow 0 \sim f(\alpha),\\ ((\alpha \leq g(\alpha)) \rightarrow 0 \sim f(g(\alpha))),\\ \alpha \leq g(\alpha) \vdash \\ f(\alpha) \sim f(g(\alpha))\\ (1)\end{array}} \rightarrow: l$$

$$
\dfrac{
\dfrac{
\begin{array}{c}
(1)\\
(\alpha \leq \alpha) \rightarrow 0 \sim f(\alpha),\\
((\alpha \leq g(\alpha)) \rightarrow 0 \sim f(g(\alpha)))\,,\\
\alpha \leq g(\alpha) \vdash\\
f(\alpha) \sim f(g(\alpha))
\end{array}
\qquad
f(g(\alpha)) \prec 0 \vdash
}{
\begin{array}{c}
(\alpha \leq \alpha) \rightarrow 0 \sim f(\alpha),\\
((\alpha \leq g(\alpha)) \rightarrow 0 \sim f(g(\alpha)))\\
\vee f(g(\alpha)) \prec 0,\\
\alpha \leq g(\alpha) \vdash\\
f(\alpha) \sim f(g(\alpha))
\end{array}
}\ \vee : l
\qquad
f(\alpha) \prec 0 \vdash
}{
\begin{array}{c}
((\alpha \leq \alpha) \rightarrow 0 \sim f(\alpha))\\
\vee f(\alpha) \prec 0,\\
((\alpha \leq g(\alpha)) \rightarrow 0 \sim f(g(\alpha)))\\
\vee f(g(\alpha)) \prec 0,\\
\alpha \leq g(\alpha) \vdash\\
f(\alpha) \sim f(g(\alpha))
\end{array}
}\ \vee : l
$$

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\begin{array}{c}
((\alpha \leq \alpha) \rightarrow 0 \sim f(\alpha))\\
\vee f(\alpha) \prec 0,\\
((\alpha \leq g(\alpha)) \rightarrow 0 \sim f(g(\alpha)))\\
\vee f(g(\alpha)) \prec 0,\\
\alpha \leq g(\alpha) \vdash\\
f(\alpha) \sim f(g(\alpha))
\end{array}
}{
\begin{array}{c}
((\alpha \leq \alpha) \rightarrow 0 \sim f(\alpha))\\
\vee f(\alpha) \prec 0,\\
((\alpha \leq g(\alpha)) \rightarrow 0 \sim f(g(\alpha)))\\
\vee f(g(\alpha)) \prec 0 \vdash\\
\alpha \leq g(\alpha) \rightarrow f(\alpha) \sim f(g(\alpha))
\end{array}
}\ \rightarrow: r
}{
\begin{array}{c}
((\alpha \leq \alpha) \rightarrow 0 \sim f(\alpha))\\
\vee f(\alpha) \prec 0,\\
((\alpha \leq g(\alpha)) \rightarrow 0 \sim f(g(\alpha)))\\
\vee f(g(\alpha)) \prec 0 \vdash\\
\exists x(x \leq g(x) \rightarrow f(x) \sim f(g(x)))
\end{array}
}\ \exists : r
}{
\begin{array}{c}
((\alpha \leq \alpha) \rightarrow 0 \sim f(\alpha))\\
\vee f(\alpha) \prec 0,\\
\forall y\,(((\alpha \leq y) \rightarrow 0 \sim f(y))\\
\vee f(y) \prec 0) \vdash\\
\exists x(x \leq g(x) \rightarrow f(x) \sim f(g(x)))
\end{array}
}\ \forall : l
}{
\begin{array}{c}
\forall y\,(((\alpha \leq y) \rightarrow 0 \sim f(y))\\
\vee f(y) \prec 0)\,,\\
\forall y\,(((\alpha \leq y) \rightarrow 0 \sim f(y))\\
\vee f(y) \prec 0) \vdash\\
\exists x(x \leq g(x) \rightarrow f(x) \sim f(g(x)))
\end{array}
}\ \forall : l
}{
\begin{array}{c}
\forall y\,(((\alpha \leq y) \rightarrow 0 \sim f(y))\\
\vee f(y) \prec 0) \vdash\\
\exists x(x \leq g(x) \rightarrow f(x) \sim f(g(x)))
\end{array}
}\ c : l
}{
\begin{array}{c}
\exists x \forall y\,(((x \leq y) \rightarrow 0 \sim f(y))\\
\vee f(y) \prec 0) \vdash\\
\exists x(x \leq g(x) \rightarrow f(x) \sim f(g(x)))\\
(4)
\end{array}
}\ \exists : l
$$

$$\dfrac{\dfrac{\dfrac{\dfrac{0 \sim f(\alpha) \vdash 0 \sim f(\alpha)}{f(\alpha) \preceq f(\alpha), 0 \sim f(\alpha) \vdash 0 \sim f(\alpha)}\, w : l}{f(\alpha) \preceq f(\alpha), 0 \leq \alpha, 0 \sim f(\alpha) \vdash 0 \sim f(\alpha)}\, w : l}{0 \sim f(\alpha), f(\alpha) \preceq f(\alpha), 0 \leq \alpha \vdash 0 \sim f(\alpha), f(\alpha) \prec 0}\, w : r \qquad \vdash \alpha \leq \alpha}{0 \sim f(\alpha), \alpha \leq \alpha \to f(\alpha) \preceq f(\alpha), 0 \leq \alpha \vdash 0 \sim f(\alpha), f(\alpha) \prec 0} \to: l$$

(1)

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c}(1)\\ 0 \sim f(\alpha),\\ \alpha \leq \alpha \to f(\alpha) \preceq f(\alpha), 0 \leq \alpha \vdash\\ 0 \sim f(\alpha), f(\alpha) \prec 0\end{array}}{\begin{array}{c}0 \sim f(\alpha),\\ \forall y\Big(\alpha \leq y \to f(y) \preceq f(\alpha)\Big), 0 \leq \alpha \vdash\\ 0 \sim f(\alpha), f(\alpha) \prec 0\end{array}}\;\forall : l}{\begin{array}{c}0 \sim f(\alpha),\\ \forall x\forall y\Big(x \leq y \to f(y) \preceq f(x)\Big), 0 \leq \alpha \vdash\\ 0 \sim f(\alpha), f(\alpha) \prec 0\end{array}}\;\forall : l}{\begin{array}{c}\forall x(0 \sim f(x)),\\ \forall x\forall y\Big(x \leq y \to f(y) \preceq f(x)\Big), 0 \leq \alpha \vdash\\ 0 \sim f(\alpha), f(\alpha) \prec 0\end{array}}\;\forall : l}{\begin{array}{c}\forall x(0 \sim f(x)),\\ \forall x\forall y\Big(x \leq y \to f(y) \preceq f(x)\Big) \vdash\\ ((0 \leq \alpha) \to 0 \sim f(\alpha))\\ , f(\alpha) \prec 0\end{array}}\;\to : r}{\begin{array}{c}\forall x(0 \sim f(x)),\\ \forall x\forall y\Big(x \leq y \to f(y) \preceq f(x)\Big) \vdash\\ ((0 \leq \alpha) \to 0 \sim f(\alpha))\\ \lor f(\alpha) \prec 0\end{array}}\;\lor : r}{\begin{array}{c}\forall x(0 \sim f(x)),\\ \forall x\forall y\Big(x \leq y \to f(y) \preceq f(x)\Big) \vdash\\ \forall y\,(((0 \leq y) \to 0 \sim f(y))\\ \lor f(y) \prec 0)\end{array}}\;\forall : r}{\begin{array}{c}\forall x(0 \sim f(x)),\\ \forall x\forall y\Big(x \leq y \to f(y) \preceq f(x)\Big) \vdash\\ \exists x\forall y\,(((x \leq y) \to 0 \sim f(y))\\ \lor f(y) \prec 0)\\ (3)\end{array}}\;\exists : r
$$

$$
\dfrac{\begin{array}{c}(3)\\ \forall x(0 \sim f(x)),\\ \forall x\forall y\Big(x \leq y \to f(y) \preceq f(x)\Big) \vdash\\ \exists x\forall y\,(((x \leq y) \to 0 \sim f(y))\\ \lor f(y) \prec 0)\end{array}\qquad\qquad \begin{array}{c}(4)\\ \exists x\forall y\,(((x \leq y) \to 0 \sim f(y))\\ \lor f(y) \prec 0) \vdash\\ \exists x(x \leq g(x) \to f(x) \sim f(g(x)))\end{array}}{\begin{array}{c}\forall x(0 \sim f(x)),\\ \forall x\forall y\Big(x \leq y \to f(y) \preceq f(x)\Big) \vdash\\ \exists x(x \leq g(x) \to f(x) \sim f(g(x)))\end{array}}\;cut
$$

3.2 Clauses and Resolution

The core of the CERES method of cut elimination is extracting a clause set from the given LK-proof, which represents the cut structure, and using resolution to refute the said clause set. This resolution refutation is then used as a proof skeleton. If we plan to use the resolution refutation as a proof skeleton we ought to have a way to interpret sequents as clauses and resolution steps as cuts. In the book "The Resolution Calculus" [42] Resolution is defined over atomic first-order formulae (this is exactly how we will define resolution in this section). Also, to clearly outline what we mean by resolution, we will define resolution in this section. We will use the following definition for clauses as sequents:

Definition 3.2.1 (Clause [14])**.** *A sequent of the form $A_1, \cdots, A_j \vdash$ where $j, k \in \mathbb{N}$ and $A_1, \cdots, A_j, B_1, \cdots, B_k$ are atomic, will be referred to as a clause.*

Definition 3.2.2 (Reduct [14])**.** *Let C be a clause and D an atom occurring in the antecedent or consequent of C. Let C' be the clause where the multiplicity of D in antecedent (or in the consequent) is reduced to one, then C' is called a* reduct of C.

One can consider the Reduct of a clause C for an atom D occurring n times as the application of n contractions. We will assume that clauses are always in there most reduced form for resolution. Next we define unifiers of two atoms, an essential part of defining resolution.

Definition 3.2.3 (Unifier [14])**.** *Let $\mathcal{M}$ be a nonempty set of atoms and σ be a substitution with $A\sigma = B\sigma$ for all $A, B \in \mathcal{M}$; then σ is called a unifier of $\mathcal{M}$. σ is called a most general unifier of $\mathcal{M}$ if for all unifiers τ of $\mathcal{M}$ there exists a substitution ρ with $\tau = \sigma\rho$.*

Definition 3.2.4 (Factor [14])**.** *Let $C : A_1, \cdots, A_n \vdash B_1, \cdots, B_m$ be a clause and D be a subsequent of $A_1, \cdots, A_n$ n or of $B_1, \cdots, B_m$. Let σ be a most general unifier of the set of atoms in D. Then the reduct of $C\sigma$ w.r.t. (an element in) $D\sigma$ is called a factor of C.*

We will use the definition of a factor to define resolution: such that C_1 contains a literal $A \vdash$ and C_2 contains a literal $\vdash B$ which are unifiable.

Definition 3.2.5 (Resolution)**.** *Let C_1 and C_2 be two clauses which are variable disjoint. Let C_1' and C_2' be factors of C_1 and C_2 respectively. Let C_1' be $\Gamma_1, A \vdash \Delta_1$ and C_2' be $\Gamma_2 \vdash \Delta_2, B$ such that A and B are unifiable using the most general unifier σ. Then, the clause $\Gamma_1\sigma, \Gamma_2\sigma \vdash \Delta_1\sigma, \Delta_2\sigma$ is called the resolvent of C_1 and C_2.*

By this definition, and as pointed out in [14], The resolution rule defined above is just atomic cuts on factors. Given a set of clauses which can be resolved in such a way that the last resolution step results in $\vdash$, i.e. no formulae in the antecedent or consequent, the sequence of resolutions step is called a *resolution refutation* . Though, the resolution outlined above differs from the original resolution method constructed by Robinson [53], It is easy to show that the two methods of resolution are equivalent. Thus, it holds for this method of resolution, as it does for Robinson's method, that resolution is complete for unsatisfiable clause sets, that is if a clause set is unsatisfiable , then there is a resolution refutation of this clause set.

Though, up till this point, the resolution refutations which are derived by the given resolution principle require the usage of substitutions, something which is not part of the definition of the sequent calculus. However, it is a well known fact that once one has found a resolution refutation, one can make a ground resolution refutation, that is a resolution refutation where all the leaves of the refutation tree already have the instantiations of the variables they will need to refute the clause set. This type of resolution refutation was referred to as a *ground projection* in [13]. There is a quite large example of a ground projection of a resolution refutation for the NIA-schema instance 2 in B.2.

3.3 Characteristic Clause set Extraction

Before we delve into the extraction of the characteristic clause set, we ought to discuss proof skolemization and the CERES cut-elimination method, specifically, the CERES method only works for skolemized proof . We will not go into great detail about why this is necessary, but one can consider what happens when one eigenvariable is used in a proof for both a weak and a strong quantifier. The Strong quantifier essentially states that the quantification is over all terms, and the weak quantifier only takes a witness that the quantifier is true. Skolemization ensures that these interpretation of the two quantifiers hold true in the characteristic clause set by defining the canonical model to be used by resolution. If Skolemization was not done, problems can occur when constructing the proof projection we introduce later in this chapter. Proof skolemization was introduced in [12] and is a reversible process, i.e. after cut-elimination has been performed one can reintroduce the strong quantifiers . Proof skolemization will be a factor in the proof analysis of Ch. 5, though we will automatically perform the skolemization and will not discuss it any further.

Definition 3.3.1. *Let Ω be a set of formula occurrences in an* $\mathbf{LK}$*-derivation φ and ν be a node in φ. Then $S(\nu, \Omega)$ is the subsequent of $Seq(\nu)$ (The sequent located at position ν) obtained by deleting all formula occurrences which are not ancestors of occurrences in Ω.*

Definition 3.3.2 (Characteristic Clause term [15])**.** *Let φ be an* $\mathbf{LK}$*-derivation of a sequent* $\mathbf{S}$ *and let Ω be the set of all occurrences of cut formulas in φ. We define the characteristic (clause) term $\Theta(\varphi)$ inductively via $\Theta(\varphi) \setminus \nu$ for occurrences of sequents ν in φ:*

- *Let ν be the occurrence of an initial sequent in φ. Then $\Theta(\varphi) \setminus \nu = \{S(\nu, \Omega)\}$.*

  Let us assume that the clause terms $\Theta(\varphi) \setminus \nu$ are already constructed for all sequent occurrences ν in φ with $depth(\nu) \leq k$. Now let ν be an occurrence with $depth(\nu) = k+1$. We distinguish the following cases:

  (a) ν is the conclusion of μ, i.e. a unary rule applied to μ gives ν. Here we simply define $\Theta(\varphi) \setminus \nu = \Theta(\varphi) \setminus \mu$.

  (b) ν is the consequent of μ_1 and μ_2, i.e. a binary rule ξ applied to μ_1 and μ_2 gives ν.

  (b1) The occurrences of the auxiliary formulas of ξ are ancestors of Ω, thus the formulas occur in $S(\mu_1, \Omega)$, $S(\mu_2, \Omega)$. Then $\Theta(\varphi) \setminus \nu = \Theta(\varphi) \setminus \mu_1 \oplus \Theta(\varphi) \setminus \mu_2$..

(b2) The occurrences of the auxiliary formulas of ξ are not ancestors of Ω. In this case we define $\Theta(\varphi) \setminus \nu = \Theta(\varphi) \setminus \mu_1 \oplus \Theta(\varphi) \setminus \mu_2$.

Note that, in a binary inference, either the occurrences of both auxiliary formulas are ancestors of ω or none of them. Finally the characteristic term $\Theta(\varphi)$ is defined as $\Theta(\varphi) \setminus \nu$ where ν is the occurrence of the end-sequent.

Definition 3.3.3 ([15]). *Let φ be an LK-derivation and $\Theta(\varphi)$ be the characteristic term of φ. Then $CL(\varphi)$, for $CL(\varphi) = |\Theta(\varphi)|$, is called the characteristic clause set of φ.*

Theorem 3.3.1 ([15]). *Let φ be a regular LK-proof of a closed sequent and such that the end sequent of φ has been skolemized. Then $CL(\varphi)$ is unsatisfiable.*

This definition is quite heavy to digest, though it translates to simply following the paths up the proof which contain parts of formulae found lower in the proof (i.e. the formula occurrence). We will use the following example proof from [14]. More complex schematic clause sets will be discussed in later chapters.

Example 3.3.1. *Let φ be the proof (we will use u and v as free variables, and c as a constant symbol):*

$$\dfrac{\varphi_1 \qquad \varphi_2}{\forall x(P(x) \to Q(x)) \vdash \exists x(P(c) \to Q(x))}\, cut$$

Where φ_1 is defined as:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P(u)* \vdash P(u) \qquad Q(u) \vdash Q(u)*}{P(u) \to Q(u), P(u)* \vdash Q(u)*}\to: l}{P(u) \to Q(u) \vdash P(u) \to Q(u)*}\to: r}{P(u) \to Q(u) \vdash \exists y(P(u) \to Q(y))*}\exists : l}{\forall x(P(x) \to Q(x)) \vdash \exists y(P(v) \to Q(y))*}\forall : l}{\forall x(P(x) \to Q(x)) \vdash \forall x \exists y(P(x) \to Q(y))*}\forall : r$$

And where φ_2 is defined as:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P(c) \vdash P(c)* \qquad Q(v)* \vdash Q(v)}{P(c) \to Q(v)*, P(c) \vdash Q(v)}\to: l}{P(c) \to Q(v)* \vdash P(c) \to Q(v)}\to: r}{P(c) \to Q(v)* \vdash \exists x(P(c) \to Q(x))}\exists : r}{\exists y(P(c) \to Q(y))* \vdash \exists x(P(c) \to Q(x))}\exists : l}{\forall x \exists y(P(x) \to Q(y))* \vdash \exists x(P(c) \to Q(x))}\forall : l$$

We marked the cut ancestors with the $*$ symbol. Following the definition above, we start with the following 4 clause sets:

$$\begin{gathered} C_1 = \{\vdash P(c)\} \\ C_2 = \{Q(v) \vdash\} \\ C_3 = \{P(u) \vdash\} \\ C_4 = \{\vdash Q(u)\} \end{gathered}$$

The clause sets after the first inference after the axioms are:

$$\begin{gathered} C^1_{\to:l} = C_1 \cup C_2 \\ C^2_{\to:l} = C_3 \otimes C_4 = \{Q(u) \vdash P(u)\} \end{gathered}$$

The rest of the rules in φ_1 and φ_2 are unary, thus, we can skip them. The next binary rule is the cut inference thus, the final clause set will be:

$$C_{End} = C^1_{\to:l} \cup C^2_{\to:l}$$

Which is as follows:

$$\{Q(u) \vdash P(u); P(c) \vdash; \vdash Q(v)\}$$

which has an obvious resolution refutation and ground projection . *A ground projection is a resolution refutation where instead of unifying terms at each resolution step, the exact terms which are necessary are already instantiated at within the Literals in the axioms.*

Now that we have a definition of the characteristic clause set and an idea of how to construct the proof skeleton using the resolution refutation, when there is one, the only question left is if the clause set will always be refutable. This result was proven in [14].

Theorem 3.3.2 ([14])**.** *Let φ be a cut-free proof of the sequent S and Ω is the set of occurrences of cut formulae in φ. Then the set of clauses $CL(\varphi, \alpha)$ is unsatisfiable.*

3.4 Proof Projections Short description

Though proof projections are the addition to the resolution refutation allowing construction of a proof for the original end sequent, originally defined in [13], and are important to the first order CERES method, due to the complexity of defining projections for schematic CERES, the concept was dropped. Though, it is not only because of the complexity of the definitions that the concept was dropped it is also due to the fact that the end result of the schematic CERES method was not an ANCF-schema of the original proof, but a resolution refutation schema and a projection schema. The two parts could only be joined once the proof is instantiated, which kind of defeats the purpose of the method, i.e. to make a cut-free proof schema from a proof schema with cuts. The projections themselves do not contain much information concerning the steps of the proof analysis. Though through out this dissertation we will lightly mention projections and constructing the ANCF of a cut-free proof, We will not define the projection schema and only give an outline of what projections are in this section. It turned out that the most important part of the projections is their end sequent when it comes to schematic CERES.

Essentially the proof projections are the parts of the axioms in an LK-proof which do not go to the cuts. The construction of the characteristic clause set concerns itself with the inference rules applied to the ancestors of the cuts, while the projections concern themselves with the inference rules applied to the end-sequent ancestors. All inference rules applied to cut sequent ancestors are dropped in the projections. It is important to note that each projection is attached to a specific clause in the clause set.

Projection to $Q(u) \vdash P(u)$:

$$\dfrac{\dfrac{P(u)* \vdash P(u) \qquad Q(u) \vdash Q(u)*}{P(u) \to Q(u), P(u)* \vdash Q(u)*}\ \to: l}{\forall x(P(x) \to Q(x)), P(u)* \vdash Q(u)*}\ \forall : l$$

Projection to $P(c) \vdash$:

$$\dfrac{\dfrac{\dfrac{P(c) \vdash}{P(c) \vdash Q(v), P(c)*}\ w : r}{\vdash P(c) \to Q(v), P(c)*}\ \to: r}{\vdash \exists x(P(c) \to Q(x)), P(c)*}\ \exists : r$$

Projection to $\vdash Q(v)$:

$$\dfrac{\dfrac{\dfrac{\vdash Q(v)}{P(c) \vdash Q(v)}\ w : l}{\vdash P(c) \to Q(v)}\ \to: r}{\vdash \exists x(P(c) \to Q(x))}\ \exists : r$$

3.5 constructing an ACNF for Ex. 3.3.1

We can put the projections together with the following ground resolution refutation to get an ACNF of the original proof.

$$\dfrac{\dfrac{Q(c) \vdash P(c) \qquad P(c) \vdash}{Q(c) \vdash} \qquad \vdash Q(c)}{\vdash}$$

The ACNF of φ

$$\dfrac{\dfrac{\dfrac{P(c)* \vdash P(c) \qquad Q(c) \vdash Q(c)*}{P(c) \to Q(c), P(c)* \vdash Q(c)*}\ \to: l}{\forall x(P(x) \to Q(x)), P(c)* \vdash Q(c)*}\ \forall : l \qquad \dfrac{\dfrac{\dfrac{P(c) \vdash}{P(c) \vdash Q(c), P(c)*}\ w : r}{\vdash P(c) \to Q(c), P(c)*}\ \to: r}{\vdash \exists x(P(c) \to Q(x)), P(c)*}\ \exists : r}{\forall x(P(x) \to Q(x)) \vdash Q(c), \exists x(P(c) \to Q(x))}\ cut \tag{1}$$

$$\dfrac{\dfrac{\overset{(1)}{\forall x(P(x) \to Q(x)) \vdash Q(c), \exists x(P(c) \to Q(x))} \qquad \dfrac{\dfrac{\dfrac{Q(c)* \vdash Q(c)}{Q(c)*, P(c) \vdash Q(c)}\, w: l}{Q(c)* \vdash P(c) \to Q(c)}\, \to: r}{Q(c)* \vdash \exists x(P(c) \to Q(x))}\, \exists: r}{\forall x(P(x) \to Q(x)) \vdash \exists x(P(c) \to Q(x)), \exists x(P(c) \to Q(x))}\, cut}{\forall x(P(x) \to Q(x)) \vdash \exists x(P(c) \to Q(x))}\, c: r$$

An additional contraction inference rule was added to remove the repetition in the end sequent. The CERES method outlined above can be applied to any first-order skolemized proof. There are extensions of the method to deal with equational reasoning [10] which was essential to the proof analysis of the Fürstenberg's proof [11]. In the next section we will discuss the extension of the first-order CERES method to proof schema [31].

CHAPTER 4

Cut-Elimination on First-order Proof Schemata: Ceres$_S$

4.1 Introduction

In the previous chapter we outlined and discussed the CERES method of cut elimination [14] which is a method for performing cut elimination by resolution on first order LK-calculus proofs. The method can be described shortly as follows: 1) First, one extracts a clause set from the proof which represents the structure of the cuts in the proof. 2) Then one finds a resolution refutation of the clause set and grounds the resolution refutation (create a *ground projection* of the refutation). 3) From the original proof one extracts proof projections, which are sub-proofs of the original LK-proof, of which only contains inference rules applied to the end-sequent ancestors. 4) As a last step, we attach the projections to the resolution refutation and get an LK-proof in ACNF proving the original end sequent.

Our goal in this chapter is to describe the work of [30, 31] which extended this method to cut-elimination on proof schemata. Proof schemata are essentially LK-proofs which can have recursive elements, i.e. a forest of proof trees with a total ordering and special 0-ary inference rules connecting the different proof trees in the forest. Like other extensions of the CERES method [37, 43], many problems arose which do not exist in the standard first-order version of the method. For example, when constructing the characteristic clause set, calling the same proof tree in the forest can have different effects on the characteristic clause set depending on when it is called in the proof schema. This effect resulted in the concept of *configurations* which we will discuss later on in this chapter.

Before moving on to discussing and outlining the method, we will discuss the motivation for this work. Similarly to the higher-order CERES method [37], the motivation for investigating cut elimination for proof schemata was the analysis of the Fürstenberg's proof [11]. When formalizing the Fürstenberg proof there was a branch in the research directions, one being a formalization of the proof in second order arithmetic, and the other being a formalization of the proof as a sequence of proofs indexed by a free parameter representing the number of prime numbers assumed to exists. In the case of the second-order arithmetic formalization, the clause set resulting from the formalization was overly complex and pretty much unparsible by a human. For theorem provers the problem was pretty much the same (though the computers could read the clause set, but had no clue what to do with it). In the published work concerning the analysis of the Fürstenberg proof [11], the schematic formalization was used given that the clause set was much easier to handle (though nonetheless very large and complex). This was the first success for proof schema as a tool in proof analysis. However, at the time of the Fürstenberg proof analysis, there was no formal framework for performing proof analysis/cut elimination on a proof schema. This was the motivation for the work of [30, 31].

Even though the original motivation was the formalization of the Füstenberg proof, a even more interesting phenomena was discussed and presented in [30], namely, the connection between induction and proof schema . Essentially, when we formalize a proof as a proof schema instead of proving an end sequent, or a formula, we are proving a first-order definable sequence of end sequents/formulae, of which is more expressive than what can be proved in first order logic using the standard LK-calculus (consider formulae with a schematic number of quantifiers). Though, we are not stating that all first-order definable sequences can be proven in the proof schema framework of [30, 31], but rather that an interesting set of them, a subset of the inductively provable ones, is formalizable as proof schema. The problem which arises with formulae which are only provable in the presence of induction is that the introduction of an induction rule into the LK-calculus stops reductive cut elimination from working [30, 31]. The problem being that induction introduces an eigenvariable in the middle of an LK-derivation and reductive cut elimination gets stuck at this point. Using proof schema and the schematic CERES method allows one to get around this problem as it was shown in [30, 31]. However, the work of [30, 31] was not the first attempt and success of getting cut-elimination results for a sequent calculus with an induction rule. In the work of [47], a logic was formulated which allowed for cut-elimination in the presence of induction. However, the formulation of induction found in this logic did not allow for the subformula property , i.e. it is not necessarily the case that after cut elimination all formulae in a proof are subformulae of the end sequent. The work of [30, 31] has the subformula property and thus opens the doors to Herbrand sequent extraction [38].

In this section we will introduce proof schemata, show how induction becomes a problem for cut-elimination, and introduce generalized versions of the constructs from the previous chapter, versions of the CERES constructions designed specifically to handle proof schema. The methods introduced here will be used for the rest of the thesis. One thing to note is that schematic proof projections, which are introduced [30], but where removed in [31], will not be discussed in this chapter. They are complex to work with and do not play a role in this dissertation. The schematic projection extraction method was removed from [31] because it is overly complex and ends up not playing a role in the proof analysis as much as it did in the first order CERES method, namely, the end goal in the schematic CERES method is the construction of a Herbrand sequent rather than the construction of an ACNF. Every formal proof schema discussed in this work has projections which can be constructed using the projection extraction method introduced in the previous chapter. This is all we need for extraction of the Herbrand sequent, because we only need to know the end sequent of each of the projections. For those interested in schematic projections extraction please see [30].

4.2 Notations and Definitions

We will work with the exact same calculus as defined in the previous chapter 3.1, only difference being that we will add a few extensions to account for the schematic part of the calculus. The extended calculus, with the addition of an equational theory and induction rule, will be referred to as $\mathbf{LKIE}$ as was done in [31]. The calculus extended by only the equational theory will be referred to as $\mathbf{LKE}$ and the schematic version of this equational theory calculus will be referred to as $\mathbf{LKS}$. For example, instead of working with a single sorted logic, we will work with a two sorted logic which has a sort for the schematic terms (the *schematic sort* ω) and the *individual sort* ι for the standard term language. In the case of the individual sort ι, the terms are constructed exactly as the terms where constructed in the term language of the previous chapter. However, for the schematic sort ω, all terms are built from the constant 0 and the monadic function $s : \omega \to \omega$, essentially all natural numbers. The schematic sort is used to iterate through schematic terms, predicates and the proof schemata themselves. Consider this iteration as being analogous to the iteration found in primitive recursion functions. We will have free variables specifically for the schematic sort, i.e. the free parameters. The schematic sort does not need bound variables being that there is no quantification over schematic terms, only individual sort terms. To highlight the problem with induction and the sequent calculus we will introduce an induction rule and a rule for equality modulo a given equational theory ε. Later on, the induction rule will be replaced by *proof links* (the 0-ary rules which connect trees in the given forest of proof trees), though the equational theory will be kept and used to define primitive recursive terms.

$$\frac{S\,[t]}{S\,[t']}\,\varepsilon \qquad\qquad \frac{\Gamma \vdash \Delta, A(0) \qquad \Pi, A(k) \vdash \Lambda, A(s(k))}{\Gamma, \Pi \vdash \Lambda, \Delta, A(t)}\,ind$$

The idea behind the ε inference rule is that S is a sequent and the term t is replaced by a term t' such that given the equational theory ε the following holds, $\varepsilon \models t = t'$. Concerning the induction rule, we have an auxiliary sequent proving $A(0)$ and another auxiliary sequent proving $A(k) \to A(s(k))$, thus, any term constructable in the term language can be placed in the formula A. The calculus constructed from the calculus of Sec. 3.1 with the addition of these two inference rules will be known as $\mathbf{LKIE}$. A note made in [31] concerning the equational theory is that only terminating and confluent rewriting systems will be considered for the equational theory and thus the chosen theory will always be decidable. We make the same consideration in this work. Later on, when we drop the induction rule, we will consider the calculus without it $\mathbf{LKE}$.

4.3 Cut Elimination and Induction: A Motivating Example [31]

In this section we will outline and discuss the motivating example from Sec. 2.1 of [31] where a equational theory and an end sequent are given, such that the end sequent modulo the equational theory requires induction to construct a proof. The proof which is constructed contains a cut which is above the induction rule in the proof tree (closer to the end sequent). It is then shown that this cut rule cannot be reductively pushed over the induction rule in the proof tree.

The required equational theory is quite simple and can be written in the following primitive recursive form. A note to make here is that all equational theories we will be considering in this dissertation will have this recursive form and a general definition will be given later on in this chapter.

$$\varepsilon = \begin{cases} f_I(0,x) = x \\ f_I(s(n),x) = f(f_I(n,x)) \end{cases}$$

The end sequent we will prove modulo this equational theory is the following:

$$\forall x(P(x) \rightarrow P(f(x))) \vdash \forall n((P(f_I(n,c)) \rightarrow P(g(n,c))) \rightarrow (P(c) \rightarrow P(g(n,c))))$$

As surreptitiously pointed out earlier, this sequent is not valid in pure first order logic and requires induction. One needs to use the calculus **LKIE** to prove the sequent. The inductive lemma (cut formula) used in [31] to prove the sequent is as follows:

$$\forall x(P(x) \rightarrow P(f(x)) \vdash \forall n \forall x(P(x) \rightarrow P(f_I(n,x)))$$

The following proof using this inductive lemma was provided in [31]. The simple, but tedious first-order and cut-free parts of the proof are skipped. The proof ψ of the inductive lemma is as follows (we will mark the non-inductive and cut-free part of the proof with ψ_{fo}):

$$\dfrac{\dfrac{\dfrac{\dfrac{P(f_I(0,u)) \vdash P(f_I(0,u))}{P(u) \vdash P(f_I(0,u))}\,\varepsilon}{\vdash (P(u) \rightarrow P(f_I(0,u)))}\rightarrow: r}{\vdash \forall x(P(x) \rightarrow P(f_I(0,x)))}\,\forall : r \qquad \begin{array}{c} \vdots \\ \psi_{fo} \\ P(x) \rightarrow P(f_I(\gamma,x)), \\ \forall x(P(x) \rightarrow P(f(x)) \vdash \\ \forall x(P(x) \rightarrow P(f_I(s(\gamma),x))) \end{array}}{\dfrac{\forall x(P(x) \rightarrow P(f(x)) \vdash \forall x(P(x) \rightarrow P(f_I(\alpha,x)))}{\forall x(P(x) \rightarrow P(f(x)) \vdash \forall n \forall x(P(x) \rightarrow P(f_I(n,x)))}\,\forall : r}\,Ind$$

Using the proof ψ and another proof ϕ_{fo} which is again non-inductive and cut-free, we can prove the end sequent outlined above with a single cut inference rule. The following proof will be labelled ϕ:

$$\dfrac{\begin{array}{c} \psi \\ \forall x(P(x) \rightarrow P(f(x)) \vdash \\ \forall n \forall x(P(x) \rightarrow P(f_I(n,x))) \end{array} \qquad \begin{array}{c} \vdots \\ \phi_{fo} \\ \forall n \forall x(P(x) \rightarrow P(f_I(n,x))) \vdash \\ \forall n((P(f_I(n,c)) \rightarrow P(g(n,c))) \rightarrow (P(c) \rightarrow P(g(n,c)))) \end{array}}{\forall x(P(x) \rightarrow P(f(x))) \vdash \forall n((P(f_I(n,c)) \rightarrow P(g(n,c))) \rightarrow (P(c) \rightarrow P(g(n,c))))}\,cut$$

If we were to attempt reductive cut elimination on the proof ϕ in an attempt to show that reductive cut elimination does not work on a proof with an induction rule, we do not have to consider the subproof ϕ_{fo} being that there is no induction necessary to prove the end sequent of ϕ_{fo}. Thus, we instead consider ψ. We can eliminate the universal quantification at the beginning of ψ but when we reach the induction inference, we see that the eigenvariable above the induction inference does not match the variable used below the inference and reductive cut-elimination gets stuck. As mentioned in [31], if the eigenvariable was replaced with a ground term (not possible in the case of our proof being that we require a strong quantification) which is provably equivalent to the term used in the induction stepcase over the equational theory ε, in the process, matching the auxiliary sequents with the main sequent, the induction inference can be replaced by a sequence of cuts and cut-elimination can proceed as normal. This change from induction to a sequence of cuts is exactly the idea behind schematic proofs and schematic cut elimination.

There are no cut-free proofs of the sequent S in the calculus $\mathbf{LKIE}$. Instead of considering S, we can consider the sequence of sequents S_n:

$$\forall x(P(x) \to P(f(x))) \vdash (P(f_I(\overline{n}, c)) \to P(g(\overline{n}, c))) \to (P(c) \to P(g(\overline{n}, c)))$$

where instead of $\overline{n}$ being an eigenvariable, it is a numeral from the schematic sort ω. These sequents have cut-free proofs in $\mathbf{LKE}$. What we can now consider is an infinitary (potentially infinitary) cut-elimination over these sequences of sequents in the system $\mathbf{LKE}$ as a replacement for induction. This is the main motivating idea for the formalism developed in the next section and rest of this chapter.

4.4 Schematic language

Now that we have introduced the motivation behind proof schemata as a replacement for induction. We can consider the formal definition of our schematic language. In this section we introduce the *schematic first-order language* which allows for the formal definition of terms like the one found in the equational theory presented in the previous section, as well as recursively defined predicates. As mentioned earlier there are two sorts, one for the standard term language ι and the second for the schematic terms ω. The ι sort has both countably many free and bound variables, and the ω sort has countably many free variables. An additional type of variable we introduce in this section is *schematic variables* , which can be considered as a type of second-order variable. Schematic variables are of the type $\omega \to \iota$ and have an argument (they resemble 1-ary functions symbols), which is a term from the ω sort. We will represent these variable sequences with a $^-$ on top of the variable name. The point of this type of variable is to take care of the case when we have a schematic sequence of variables. Schematic sequences of variables are a regular occurrence in both the proof schemata as well as the refutations of the characteristic clause sets. As a side note, being that schematic variables are a generalization of the standard variables found in classical first order logic, rwe can use them to define standard first-order logic variables as well. Consider a schematic variable α instantiated with free parameter n, i.e. $\alpha(n)$. If we ground n such that it is any numeral of the ω sort, i.e. $\alpha(\bar{n})$, then $\alpha(\bar{n})$ is a standard first order variable. We will write main definitions in this chapter considering only schematic variables based on the above described property.

The term language used in our schematic language is the same as standard first-order logic (as outlined in ch. 3) plus the term rewriting rules found in the equational theory ε; this is specifically concerning the ι sort. To make this distinction between what is a term in the standard sense and what is a term concerning the equational theory we need to distinguish two types of function symbols, namely, the *constant function symbols* and the *defined function symbols*. The constant function symbols are the function symbols defined for the language found in ch. 3. on the other hand, the defined function symbols have a recursive definition found in the equational theory.

Definition 4.4.1 ([31]). *For every defined function symbol f we assume that it has type $\omega \times \tau_1 \times \cdots \times \tau_n \to \tau$ for $n \geq 0$ and we assume that provided in the equational theory ε the following two rewrite rules are provided*

$$f(s(y), \overline{x}) \to t\,[f(y, \overline{x})]$$
$$f(0, \overline{x}) \to t_0$$

where $t\,[\cdot]$ is a context, $V(t_0) \subseteq \{x_1, \cdots, x_n\}$, $V(t\,[f(y,x)]) \subseteq \{y, x_1, \cdots, x_n\}$, and t_0 , t are terms not containing f; if a defined function symbol g occurs in t_0 or t then $g \prec f$, where $\prec$ is an irreflexive ordering on defined function symbols.

There are two simple results concerning these types of functions provided in [31]. We will not reprove these results, rather we will only state them.

Theorem 4.4.1 ([31]). *The unification problem of terms is undecidable.*

Using the same construction outlined above we can make recursively defined predicate symbols. Like we did with function symbols, we separate the set of predicate symbols used in constructing formulae into two types, *constant predicate symbols* and *defined predicate symbols* . Also, there is an irreflexive ordering relation on the defined predicate symbols as there was on the function symbols. We use these primitive recursive predicate definitions to build *schematic formulae* . One extra thing to note about schematic formulae is the binding of bound variables to quantifiers. In the case when a bound variable name repeats within an unfolded defined predicate definition, only the innermost binding above the bound variable occurrence counts as binding the variable. This fixes some of the ambiguity found in recursive formula definitions.

Proposition 4.4.1. *Let A be a formula. Then every rewrite sequence starting at A terminates, and A has a unique normal form.*

Example 4.4.1. *The usual primitive recursive definition of addition and multiplication can be written in this language outline above:*

$$x + s(y) \equiv \begin{cases} +(s(y), x) \to s(+(y, x)) \\ +(0, x) \to x \end{cases}$$

$$x \cdot s(y) \equiv \begin{cases} mul(s(y), x) \to +(mul(y, x), x) \\ mul(0, x) \to 0 \end{cases}$$

We can also define the iterated $\bigwedge$ from Ch. 2:

$$\bigwedge_{i=0}^{s(y)} P(i) \equiv \begin{cases} \wedge_P(s(y)) \to \wedge_P(y) \wedge P(s(y)) \\ \wedge_P(0) \to P(0) \end{cases}$$

For a more complex example we can construct the following defined predicate which is used to define the gNiA-schema:

Example 4.4.2.

$$\begin{cases} eq(n+1,p) \rightarrow \exists q(q < p \wedge eq(n,q) \wedge f(q) \sim f(p)) \\ eq(0,p) \rightarrow \exists q(q < p \wedge f(q) \sim f(p)) \end{cases}$$

Where p, q are of the ι sort, f is a function from the ι sort to the ω sort, and $\sim$ is a predicate taking two members of the ω sort.

4.5 Schematic Proofs

Now that we have a formal notion of the schematic language, we can discuss the construction of schematic proofs. We will now give a natural notion of *proof schema* for the language defined in the previous section, and compare these proof schemata with the calculus $\mathbf{LKIE}$. For examples of proof schemata see Appendices A, B, & C. To define this concept we need additional notions [31]: Let us consider the sequent $S(\bar{x})$ with a vector of free variables $\bar{x}$ (a schematic variable), then by the sequent $S(\bar{t})$ we use to denote the sequent $S(\bar{x})$ where each of the variables in $\bar{x}$ is replaced by the terms in the vector $\bar{t}$ respectively, assuming that they have the appropriate type. We assume a countably infinite set of *proof symbols* denoted by $\varphi, \psi, \varphi_i, \psi_j \ldots$. Let φ be a proof symbol and $S(\bar{x})$ a sequent, then the expression $\dfrac{(\varphi(\bar{t}))}{S(\bar{t})}$ is called a *proof link* . For a variable $k : \omega$, proof links such that $\mathrm{V}(a_1) \subseteq \{k\}$ are called *k-proof links* .

Definition 4.5.1 ([31]). *The sequent calculus* $\mathbf{LKS}$ *consists of the rules of* $\mathbf{LKE}$*, where proof links may appear at the leaves of a proof.*

Definition 4.5.2 (Proof schemata [31]). *Let ψ be a proof symbol and $S(n, \bar{x})$ be a sequent such that $n : \omega$. Then a* proof schema pair for *ψ is a pair of* $\mathbf{LKS}$*-proofs $(\pi, \nu(k))$ with end-sequents $S(0, \bar{x})$ and $S(k+1, \bar{x})$ respectively such that π may not contain proof links and $\nu(k)$ may contain only proof links of the form* $\dfrac{(\psi(k, \bar{a}))}{S(k, \bar{a})}$

and we say that it is a proof link to ψ. We call $S(n, \bar{x})$ the end sequent of ψ and assume an identification between the formula occurrences in the end sequents of π and $\nu(k)$ so that we can speak of occurrences in the end sequent of ψ. Finally a proof schema Ψ is a tuple of proof schema pairs for $\psi_1, \cdots \psi_\alpha$ written as $\langle \psi_1, \cdots \psi_\alpha \rangle$, such that the $\mathbf{LKS}$*-proofs for ψ_β may also contain k-proof links to ψ_γ for $1 \leq \beta < \gamma \leq \alpha$. We also say that the end sequent of ψ_1 is a the end sequent of Ψ.*

The above definition will be use for the rest of this dissertation for constructing and describing infinite sequences of proofs. However in the case of the gNiA-schema of Ch. 9, a more complex structure is used being that the gNiA-schema has two parameters. However, we will not define a new proof schema structure for this case as multiple parameter schema is left to future work. Instead we use the proof schema structure defined here and assume an ordering on the free parameters, i.e. once one uses a parameter higher in the order (we assume an ordering on the free parameters), the parameters lower in the ordering cannot be use. This simulates pairing of two numbers. Also to note, we will not provide any examples in this section and instead we refer to the next chapter and the Appendices, i.e. Ch. 5 & A, B. We now move on to evaluation of proof schema and a soundness argument for the **LKS** calculus.

Definition 4.5.3 (Evaluation of proof schemata [31]). *Let Ψ be a proof schema. We define the rewrite rules for proof links in Ψ*

$$\frac{(\psi(0,\bar{x}))}{S(0,\bar{x})} \to \pi, \qquad \frac{(\psi(s(k),\bar{x}))}{S(s(k),\bar{x})} \to \nu(k)$$

for all proof schema pairs $(\pi, \nu(k))$ for Ψ. Now for $\gamma \in \mathbb{N}$ we define $\psi \downarrow_\gamma$ as a normal form of

$$\frac{(\psi(\gamma,\bar{x}))}{S(\gamma,\bar{x})}$$

under the rewrite system just given extended by the rewrite rules for defined function and predicate symbols. Further, we define $\Psi \downarrow_\gamma = \Psi_1 \downarrow_\gamma$.

Proposition 4.5.1 (Soundness of proof schemata [31]). *Let Ψ be a proof schema with end-sequent $S(n,\bar{x})$, and let $\gamma \in \mathbb{N}$. Then there exists an LK-proof of $S(\gamma,\bar{x}) \downarrow$.*

Proof. See [31] for a proof of this theorem □

Corollary 4.5.1. *The sequent calculus* **LKS** *is sound.*

Definition 4.5.4 (k-simple [31]). *Let π be an* **LKIE**-*proof. If all induction rules in π are of the following form:*

$$\frac{\Gamma \vdash \Delta, A(0) \qquad A(k), \Pi \vdash \Lambda, A(k+1)}{\Gamma, \Pi \vdash \Delta, \Lambda, A(t)}\ \mathit{IND}$$

where $k : \omega$ and $V(t) \subseteq \{k\}$, then π is called k-simple.

Proposition 4.5.2. *Let π be a k-simple* **LKIE**-*proof of a sequent S. Then there exists a proof schema with end-sequent S.*

Proof. See [31] for a proof of this theorem □

The last few statements of this chapter establish that the calculus is sound and that given a proof in the **LKIE** calculus with simple induction inference usage, then there is a proof in the **LKS** calculus proving the same end sequent. In the next section we introduce the concept of a resolution schema and schematic clause sets.

4.6 Resolution Schemata

The heart of the CERES method was the resolution refutation of the characteristic clause set . For the schematic CERES method the resolution refutation is also an integral part of the method, though it is much more complex when defined over proof schema. In this section we provide a definition of what we mean when we refer to schematic resolution. We also provide a definition of schematic clauses/clause sets, and how to write a schematic refutation.

To define schematic resolution derivations and resolution refutations we need to extend the current language for **LKS**. We need to add two additional sets of variables, namely, *clause variables* , denoted by $X, Y, \cdots$, and *clause-set variables* denoted by $\mathcal{X}, \mathcal{Y}, \cdots$. Later on in this dissertation we will run into clauses which cannot be written in the language provided in this section, (see Ch. 6, specifically the iterated max term). To give an idea of the expressiveness of the formalism provided in this chapter we consider the clause used in the an example taken from [31]:

Example 4.6.1.

$$D_\gamma : Q(x_0, y_0) \vdash P(x_0, y_0), \cdots , P(x_0, f^\gamma(y_\gamma)), R(x_0, z)$$

The above sequent (clause) D_γ has a schematic number of variables and has a schematic length. The formalism concerning a schematic number of variable from the previous section deals with the sequence of variable $\bar{y}$ however, we have yet to introduce a formalism for the schematic length sequents. Such a definition is part of the goal of this section.

Definition 4.6.1 (Clause schema [31])**.** *Let a be an arithmetic term, $\bar{u}$ a sequence of schematic variables and $\bar{X}$ a sequence of clause variables. Then $c(a, \bar{u}, \bar{X})$ is a clause schema w.r.t. the rewrite system $\mathcal{R}$:*

$$c(0, \bar{u}, \bar{X}) \rightarrow C \circ \bar{X} \quad and \quad c(k + 1, \bar{u}, \bar{X}) \rightarrow c(k, \bar{u}, \bar{X}) \circ D$$

where C is a clause with $V(C) \subseteq \{\bar{u}\}$ and D is a clause with $V(D) \subseteq \{k, \bar{u}\}$. Clauses and clause variables are clause schemata w.r.t. the empty rewrite system.

As was done with term variables in the previous section we introduce a definition of substitution for clause variables, i.e. a mapping from clause variables to clauses. Let $C_1, \cdots , C_\alpha$ be clauses not containing variables of type ω different from $n : \omega$ and $\gamma \in \mathbf{N}$, then $\theta = [X_1 \setminus C_1, \cdots , X_\alpha \setminus C_\alpha]$ is a clause substitution. $c(n, \bar{u}, \bar{X})\theta\,[n \setminus \gamma]$ then denotes the normal form (under $\mathcal{R}$) under the assignment of n to γ after the application of θ.

For an explicit example of how exactly these clause substitutions and clause schema constructions work see Example 5.2 of [31]. In Ch. 3, Clauses where the basic element and where used to construct the clause set terms. We needed to generalize the concept of a clause in this chapter in order to construct a clause set term for proof schema. Rather than just clause sets being the basic element of the clause set term, we also need to add clause set variables. These are used to introduce clause sets when we construct the complete clause set schema. The recursion defined in the proof schema can require clause sets from two different parts of the proof schema to be introduced into each other.

Definition 4.6.2 (Clause set term [31]). *Clause-set terms are defined inductively using binary symbols $\oplus$ and $\otimes$ (which semantically correspond to conjunctions and disjunctions respectively) in the following way:*

- *Clause sets and clause-set variables are clause-set terms.*
- *If θ_1 and θ_2 are clause-set terms, then so are $\theta_1 \oplus \theta_2$ and $\theta_1 \otimes \theta_2$.*

Definition 4.6.3. *Let θ be a clause-set term not containing variables other than of type ι. Then the set of clauses $|\theta|$ assigned to θ is defined as $|C| = C$ for clause sets C, $|\theta_1 \otimes \theta_2|$ = $|\theta_1| \times |\theta_2|$, and $|\theta_1 \oplus \theta_2| = |\theta_1| \cup |\theta_2|$*

We borrow a simple example found in [31] which illustrate how these above definitions can be used to define the clause set and clause set terms.

Example 4.6.2 ([31]). *Let D_γ, where $\gamma \in \omega$, be the set of clauses $\{P(u_i) \vdash P(u_i) | i = 0, \cdots, \gamma\} \cup \left\{P(f^{i-1}(u_i)) \vdash P(f^i(u_i)) | i = 1, \cdots, \gamma\right\} \cup \{P(c); P(f^\gamma(c))\}$. We see that the number of clauses in the set, the number of variables, and the term depths increase with increasing γ. The following formalism is designed to deal with such clause sets and clause terms.*

Definition 4.6.4 ([31]). *Let θ be a clause-set term, $X_1, \cdots, X_\alpha$ clause-set variables and $C_1, \cdots, C_\alpha$ objects of appropriate type. Then $\theta\left[X_1 \setminus C_1, \cdots, X_\alpha \setminus C_\alpha\right]$ is called a clause-set term over $\{C_1, \cdots, C_\alpha\}$.*

Notice that the above definitions where constructed so that the schematic clause set terms are a strict generalization of the clause set terms of Ch. 3, in that every clause set term is also a schematic clause set term. We can consider the standard clause set term as a sort of basecase in terms of clause set construction.

Definition 4.6.5 (clause-set schema [31]). *A clause-set schema $C(n)$ is a structure $(C_1, \cdots, C_\alpha)$ together with a set of rewrite rules $\mathcal{R} = \mathcal{R}_1 \cup \cdots \cup \mathcal{R}_\alpha$, where the $\mathcal{R}_i$ (for $1 \leq i \leq \alpha$) are pairs of rewrite rules:*

$$C_i(0, \bar{u}_i, \bar{X}_i, \bar{X}_i) \rightarrow \Theta'_i \quad \textit{and} \quad C_i(k+1, \bar{u}_i, \bar{X}_i, \bar{X}_i) \Rightarrow \Theta_i$$

where Θ'_α is a clause-set term and the other Θ'_i and Θ_i are clause-set terms over terms in $C_1, \cdots, C_\alpha$, such that $V(\Theta'_i) \subseteq \left\{\bar{u}_i, \bar{X}_i, \bar{X}_i\right\}$ and $V(\Theta_i) \subseteq \left\{k, \bar{u}_i, \bar{X}_i, \bar{X}_i\right\}$. Furthermore, we assume that $C_i(\gamma, \bar{u}_i, \bar{X}_i, \bar{X}_i)$ is strongly normalizing for all $\gamma \in \mathbf{N}$.

In [31], a note was made contrasting the definitions of proof schemata and schematic language with Def. 4.6.5, namely that the prior definitions required primitive recursive definitions and thus, forcing strong normalization of the rewrite systems. In the case of definition Def. 4.6.5, there is no requirement for the clause-set schema to be primitive recursive, but only the requirement that the system is strongly normalizing, and thus it is allows more general rewriting than the previous definitions.

Theorem 4.6.1. *A clause-set schema $C(n)$ is unsatisfiable iff for all $\gamma \in \mathbb{N}$, $C(\gamma)\downarrow$ is unsatisfiable.*

Example 4.6.3. *For an example of a clause set schema, see Ch. 5, specifically Sec. 5.2*

The final substitution we need to define concerning clause sets is the substitution mapping clause set variables to clause sets. Let ϑ be a clause-set substitution, θ be a clause substitution and $\gamma \in \mathbb{N}$, then $C(\gamma)\downarrow$ denotes a clause set $|C|$ where C is a normal form of $C_1(n, \bar{u}_1, \bar{X}_1, \bar{X}_1)\vartheta\theta[n\backslash \gamma]$ w.r.t. $\mathcal{R}$ extended with the rewrite rules for defined function and predicate symbols.

Up to this point we have provided all the information concerning clause sets for proof schema which will be needed for the schematic CERES method. For more detail on each of the individual definitions see [31]. We now move on to defining the resolution schema and resolution refutations using clause schemata.

Definition 4.6.6 (resolution term [31])**.** *Clause schemata are resolution terms; if ρ_1 and ρ_2 are resolution terms, then $r(\rho_1; \rho_2; P)$ is a resolution term, where P is an atom formula schema. $r(\rho_1; \rho_2; P)$ expresses the result obtained by resolving the clauses derived by ρ_1 and ρ_2 , where P is the resolved atom (still without specification of the most general unifier).*

Even though a resolution refutation using standard sequent syntax can be written in a recursive way (we give examples of this in Sec. 5.4, Defining such a structure is much more difficult then define a term structure as provided in the above definition. Examples of resolution terms and resolution schemata will be provided in Sec. 5.4 as well.

Definition 4.6.7 (resolution proof schema [31])**.** *A resolution proof schema $R(n)$ is a structure $(\varrho_1, ..., \varrho_\alpha)$ together with a set of rewrite rules $\mathcal{R} = \mathcal{R}_1 \cup \cdots \cup \mathcal{R}_\alpha$, where the $\mathcal{R}_i$ (for $1 \leq i \leq \alpha$) are pairs of rewrite rules*

$$\varrho_i(0, \bar{u}_i, \bar{X}_i) \to \rho_i' \quad and \varrho_i(k+1, \bar{u}_i, \bar{X}_i) \to \rho_i$$

where ρ_i' is a resolution term over terms of the form $\varrho_j(a_j, \bar{t}_j, \bar{C}_j)$, and ρ_i is a resolution term over terms of the form $\varrho_j(a_j, \bar{t}_j, \bar{C}_j)$ and $\varrho_j(k, \bar{t}_j, \bar{C}_j)$ for $1 \leq i < j \leq \alpha$.

An interesting note pertaining to both resolution terms and the resolution proof schema is that the required substitutions for the unification steps are not defined. It was a choice made during the construction of the system that unification substitutions would be defined at the global level rather than the local level. Thus, rather then each step having a unifier, there is a global unifier (a second order unifier) which works for every step of the resolution proof schema.

Definition 4.6.8 (substitution schema [31])**.** *Let $u_1, ..., u_\alpha$ be schematic variable symbols of type $\omega \to \iota$ and $t_1, \cdots, t_\alpha$ be term schemata containing only k as variable of the ω sort. Then a substitution schema is an expression of the form $[u_1 \backslash \lambda k.t_1, \cdots, u_\alpha \backslash \lambda k.t_\alpha]$.*

To provide a better understanding of how one ought to use the substitution schema consider that for every $\gamma \in \mathbb{N}$ the substitution schema can be defined to substitute terms into the variables of the variable schema the substitution schema is defined for. In [31] the following semantic definition is provided for substitution schema. for all $\gamma \in \mathbb{N}$ we have a substitution $[u_1(\gamma) \backslash t_1 \downarrow \gamma, \cdots, u_\alpha(\gamma) \backslash t_\alpha \downarrow \gamma]$. Let $R(n) = (\varrho_1, ..., \varrho_\alpha)$ be a resolution proof schema, ϑ be a clause substitution, θ be a substitution schema and $\gamma \in \mathbb{N}$, then $R(\gamma)\downarrow$ denotes a resolution term which is normal form of $\varrho_1(n, \bar{u}_1, \bar{X}_1)\theta\vartheta\,[n \setminus \gamma]$ w.r.t. $\mathcal{R}$ extended with rewrite rules for defined function and predicate symbols.

We now define the application of a resolution proof schema to a clause set schema in order to construction resolution derivations and resolution refutations.

Definition 4.6.9 (Resolvent [31])**.** *Let $C : \Gamma \vdash \Delta$ and $D : \Pi \vdash \Lambda$ be clauses not containing arithmetic variables and let P be an atom. Then the clause $res(C, D, P) = \Gamma \vdash \Pi \setminus P \Delta \setminus P \vdash \Lambda$ is called the resolvent of C and D on P , where $\Pi \setminus P$ (resp. $\Delta \setminus P$) denotes the multi-set of atoms in Π (resp. Δ) after removal of all occurrences of P . In case P does not occur in Δ and Π, $res(C, D, P)$ is called a pseudo-resolvent.*

Definition 4.6.10 (Resolution Deduction [31])**.** *If C is a clause then C is a resolution deduction with end-sequent C. If ρ_1 and ρ_2 are resolution deductions with end-sequents respectively C_1 and C_2 , such that for an atom P there is a resolvent $res(C_1, C_2, P) = D$, then $\rho = r(\rho_1, \rho_2, P)$ is a resolution deduction with end-sequent D. Let C be a set of clauses. If the set of all clauses occurring in ρ is an instantiation of C, then ρ is called a resolution deduction from C and if $D = \vdash$ then ρ is called a resolution refutation of C.*

Notice that in the definition of as resolution deduction we switch from res terms to r terms where the r are representing resolution of resolvents rather than pseudo resolvents . A resolution deduction r is a consistent structure of resolution terms which are resolvents.

Definition 4.6.11 (Refutation Schema [31])**.** *A resolution proof schema $R(n)$ is called a resolution deduction schema from a clause-set schema $C(n)$ if there exist a clause substitution and a substitution schema such that for every $\gamma \in \mathbb{N}$, $R(\gamma) \downarrow$ is a resolution deduction from $C(\gamma) \downarrow$. Furthermore, if for all $\gamma \in \mathbb{N}$, $R(\gamma) \downarrow$ is a resolution refutation of $C(\gamma) \downarrow$, then $R(n)$ is called a resolution refutation schema of $C(n)$.*

Proposition 4.6.1. *If R is a resolution refutation schema of a clause-set schema C then C is unsatisfiable.*

Proof. See [31]. □

We will refer to the above defined resolution calculus as **RS**.

Corollary 4.6.1. *The resolution calculus* **RS** *is sound.*

We do not provide an example of a resolution refutation schema in this section however one can find an example in the following chapter, specifically in Sec. 5.4. As noted in [31], completeness does not hold for **RS**, since unsatisfiability of schemata is a property which is not semi-decidable even for propositional schemata.

4.7 The CERES Method for First-Order Schemata

So far, all the results presented in this chapter can be used to define proof schemata, such that after instantiating the free parameter, the cut-elimination method presented in Ch. 3 or reductive cut-elimination can be applied, resulting in a cut free proof. However, this only gives a characterisation of cut elimination for the instances of a given proof schemata rather than of the entire proof schemata. In this section we take the pieces presented so far and put them together in order to construct a method of cut-elimination which does not rely on the instantiation of the proof schemata to provide a cut-free proof.

We will base the cut-elimination method provided in this section on the CERES method rather than Gentzen's reductive cut elimination method being that it is not entirely clear how one ought to proceed with reductive cut elimination on proof schemata in certain instances. For example, the following sub-proof of a proof schemata was provided in [31]:

$$\dfrac{\dfrac{\psi_1(a_1)}{\Gamma \vdash \Delta, C} \qquad \dfrac{\psi_2(a_2)}{\Pi, C \vdash \Lambda}}{\Gamma, \Pi \vdash \Delta, \Lambda}\, cut$$

The problem being that the cut formula will have to pass through the proof links when reductive cut elimination is applied. As mentioned in [31], this is a problem found in cyclic proof systems in general and has been mentioned before in other work [21]. However, the CERES method takes a different route and collects the cut structure of the proof schema, this allows us to avoid passing the cut formulae through proof links. Instead of the proof links causing problem, the main issue is that we essentially need to construct an entirely new proof from the schematic characteristic clause set. The majority of this dissertation will be focused on dealing with these issues. The rest of this section will be focused on redefining the definitions found in Ch. 3 for proof schemata.

The Characteristic Term

The construction of the characteristic term as defined in Ch. 3 required inductively following the formula occurrences of cut formula ancestors up the proof tree to the leaves. The problem with doing something similar in proof schemata is that the concept of ancestors and formula occurrence is more complex. This problem occurs because at some point in a given proof schema a formula occurrence might be an ancestor of a cut formula and at other points it might not be. A good example of this would be if the upper most proof schema pair does not have any cut formulae, but everyone of the proof schema pairs below it does have cut formulae. Then it can occur, that is a formula occurring at the top of the proof schema is not a cut ancestor, but it is a cut ancestor at ever point below in the proof schema. In [31], additional machinery was added to the characteristic term to deal with this problem of tracking formula occurrences. A set Ω of formula occurrences from the end-sequent of an LKS-proof π is called *a configuration for* π . Specifically, the only configurations which are of interest are configurations which track of all cut-ancestors in a proof schema ψ as well as the propagation of the cut-ancestors through the proof links. A configuration Ω for π is called relevant w.r.t. a proof schema Ψ if π is a proof in Ψ and there is a $\gamma \in \mathbb{N}$ such that π induces a subproof π of $\Psi \downarrow \gamma$ such that the occurrences in Ω correspond to cut-ancestors below π [29]. Note that the set of relevant cut-configurations can be computed given a proof schema Ψ.

With the addition of the concept of configuration and relevant configuration we can represent a characteristic term of a proof link in our language. Each pair of proof link symbol φ and configuration Ω (of course only configurations relevant to the given proof link) will be given a *clause set symbol* $cl^{\varphi,\Omega}$. A clause set symbol $cl^{\varphi,\Omega}(a)$, where a is an arithmetic term, is intended to be read as the clause set of $\varphi(a)$ given the configuration Ω. We now proceed to defining the characteristic term.

Definition 4.7.1 (Characteristic term [31]). *Let π be an* $\mathbf{LKS}$*-proof and Ω a configuration. In the following, by Γ_Ω , Δ_Ω and Γ_C , Δ_C we will denote multisets of formulas of Ω- and cut-ancestors respectively. Let r be an inference in π. We define the clause-set term $\Theta^{\pi,\Omega}_r$ inductively:*

- *if r is an axiom of the form $\Gamma_\Omega, \Gamma_C, \Gamma \vdash \Delta_\Omega, \Delta_C, \Delta$, then $\Theta^{\pi,\Omega}_r = \{\Gamma_\Omega, \Gamma_C \vdash \Delta_\Omega, \Delta_C\}$*
- *if r is a proof link of the form*

$$\frac{\psi(a, \bar{u})}{\Gamma_\Omega, \Gamma_C, \Gamma \vdash \Delta_\Omega, \Delta_C, \Delta}$$

  then define Ω' as the set of formula occurrences from $\Gamma_\Omega, \Gamma_C \vdash \Delta_\Omega, \Delta_C$ and $\Theta^{\pi,\Omega}_r = cl^{\psi,\Omega}(a, \bar{u})$
- *if r is a unary rule with immediate predecessor r' , then $\Theta^{\pi,\Omega}_r = \Theta^{\pi,\Omega}_{r'}$*
- *if r is a binary rule with immediate predecessors r_1, r_2, then*
  - *if the auxiliary formulas of r are Ω- or cut-ancestors, then $\Theta^{\pi,\Omega}_r = \Theta^{\pi,\Omega}_{r_1} \oplus \Theta^{\pi,\Omega}_{r_2}$*
  - *otherwise, $\Theta^{\pi,\Omega}_r = \Theta^{\pi,\Omega}_{r_1} \otimes \Theta^{\pi,\Omega}_{r_2}$*

Finally, define $\Theta^{\pi,\Omega} = \Theta^{\pi,\Omega}_{r_0}$ where r_0 is the last inference in π and $\Theta^\pi = \Theta^{\pi,\emptyset}$. We call Θ^π the characteristic term of π.

A *normal* clause set term is a clause set term that does not contain clause set symbols defined predicate or defined function symbols.

Definition 4.7.2 (Characteristic term schema [31]). *Let $\Psi = \langle \psi_1, \cdots, \psi_\alpha \rangle$ be a proof schema. We define the rewrite rules for clause-set symbols for all proof symbols ψ_β and configurations Ω:*

$$cl^{\psi_\beta,\Omega}(0, \bar{u}) \rightarrow \Theta^{\pi_\beta,\Omega} \quad \textit{and} \quad cl^{\psi_\beta,\Omega}(k+1, \bar{u}) \rightarrow \Theta^{\nu_\beta(k),\Omega}$$

where $1 \leq \beta \leq \alpha$. Next, let $\gamma \in \mathbb{N}$ and let $cl^{\psi_\beta,\Omega} \downarrow_\gamma$ be a normal form of $cl^{\psi_\beta,\Omega}(\gamma, \bar{u})$ under the rewrite system just given extended by rewrite rules for defined function and predicate symbols. Then define $\Theta^{\psi_\beta,\Omega} = cl^{\psi_\beta,\Omega}$ and $\Theta^{\Psi,\Omega} = \Theta^{\psi_1,\Omega}$. Lastly, the characteristic term schema is defined as $\Theta^\Psi = \Theta^{\Psi,\emptyset}$

The characteristic term schema defined above is a special case of Def. 4.6.5. The following proposition proven in [31], shows that the Characteristic term schema is strongly normalizing.

Proposition 4.7.1. *Let Ψ be a proof schema and Θ^{Ψ} be a characteristic term schema of Ψ. Then Θ^{Ψ} is strongly normalizing .*

Proof. See [31] for a proof. □

The following few propositions prove that the expected relationship which exists between standard clause set terms and clause schema and their schematic counter parts.

Proposition 4.7.2. *Let Ψ be a proof schema and Ω be a configuration, then $\Theta^{\Psi_{\beta},\Omega} \downarrow \gamma$ is a normal clause-set term for all $1 \leq \beta \leq \alpha$ and $\gamma \in \mathbb{N}$. Hence $\Theta^{\Psi} \downarrow \gamma$ is a normal clause-set term.*

Proposition 4.7.3. *Let Ψ be a proof schema, Ω be a configuration and $\gamma \in \mathbb{N}$. Then $\Theta^{\Psi\downarrow_{\gamma},\Omega} = \Theta^{\Psi,\Omega} \downarrow_{\gamma}$.*

The above proposition, i.e. Prop. 4.7.3, is important for defining the relationship between extraction of the schematic characteristic clause set term prior to instantiation of the free parameter and after the instantiation of the free parameter. It essentially states that the same term will be extracted if we first instantiate the proof schema and then extract the term and if we first extract the term and then instantiate the extracted term. This is important for comparing this method with the original CERES method.

Now we can define the schematic characteristic clause set from the characteristic clause set terms schema. As defined in [31], For a normal LKS-proof π and configuration Ω, $CL(\pi, \Omega) = |\Theta^{\pi,\Omega}|$. We define the standard characteristic clause set $CL(\pi) = CL(\pi, \emptyset)$ and the schematic characteristic clause set $CL(\Psi) \downarrow \gamma = |\Theta^{\Psi} \downarrow_{\gamma} |$.

The next two propositions from [31] are the most important of all, That is that the schematic characteristic clause set developed in this section is always unsatisfiable and that the instances of the schematic characteristic clause set are also always unsatisfiable.

Proposition 4.7.4. *Let π be a normal* **LKS***-proof. Then $CL(\pi)$ is unsatisfiable.*

Proof. This is trivial to prove being that we have shown that **LKS**-proof can be corresponded with **LK**-proof. The extracted characteristic clause sets for **LK**-proofs are always unsatisfiable. The proof of this result, that the characteristic clause sets are always unsatisfiable for **LK**-proofs, Can be found in [14]. □

Proposition 4.7.5. *For a proof schema Ψ , $CL(\Psi) \downarrow_{\gamma}$ is unsatisfiable for all $\gamma \in \mathbb{N}$.*

Now that we have constructed the schematic clause set as well as a schematic resolution refutation language we can finally construct an ACNF of a proof schema. The difference between the ACNFs we construct in this chapter when compared to the ACNF of Ch. 3 is that the schematic ACNF does not have proof projections . As mentioned earlier, the concept of proof projection was dropped from [31] due to its complexity in the schematic case and the fact that it is essentially useless. This uselessness is derived from the fact that the projection schema must remain a separate object until the instantiation of the free parameter. Thus, we do not gain much information of the entire schematic structure of cut-free proof schema, but rather only information as to the structure of individual instantiations. This is very similar to the results of the Fürstenberg analysis. The ACNF definition provided for schematic CERES deals with a completely constructed resolution refutation using the substitution schema and the characteristic clause set schema.

Definition 4.7.3 (ACNF Schema [31]). *Let Ψ be a proof schema with end-sequent $S(n)$, and $\mathcal{R}$ be a resolution refutation schema of the schematic characteristic clause set $CL(\Psi)$. Let ϑ be the substitution schema corresponding to $\mathcal{R}$. Then the pair $(\mathcal{R}, \vartheta)$ is an ACNF schema of Ψ.*

An example ACNF can be found in Appendix B. We will not go into a discussion of the concept of Herbrand systems in the chapter being that we use throughout this dissertation an older definition of what a schematic Herbrand sequent is and how it is constructed. However, for readers interested in this newer notion of a Herbrand sequent see [31]. The definitions and constructions of this chapter are the foundation for the rest of this dissertation.

We will now end the chapter by providing a characteristic term for the proof φ found in Appendix A.1.

Example 4.7.1. *We start at the end sequent of φ construct of the characteristic term. The end sequent will be marked with an ES. Only the proof φ will be considered from the proofs provided in Appendix A.1 and thus, we start with an empty configuration and the following derivation:*

$$\Theta^{\varphi} = \Theta^{\varphi,\emptyset} = \Theta^{\varphi,\emptyset}_{es}.$$

Using the procedure provided in 4.7.1 and following the inference steps found in the **LKS** *proof of Appendix A.1, we get the following derivation.*

$$\Theta^{\varphi,\emptyset}_{es(n+1)} = \Theta^{\varphi,\emptyset}_{c:l} = \Theta^{\varphi,\emptyset}_{cut} = \Theta^{\varphi,\emptyset}_{cut'} \oplus \Theta^{\varphi,\Omega}_{es(n)}$$

where $\Omega \equiv \{\exists x \forall y (((x \leq y) \rightarrow n = f(y)) \ \vee f(y) < n)\}$. $\Theta^{\varphi,\Omega}_{es(n)}$ can be replaced by $cl^{\varphi,\Omega}(n)$ by definition 4.7.1, and thus we have so far derived that the following holds:

$$\Theta^{\varphi} \equiv \Theta^{\varphi,\emptyset}_{cut'} \oplus cl^{\varphi,\Omega}(n)$$

We continue expanding on the left side of the $\oplus$ and get:

$$(\Theta^{\varphi,\emptyset}_{cut'} \oplus \Theta^{\varphi,\emptyset}_{\exists:l}) \oplus cl^{\varphi,\Omega}(n)$$

In the proof φ, starting at $\Theta^{\varphi,\emptyset}_{\exists:l}$ there is a long list of unary inference rules which can be skipped by Def. 4.7.1. We skip this rules and go directly to the next binary inference rule.

$$(\Theta^{\varphi,\emptyset}_{cut} \oplus \Theta^{\varphi,\emptyset}_{\rightarrow:l}) \oplus cl^{\varphi,\Omega}(n) \equiv (\Theta^{\varphi,\emptyset}_{cut} \oplus (\Theta^{\varphi,\emptyset}_{\rightarrow:l} \otimes \Theta^{\varphi,\emptyset}_{Ax})) \oplus cl^{\varphi,\Omega}(n)$$

where we can replace the term $\Theta^{\varphi,\emptyset}_{Ax}$ by the sequent $\vdash \alpha \leq g(\alpha)$ in the following term.

$$(\Theta^{\varphi,\emptyset}_{cut} \oplus (\Theta^{\varphi,\emptyset}_{\rightarrow:l} \otimes \{\alpha \leq g(\alpha)\})) \oplus cl^{\varphi,\Omega}(n)$$

Next we can unroll the term $\Theta^{\varphi,\emptyset}_{\rightarrow:l}$ and get the following result:

$$(\Theta^{\varphi,\emptyset}_{cut} \oplus ((\Theta^{\varphi,\emptyset}_{Ax_1} \oplus \Theta^{\varphi,\emptyset}_{Ax_2}) \oplus \{\alpha \leq g(\alpha)\})) \oplus cl^{\varphi,\Omega}(n)$$

Replacing the terms as we did with the previous axioms we get the following term as the result.

$$(\Theta^{\varphi,\emptyset}_{cut} \oplus ((\{n+1 = f(\alpha), n+1 = f(g(\alpha)) \vdash\} \oplus \{\vdash \alpha \leq \alpha\}) \oplus \{\alpha \leq g(\alpha)\})) \oplus cl^{\varphi,\Omega}(n)$$

The only part left to unroll is the left most characteristic clause term $\Theta^{\varphi,\emptyset}_{cut}$:

$$\Theta^{\varphi,\emptyset}_{cut} \equiv \Theta^{\varphi,\emptyset}_{\exists:l} \oplus \Theta^{\varphi,\emptyset}_{\exists:l}$$

The term $\Theta^{\varphi,\emptyset}_{\exists:l}$ on the right side of the $\oplus$ only has unary rules applied above it and thus can be replaced by the axiom.

$$\Theta^{\varphi,\emptyset}_{cut} \equiv \Theta^{\varphi,\emptyset}_{\exists:l} \oplus \{f(\alpha) < n+1, \alpha \leq \beta \vdash n = f(\beta), f(\beta) < n\}$$

The $\Theta^{\varphi,\emptyset}_{\exists:l}$ on the left side of the $\oplus$ can be replaced by $\Theta^{\varphi,\emptyset}_{\vee:l}$, a binary inference rule at the end of a chain of unary inference rules.

$$\Theta^{\varphi,\emptyset}_{\vee:l} \oplus \{f(\alpha) < n+1, \alpha \leq \beta \vdash n = f(\beta), f(\beta) < n\}$$

The $\vee : l$ inference rule act on end sequent ancestors and thus can be replaced with

$$(\Theta^{\varphi,\emptyset}_{\rightarrow:l} \otimes \Theta^{\varphi,\emptyset}_{Ax}) \oplus \{f(\alpha) < n+1, \alpha \leq \beta \vdash n = f(\beta), f(\beta) < n\}$$

The term $\Theta^{\varphi,\emptyset}_{Ax}$ is replaced with $\{\vdash f(\beta) < n+1\}$

$$(\Theta^{\varphi,\emptyset}_{\rightarrow:l} \otimes \{\vdash f(\beta) < n+1\}) \oplus \{f(\alpha) < n+1, \alpha \leq \beta \vdash n = f(\beta), f(\beta) < n\}$$

Again, the term $\Theta^{\varphi,\emptyset}_{\rightarrow:l}$ is a binary rule acting on end sequent ancestors and can be replaced with the following:

$$((\Theta^{\varphi,\emptyset}_{Ax_1} \otimes \Theta^{\varphi,\emptyset}_{Ax_2}) \otimes \{\vdash f(\beta) < n+1\}) \oplus \{f(\alpha) < n+1, \alpha \leq \beta \vdash n = f(\beta), f(\beta) < n\}$$

where $\Theta^{\varphi,\emptyset}_{Ax_1} \equiv \{\vdash n+1 = f(\beta)\}$ and $\Theta^{\varphi,\emptyset}_{Ax_2} \equiv \{\alpha \leq \beta \vdash\}$
And thus, we finally derive:

$$(((\{\vdash n+1 = f(\beta)\} \otimes \{\alpha \leq \beta \vdash\}) \otimes \{\vdash f(\beta) < n+1\})\oplus$$
$$\{f(\alpha) < n+1, \alpha \leq \beta \vdash n = f(\beta), f(\beta) < n\})$$

The full term is as follows:

$$\Theta^{\varphi} \equiv ((((\{\vdash n+1 = f(\beta)\} \otimes \{\alpha \leq \beta \vdash\}) \otimes \{\vdash f(\beta) < n+1\})\oplus$$
$$\{f(\alpha) < n+1, \alpha \leq \beta \vdash n = f(\beta), f(\beta) < n\})$$

$$\oplus((\{n+1=f(\alpha),n+1=f(g(\alpha))\vdash\}\oplus\{\vdash\alpha\leq\alpha\})\oplus\{\alpha\leq g(\alpha)\}))\oplus cl^{\varphi,\Omega}(n)$$

We can derive a normal clause set term for Θ^{φ} by iterating over $cl^{\varphi,\Omega}(n)$ which differs slightly from the clause set derived on the lefthand side because it has the configuration Ω. Thus, we get the following derivation for $cl^{\varphi,\Omega}(n)$:

$$cl^{\varphi,\Omega}(n+1)\equiv((((\{n+1=f(\beta)\vdash n+1=f(\beta)\}\oplus\{\alpha\leq\beta\vdash\alpha\leq\beta\})\otimes$$
$$\{f(\beta)<n+1\vdash f(\beta)<n+1\})\oplus$$
$$\{f(\alpha)<n+1,\alpha\leq\beta\vdash n=f(\beta),f(\beta)<n\})$$
$$\oplus((\{n+1=f(\alpha),n+1=f(g(\alpha))\vdash\}\oplus$$
$$\{\vdash\alpha\leq\alpha\})\oplus\{\alpha\leq g(\alpha)\}))\oplus cl^{\varphi,\Omega}(n)$$

We switch an $\otimes$ for an $\oplus$ given the configuration Ω. Thus, the "almost" normal clause set term we get is as follows:

$$\Theta^{\varphi}\equiv((((\{\vdash n+1=f(\beta)\}\otimes\{\alpha\leq\beta\vdash\})\otimes\{\vdash f(\beta)<n+1\})\oplus$$
$$\{f(\alpha)<n+1,\alpha\leq\beta\vdash n=f(\beta),f(\beta)<n\})$$
$$\oplus((\{n+1=f(\alpha),n+1=f(g(\alpha))\vdash\}\oplus\{\vdash\alpha\leq\alpha\})\oplus\{\alpha\leq g(\alpha)\}))\oplus$$
$$\oplus_{i=1}^{n-1}(((((\{i=f(\beta)\vdash i=f(\beta)\}\oplus\{\alpha\leq\beta\vdash\alpha\leq\beta\})\oplus\{f(\beta)<i\vdash f(\beta)<i\})\oplus$$
$$\{f(\alpha)<i,\alpha\leq\beta\vdash n=f(\beta),f(\beta)<n\})$$
$$\oplus((\{i=f(\alpha),i=f(g(\alpha))\vdash\}\oplus\{\vdash\alpha\leq\alpha\})\oplus\{\alpha\leq g(\alpha)\})))\oplus cl^{\varphi,\Omega}(0)$$

The last part to unroll is the clause set symbol $cl^{\varphi,\Omega}(0)$ which is as follows:

$$cl^{\varphi,\Omega}(0)\equiv(((\{0=f(\alpha),0=f(g(\alpha))\vdash\}\oplus\{\vdash\alpha\leq g(\alpha)\})\ \oplus\{\vdash\alpha\leq\alpha\})$$
$$\oplus\{f(g(\alpha))<0\})\oplus\{f(\alpha)<0\}$$

Thus the entire normal term put together is the following:

$$\Theta^{\varphi}\equiv((((\{\vdash n+1=f(\beta)\}\otimes\{\alpha\leq\beta\vdash\})\otimes\{\vdash f(\beta)<n+1\})\oplus$$
$$\{f(\alpha)<n+1,\alpha\leq\beta\vdash n=f(\beta),f(\beta)<n\})$$
$$\oplus((\{n+1=f(\alpha),n+1=f(g(\alpha))\vdash\}\oplus\{\vdash\alpha\leq\alpha\})\oplus\{\alpha\leq g(\alpha)\}))\oplus$$
$$\oplus_{i=1}^{n-1}(((((\{i+1=f(\beta)\vdash i+1=f(\beta)\}\oplus\{\alpha\leq\beta\vdash\alpha\leq\beta\})\oplus\{f(\beta)<i+1\vdash f(\beta)<i+1\})\oplus$$
$$\{f(\alpha)<i+1,\alpha\leq\beta\vdash i=f(\beta),f(\beta)<i\})$$
$$\oplus((\{i+1=f(\alpha),i+1=f(g(\alpha))\vdash\}\oplus\{\vdash\alpha\leq\alpha\})\oplus\{\alpha\leq g(\alpha)\})))\oplus$$
$$((((\{0=f(\alpha),0=f(g(\alpha))\vdash\}\oplus\{\vdash\alpha\leq g(\alpha)\})\ \oplus\{\vdash\alpha\leq\alpha\})$$
$$\oplus\{f(g(\alpha))<0\vdash\})\oplus\{f(\alpha)<0\vdash\})$$

This normalized clause set is different from the clause set extracted in Ch. 5 being that it does not take into consideration the configuration that is passed down the proof from ψ. We can write this clause set in a much more digestible form as follows (tautology elimination has been applied):

$$\alpha \leq \beta \vdash n+1 = f(\beta), f(\beta) < n+1$$

$$f(\alpha) < n+1, \alpha \leq \beta \vdash n = f(\beta), f(\beta) < n$$

$$n+1 = f(\alpha), n+1 = f(g(\alpha)) \vdash$$

$$\vdash \alpha \leq \alpha$$

$$\vdash \alpha \leq g(\alpha)$$

$$\bigwedge_{i=1}^{n-1} f(\alpha) < i+1, \alpha \leq \beta \vdash i = f(\beta), f(\beta) < i$$

$$\bigwedge_{i=1}^{n-1} i+1 = f(\alpha), i+1 = f(g(\alpha)) \vdash$$

$$0 = f(\alpha), 0 = f(g(\alpha)) \vdash$$

$$f(g(\alpha)) < 0 \vdash$$

$$f(\alpha) < 0 \vdash$$

We will not construct an ANCF of this proof in this chapter, but instead refer to the Resolution refutation and substitution schema found in Ch. 5 and the instantiated ACNF found in Appendix A.2.

CHAPTER 5

Proof Analysis by Ceres$_S$: the ECS-Schema

The schematic CERES method explained in Ch. 4 is a framework for performing cut-elimination on proof schemata defined in the **LKS** calculus. A simple example was provided in [30] and it was discussed, without the technical details, that it is possible to formalize refutations with non-elementary length . The example provided in [30] did not emphasize the complexity of the refutation step being that the refutation used a complex term structure and thus, a complex unifier, but had a very simple tree structure. In reference to a proof schema which has a refutation with non-elementary growth rate there is the Aronkov proof sequence which can be found in [52] section 8.2. However, it was quite clear from the characteristic clause set how the clause set ought to be refuted. Also, the branching of the resolution refutation is quite simple in that it grows depthwise much faster than it grows breathwise. We will see in Ch. 6 that when the resolution refutation grows breathwise much faster than depthwise the frame work of [31] breaks down, especially that in Ch. 6, the growth of the breath is dependent on the free parameter, and not to mention that the required term structure is complex and dependent on the free parameters as well.

In reference to the Aronkov proof sequence again, it is clearly an example developed to show a theoretic point rather than an analysis of a mathematical argument analysed within the schematic CERES framework. More or less, it is a bit artificial of a proof schema. In this chapter we take a simple mathematical statement based on the pigeonhole principle and apply the steps of the schematic CERES method outlined in Ch. 4 to a proof schema whose fixed point is the full statement. As mentioned in Ch. 1, we were not sure, when we started the proof analysis of this proof schema, whether or not it would fit the framework of the schematic CERES method, and to top it off, we actually developed the proof schema with the intention to find another proof schema which did not fit into the framework of Ch. 4. However, in the process of analysing the proof schema we found out that, to our surprise, there exists a resolution refutation which fit the framework of Ch. 4. This refutation will be provided in this chapter. An interesting observation is that, like the Aronkov sequence and the proof schema from [30], the signature of the term language only contains monadic and constant function symbols. Though, this line of thought has not been properly investigated, it seems that there is a connection between the term signature and the complexity of the resolution calculus needed for performing cut-elimination. This is similar to the jump in complexity we see in classical first order logic [18]. For the rest of this chapter, we will go through the steps of proof analysis using schematic CERES outlining which parts of Ch. 4 are used at which point of the proof analysis. The mathematical statement we will be analysing is as follows:

Statement 1 (Eventually Constant Statement (ECS))**.** *Let f be a total monotonically decreasing function over the natural numbers, then f is eventually constant.*

The ECS statement is a weakened version of the NiA-Schema we will see in Ch. 6.

To reiterate what was stated earlier in the introduction to this dissertation, the SPASS Theorem Prover [56] was used, exclusively, to aid in the construction of resolution refutation found in this chapter. SPASS was not used to directly find the schematic refutation by automated means, but rather by using the software to refute instances of the characteristic clause set we were able to construct a refutation of the schematic clause set. Using these refutations of instances of the clause set found using SPASS, we found patterns in the derived clauses used by SPASS which helped us construct the induction invariants needed for showing refutability of the clause set within the language of Ch. 4. These empirical results as well as analysis will be explained in more detail in Ch. 10. There is no special reason why we chose to use SPASS almost exclusively rather than other theorem provers, other then the output of this prover is one of the most readable output formats.

The rest of this chapter is structured as follows, Sec. 5.1 will be an outline of our **LKS** proof of the eventually constant statement, in Sec. 5.2 we provide the clause set of the proof found in A.1 , in Sec. 5.3 we take a refutation derived using SPASS [56] and analyse the structure, in Sec. 5.4 We provide the resolution refutation schema of the clause set in Sec. 5.2, in Sec. 5.5 we provide the sequent calculus representation of the projections for the proof of A.1. And finally, in Sec. 5.6 we provide a Herbrand sequent which is easily derivable from analysis of the projections and the resolution refutation.

5.1 Outline of Sequent Calculus Proof in LKS

A full formal proof of the Statement 1 in the **LKS** calculus of Sec 4.2 can be found in A.1. The proof uses one defined predicate symbol (iterated $\vee$), but no defined term symbols (See Sec. 4.4). To aid the understanding and the motivation behind the arguments found in the formal proof of Statement 1 in A.1, rather than writing out every logical step as is done in A.1 we instead skip the steps which would be trivial to a human and focus on the steps which are important for understanding; with such a simple statement, even the proof of this section is overkill in terms of human understanding. In the formal proof found in A.1, the end sequent, i.e. our theorem, can be stated as follows:

$$\begin{gathered}\forall x (\bigvee_{i=0}^{n+1} i = f(x)),\\ \forall x \forall y \Big(x \leq y \rightarrow f(y) \leq f(x) \Big) \vdash \\ \exists x \forall y (x \leq y \rightarrow f(x) = f(y))\end{gathered}$$

We write the same statement informally (mathematically) as as follows:

Theorem 5.1.1. *Given a total monotonically decreasing function $f : \mathbb{N} \rightarrow \{0, \cdots, n\}$, for $n \in \mathbb{N}$, then there exists an $x \in \mathbb{N}$ such that for all $y \in \mathbb{N}$, where $x \leq y$, it is the case that $f(x) = f(y)$.*

Our proof of this statement uses the same cut formulae as the **LKS** proof, however, they are obfuscated by the mathematical language. However, if you look closely, you will notice that the cuts are essential to the induction hypothesis of the proof. The principal schema of cuts used to prove the theorem is as follows:

$$\exists x \forall y \left(((x \leq y) \rightarrow n+1 \sim f(y)) \vee f(y) < n+1 \right) \tag{5.1}$$

Proof. We prove the the theorem by induction on the size of the range of the function f. When $n = 0$ the range of the function is one and the theorem trivially holds since no matter which value in the domain is chosen f will map that value to zero. Now for the induction hypothesis let us assume that the theorem holds for functions f such that the size of the range is $\leq m$. Using this assumption, we show that it holds for a range of size $m + 1$. We know that the f is monotonically decreasing and that the range is $\{0, \cdots, m\}$ when the size of the range is $m + 1$. If we assume the canonical ordering on the elements (the linear ordering of $\mathbb{N}$) of the range, it is quite obvious that if for some $z \in \mathbb{N}$, $f(z) = m - 1$ (or any value in the range $\{0, \cdots, m - 1\}$) then either there exists $w < z$ such that $f(w) = m$ or for all $w \in \mathbb{N}$, $f(w) \neq m$. If the latter holds, then we are done because it is equivalent to the case when the range is of size m, of which holds by the induction hypothesis. If the former holds, we take the largest w such that $f(w) = m$, at $w + 1$ the function is almost equivalent to the case when the range is of size m for all r, such that $w + 1 \leq r$, however, it is shifted. Though, being that the theorem is about the shape of the end of the function rather than the shifts at the beginning, this case holds by the induction hypothesis as well. The only case left is a function which only maps to m, but this case is trivial and is almost equivalent to the base case, just need to map m to 0. Thus, by induction we have proven the theorem. $\square$

5.2 Clause Set for a formal proof of ECS

In Ch. 4 , an example of the first-order CERES method [14] being applied to a formal proof is provided, Ex. 3.3.1. In this section we will ignore the extraction of the projections and focus on the construction of the characteristic clause set of the formal proof. As outlined in Ex. 3.3.1, in order to find the characteristic clause set we must locate the ancestors of the cut formulae and follow the ancestral path all the way to the leaves of the proof tree. Mattering on which rules are applied and on what type of formula they are applied (either end sequent ancestor or cut ancestor) we either union the clause sets on the two branches going into the rule (if the rule is binary), Cartesian product the clause sets on the two branches going into the rule, or use the same clause set at the next step if the rule is unary. The precise rules for when to apply the union, Cartesian product, or unary rule can be found in Def. 3.3.2 for first-order CERES and in Sec. 4.6 for schematic CERES. The major difference between extraction of the clause set in first-order

CERES when compared to schematic CERES is that rather than having a fixed finite proof tree, as in the first-order CERES, schematic CERES has a recursive proof tree indexed by a parameter . To represent these recursive proofs we use proof links (see Def. 4.5.2). The characteristic clause set can require calling the previous step of the recursion (i.e. the case when the parameter is set to $n+1$ might need to call the case when the parameter is set to n) to be constructed and thus, we would need to define how to deal with a characteristic clause set which requires recursion to build it. Also, that the same proof link can have different cut-ancestors in the main sequent depending on when it is used in the various stages of the proof. These cut ancestor configurations are just called *configurations* . Luckily, this additional difficulty is not present in the formal proof of A.1.

We will label Eq. 5.1 as $cut(n)$ for the cut configurations of the clause set terms. We will extract the clause set terms using the formal proof of A.1. We will then unfold these terms into a more traditional clause set, namely $C^{ECS}(n)$. A small note ought to be made about the CERES method in general, namely, that the method requires proof Skolemization [12]. This is essentially a transformation of the given proof such that we remove the strong quantifier instances (quantifier rules requiring an eigenvariable) and replace them with a skolem term. This transformation is only done to the strong quantifiers present in the end sequent. We cannot skolemize the strong quantifiers of the cut formula being that this would place a skolem term on one side of the cut and none on the other side. Skolemization is required so that the projections can be soundly added to the proof in order to construct the ACNF. As a result of this skolemization, in both A.1 and in the following schematic clause set one will find a new function which is not directly related to the theorem, namely, the function g. This new function is the skolem term. Interestingly enough, it will be used and act exactly like a successor function. The following formulae represent the clause set terms of the proof schema found in the ECS-schema .

Clause set terms for ψ

$$Step: \; C_{\psi}(n+1,\alpha) \equiv ((((\vdash f(\alpha) < n+1) \otimes \vdash n+1 = f(\alpha)) \otimes \vdash) \otimes \vdash) \oplus C_{\varphi}(n+1,\alpha)$$

$$Base: \; C_{\psi}(0,\alpha) \equiv ((((\vdash f(\alpha) < n+1) \otimes \vdash n+1 = f(\alpha)) \otimes \vdash) \otimes \vdash) \oplus C_{\varphi}(0,\alpha)$$

Clause set terms for φ

$$Step: \; C_{\varphi}(n+1,\alpha) \equiv (((((n+1 = f(\beta) \vdash n+1 = f(\beta)) \oplus (\alpha \leq \beta \vdash \alpha \leq \beta)) \oplus$$

$$(f(\beta) < n+1 \vdash f(\beta) < n+1) \oplus (\alpha \leq \beta, f(\alpha) < n+1 \vdash f(\beta) < n, n = f(\beta))) \oplus$$

$$(((n+1 = f(\alpha), n+1 = f(g(\alpha)) \vdash) \oplus \vdash \alpha \leq \alpha) \oplus \vdash \alpha \leq g(\alpha))) \oplus C_{\varphi}(n+1,\alpha)$$

$$Base: \; C_{\varphi}(0,\alpha) \equiv ((((0 = f(\alpha), 0 = f(g(\alpha)) \vdash) \oplus \vdash \alpha \leq g(\alpha)) \oplus \vdash \alpha \leq \alpha) \oplus$$

$$f(g(\alpha)) < 0 \vdash) \oplus f(\alpha) < 0 \vdash$$

The Clause set $C^{ECS}(n+1)$

Looking directly at the clause set terms it is quite hard to get an idea of what the clause set looks like. Sadly, unlike in the first-order case tautology elimination and subsumption elimination (removal of clauses which are equivalent to $A \vdash A$ and removal of clauses which are subsumed by a more general clause) are currently not automated and it is not known if one can even automate the procedures. We performed both tautology elimination and subsumption elimination by hand and wrote the clause set, instead of as a set of clause set terms, as a traditional clause set with a variable number of clauses based on the free parameter of the proof. The final clause set we derived is as follows.

$$\begin{array}{cc}
(C^{ECS}1) & \vdash \alpha \leq \alpha \\
(C^{ECS}2) & \vdash \alpha \leq g(\alpha) \\
(C^{ECS}3_0) & 0 = f(\alpha), 0 = f(g(\alpha)) \vdash \\
\vdots & \vdots \\
(C^{ECS}3_{n+1}) & n+1 = f(\alpha), n+1 = f(g(\alpha)) \vdash \\
(C^{ECS}4_1) & \alpha \leq \beta, f(\alpha) < 1 \vdash f(\beta) < 0, 0 = f(\beta) \\
\vdots & \vdots \\
(C^{ECS}4_n) & \alpha \leq \beta, f(\alpha) < n+1 \vdash f(\beta) < n, n = f(\beta) \\
(C^{ECS}5) & f(\alpha) < 0 \vdash \\
(C^{ECS}6) & \vdash f(\alpha) < n+1, f(\alpha) = n+1
\end{array}$$

This is the clause set we will use for the rest of this chapter.

5.3 Excerpt of a Refutation of $C^{ECS}(5)$ found by SPASS

A really nice property of the first-order CERES method [14] was that rather than being a stand alone cut-elimination procedure, it used existing advancements in automated theorem proving to perform cut-elimination. In a way, a step up from the Gentzen procedure. However, this property no longer holds in the case of schematic CERES, or more precisely, one cannot rely completely on theorem provers to provide the resolution refutation for a schematic clause set. At best, we can use theorem provers to refutate instances of the clause set rather then the entire clause set (for all instantiations of the parameter). In this section we use the SPASS theorem prover (precisely because it is my favourite theorem prover, feel free to use another prover in conjunction with schematic CERES) to refute an instance of the characteristic clause set of the ECS-schema; specifically, instance 5, clause set $C^{ECS}(5)$:

$(C^{ECS}1)$	$\vdash \alpha \leq \alpha$
$(C^{ECS}2)$	$\vdash \alpha \leq g(\alpha)$
$(C^{ECS}3_0)$	$0 = f(\alpha), 0 = f(g(\alpha)) \vdash$
$(C^{ECS}3_1)$	$1 = f(\alpha), 1 = f(g(\alpha)) \vdash$
$(C^{ECS}3_2)$	$2 = f(\alpha), 2 = f(g(\alpha)) \vdash$
$(C^{ECS}3_3)$	$3 = f(\alpha), 3 = f(g(\alpha)) \vdash$
$(C^{ECS}3_4)$	$4 = f(\alpha), 4 = f(g(\alpha)) \vdash$
$(C^{ECS}4_1)$	$\alpha \leq \beta, f(\alpha) < 1 \vdash f(\beta) < 0, 0 = f(\beta)$
$(C^{ECS}4_2)$	$\alpha \leq \beta, f(\alpha) < 2 \vdash f(\beta) < 1, 1 = f(\beta)$
$(C^{ECS}4_3)$	$\alpha \leq \beta, f(\alpha) < 3 \vdash f(\beta) < 2, 2 = f(\beta)$
$(C^{ECS}4_4)$	$\alpha \leq \beta, f(\alpha) < 4 \vdash f(\beta) < 3, 3 = f(\beta)$
$(C^{ECS}5)$	$f(\alpha) < 0 \vdash$
$(C^{ECS}6)$	$\vdash f(\alpha) < 4, f(\alpha) = 4$

The following resolution derivation is an example of the method used by SPASS to derive the clause $\vdash f(\alpha) < 2$ using $C^{ECS}(5)$. We took the SPASS output and converted it from a DAG to tree form and labelled each step with the number provided to that step by SPASS. According to the reasoning used by SPASS, this derivable clause is essential to the refutation of the clause set. Essentially, the refutation derives the clauses $\vdash f(\alpha) < i$ for $i \in [0, \cdots, n]$. When $i = 0$ we obtain $\vdash f(\alpha) < 0$ which contradicts the clause $(C^{ECS}5)$. The original SPASS output is provided after the resolution derivation in tree form. There are a few extra derivations in the spass output which we did not convert to the tree format being that they are simple to read off from the output and do not provide information as to the method SPASS uses to refute the clause set. The left out clauses just states that once one derives $\vdash f(\alpha) < i + 1$ it is possible to derive the fact that $\vdash f(\alpha) \leq i$.

Excerpt of Refutation

Following this except of the refutation will be the SPASS output. The tree format is provided up to line 316 of the SPASS output being that everything above the line 316 is not as interesting to the analogies and arguments following this subsection.

$$\dfrac{\dfrac{\dfrac{\begin{array}{c} g(\alpha)\le\beta\vdash_{316}\\ f(\alpha)<2,\\ 1=f(\beta),\\ f(\beta)<1\end{array} \qquad \vdash_{3}\alpha\le g(\alpha)}{\begin{array}{c}\vdash_{318} f(\alpha)<2,\\ 1=f(g(g(\alpha))),\\ f(g(g(\alpha)))<1\end{array}} \qquad \begin{array}{c}1=f(\alpha),\\ 1=f(g(\alpha))\vdash_{7}\end{array}}{\begin{array}{c}1=f(g(\alpha))\vdash_{321}\\ f(\alpha)<2,\\ f(g(g(\alpha)))<1\end{array}} \qquad \begin{array}{c}\alpha\le\beta,\\ f(\alpha)<1\\ \vdash_{14} 0=f(\beta)\end{array}}{\begin{array}{c}1=f(g(\alpha)),\\ g(g(\alpha))\le\beta\vdash_{322}\\ f(\alpha)<2,\\ 0=f(\beta)\end{array}}$$

$$\dfrac{\begin{array}{c}1=f(g(\alpha)),\\ g(g(\alpha))\le\beta\vdash_{322}\\ f(\alpha)<2,\\ 0=f(\beta)\end{array} \qquad \dfrac{\begin{array}{c} g(\alpha)\le\beta\vdash_{316}\\ f(\alpha)<2,\\ 1=f(\beta),\\ f(\beta)<1\end{array} \qquad \vdash_{2}\alpha\le\alpha}{\begin{array}{c}\vdash_{317} f(\alpha)<2,\\ 1=f(g(\alpha)),\\ f(g(\alpha))<1\end{array}}}{\begin{array}{c}\vdash_{325} f(g(\alpha))<1,\\ f(\alpha)<2,\\ f(\alpha)<2,\\ 0=f(g(g(g(\alpha))))\end{array}}$$

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdash_{325} f(g(\alpha))<1,\\ f(\alpha)<2,\\ f(\alpha)<2,\\ 0=f(g(g(g(\alpha))))\end{array}}{\begin{array}{c}\vdash_{327} f(g(\alpha))<1,\\ f(\alpha)<2,\\ 0=f(g(g(g(\alpha))))\end{array}} \qquad \vdash_{3}\alpha\le g(\alpha)}{\begin{array}{c}\vdash_{329} f(g(\alpha))<1,\\ f(\alpha)<2,\\ 0=f(g(g(g(\alpha))))\end{array}} \qquad \begin{array}{c}0,=f(\alpha),\\ 0,=f(g(\alpha))\vdash_{8}\end{array}}{\begin{array}{c}0=f(g(g(\alpha)))\vdash_{335}\\ f(\alpha)<2,\\ f(g(\alpha))<1\end{array}}$$

$$\dfrac{\begin{array}{c}0 = f(g(g(\alpha))) \vdash_{335} \\ f(\alpha) < 2, \\ f(g(\alpha)) < 1\end{array} \qquad \dfrac{\vdash_{2} \alpha \leq \alpha \qquad \dfrac{\begin{array}{c}\vdash_{325} f(g(\alpha)) < 1, \\ f(\alpha) < 2, \\ f(\alpha) < 2, \\ 0 = f(g(g(g(\alpha))))\end{array}}{\begin{array}{c}g(g(\alpha)) \leq \beta \vdash_{327} \\ f(g(\alpha)) < 1, \\ f(\alpha) < 2, \\ 0 = f(\beta)\end{array}}}{\begin{array}{c}\vdash_{328} 0 = f(g(g(\alpha))), \\ f(g(\alpha)) < 1, \\ f(\alpha) < 2\end{array}}}{\begin{array}{c}\vdash_{336} \\ f(\alpha) < 2, \\ f(g(\alpha)) < 1\end{array}}$$

$$\dfrac{\begin{array}{c}\vdash_{336} \\ f(\alpha) < 2, \\ f(g(\alpha)) < 1\end{array} \qquad \begin{array}{c}\alpha \leq \beta, \\ f(\alpha) < 1 \\ \vdash_{14} 0 = f(\beta)\end{array}}{\begin{array}{c}\alpha \leq \beta \vdash_{337} \\ f(\alpha) < 2, \\ 0 = f(\beta)\end{array}}$$

$$\dfrac{\dfrac{\begin{array}{c}\alpha \leq \beta \vdash_{337} \\ f(\alpha) < 2, \\ 0 = f(\beta)\end{array} \qquad \vdash_{3} \alpha \leq g(\alpha)}{\begin{array}{c}\vdash_{339} f(\alpha) < 2, \\ 0 = f(g(g(\alpha)))\end{array}} \qquad \begin{array}{c}0 = f(g(\beta)), \\ 0 = f(\beta) \vdash_{8}\end{array}}{\begin{array}{c}0 = f(g(\beta))) \vdash_{344} \\ f(\beta) < 2\end{array}}$$

$$\dfrac{\begin{array}{c}0 = f(g(\beta))) \vdash_{344} \\ f(\beta) < 2\end{array} \qquad \dfrac{\begin{array}{c}\alpha \leq \beta \vdash_{337} \\ f(\alpha) < 2, \\ 0 = f(\beta)\end{array} \qquad \vdash_{2} \alpha \leq \alpha}{\begin{array}{c}\vdash_{338} f(\alpha) < 2, \\ 0 = f(g(\alpha))\end{array}}}{\vdash_{345} f(\alpha) < 2}$$

Initial Clauses Used by SPASS

1[0:Inp]	$\vdash 4 = f(\alpha) \quad f(\alpha) < 4$
2[0:Inp]	$\vdash \alpha \leq \alpha$
3[0:Inp]	$\vdash \alpha \leq g(\alpha)$
4[0:Inp]	$4 = f(\alpha) \quad 4 = f(g(\alpha)) \vdash$
5[0:Inp]	$3 = f(\alpha) \quad 3 = f(g(\alpha)) \vdash$
6[0:Inp]	$2 = f(\alpha) \quad 2 = f(g(\alpha)) \vdash$
7[0:Inp]	$1 = f(\alpha) \quad 1 = f(g(\alpha)) \vdash$
8[0:Inp]	$0 = f(\alpha) \quad 0 = f(g(\alpha)) \vdash$
9[0:Inp]	$\alpha \leq \beta \quad f(\alpha) < 4 \vdash 3 = f(\beta) \quad f(\beta) < 3$
10[0:Inp]	$\alpha \leq \beta \quad f(\alpha) < 3 \vdash 2 = f(\beta) \quad f(\beta) < 2$
11[0:Inp]	$\alpha \leq \beta \quad f(\alpha) < 2 \vdash 1 = f(\beta) \quad f(\beta) < 1$
12[0:Inp]	$\alpha \leq \beta \quad f(\alpha) < 1 \vdash f(\beta) < 0 \;\; 0 = f(\beta)$
13[0:Inp]	$f(\alpha) < 0 \vdash$
14[0:MRR:12.2,13.0]	$\alpha \leq \beta \quad f(\alpha) < 1 \vdash 0 = f(\beta)$

Excerpt as found in SPASS Output

310[0:MRR:309.0,306.1]	$\vdash f(\alpha) < 3$
311[0:MRR:10.1,310.0]	$\alpha \leq \beta \vdash 2 = f(\beta) \quad f(\beta) < 2$
312[0:Res:2.0,311.0]	$\vdash 2 = f(\alpha) \quad f(\alpha) < 2$
314[0:Res:312.0,6.1]	$2 = f(\alpha) \vdash f(g(\beta)) < 2$
315[0:Res:314.1,11.1]	$2 = f(\alpha) \quad g(\alpha) \leq \beta \vdash 1 = f(\beta) \quad f(\beta) < 1$
316[0:Res:312.0,315.0]	$g(\alpha) \leq \beta \vdash f(\alpha) < 2 \;\; 1 = f(\beta) \quad f(\beta) < 1$
317[0:Res:2.0,316.0]	$\vdash f(\alpha) < 2 \;\; 1 = f(g(\alpha)) \quad f(g(\alpha)) < 1$
318[0:Res:3.0,316.0]	$\vdash f(\alpha) < 2 \;\; 1 = f(g(g(\alpha))) \quad f(g(g(\alpha))) < 1$
321[0:Res:318.1,7.1]	$1 = f(g(\alpha)) \vdash f(\alpha) < 2 \;\; f(g(g(\alpha))) < 1$
322[0:Res:321.2,14.1]	$1 = f(g(\alpha)) \quad g(g(\alpha)) \leq \beta \vdash f(\alpha) < 2 \;\; 0 = f(\beta)$
325[0:Res:317.1,322.0]	$g(g(\alpha)) \leq \beta \vdash f(\alpha) < 2 \;\; f(g(\alpha)) < 1 \;\; f(\alpha) < 2 \;\; 0 = f(\beta)$
327[0:Obv:325.1]	$g(g(\alpha)) \leq \beta \vdash f(g(\alpha)) < 1 \;\; f(\alpha) < 2 \;\; 0 = f(\beta)$
328[0:Res:2.0,327.0]	$\vdash f(g(\alpha)) < 1 \;\; f(\alpha) < 2 \;\; 0 = f(g(g(\alpha)))$
329[0:Res:3.0,327.0]	$\vdash f(g(\alpha)) < 1 \;\; f(\alpha) < 2 \;\; 0 = f(g(g(g(\alpha))))$
335[0:Res:329.2,8.1]	$0 = f(g(g(\alpha))) \vdash f(g(\alpha)) < 1 \;\; f(\alpha) < 2$
336[0:MRR:335.0,328.2]	$\vdash f(g(\alpha)) < 1 \;\; f(\alpha) < 2$
337[0:Res:336.0,14.1]	$g(\alpha) \leq \beta \vdash f(\alpha) < 2 \;\; 0 = f(\beta)$
338[0:Res:2.0,337.0]	$\vdash f(\alpha) < 2 \;\; 0 = f(g(\alpha))$
339[0:Res:3.0,337.0]	$\vdash f(\alpha) < 2 \;\; 0 = f(g(g(\alpha)))$
344[0:Res:339.1,8.1]	$0 = f(g(\alpha)) \vdash f(\alpha) < 2$
345[0:MRR:344.0,338.1]	$\vdash f(\alpha) < 2$

The steps 327 and 337 are the attempts to create functions which are not constant, but are monotonically decreasing. Every time the theorem prover attempt gets to a point where the function maps to zero we end up with a contradiction because there cannot be a number lower than zero. Thus, we go back a step and try another function. The step 327 is the first attempt where we try a function which is 2 at some point α, 1 at some point $g(\alpha)$ and at another point afterwards is 0. Thus it fails to find a function which is not constant for some value. The next attempt is a function which is 2 at some point α and 0 for every position after $g(\alpha)$. This is of course a failure as well, thus we have shown that none of the positions can be 2 and must be less than 2. This process is iterated until α is mapped to zero in which case there are no other possible functions to try and we have reached the contradiction.

5.4 Resolution Refutation Schemata

Even though the SPASS output is simple to read and seems to provide a simple resolution refutation, it is still too much for the $CERES_s$ method to express. The problem is, the SPASS proof output allows for the members of the schematic sort to be in any order from the root of the tree to the leaves. The $CERES_s$ method uses a resolution refutation calculus which was designed to handle paths which have a specific predefined ordering of the terms in the schematic sort. This problem will show its ugly head again in Ch. 6 when we discuss the NiA-schema, however in that case there is no way around the ordering problem, or at least we have not found a way around the ordering problem. Nonetheless in this case there is a solution, we can use the output of SPASS to help with the construction of a schematic refutation using our chosen ordering on the schematic sort terms. We do this by considering a case distinction on the occurrence of a given member of the schematic sort starting with the highest member in the ordering which we will refer to as α. If α occurs in the codomain of f we make a branch in the refutation and on this branch consider only functions f such that the codomain of f contains α. If α does not occur in the codomain of f we make a branch for all the functions f in which α does not occur in the codomain. This works because we forced the function f to be monotonically decreasing. If we remove that property, then this case distinction cannot be used. We rely on the fact that if α is in the codomain of f, then the value mapping to α must be smaller than the values mapping to every other member of the ordering and/or codomain.

In this section, we provide a schematic resolution refutation which can be expressed in the language of the $CERES_s$ method Sec. 4.6. The SPASS output was used as a guide for generating this resolution refutation. We will first write the resolution refutation parts in a recursive tree form and than write them in the syntax of Sec. 4.6. We write the refutation in this form because the syntax of Sec. 4.6 is extremely difficult to read and obfuscates the structure of the refutation.

ψ stepcase

$$\dfrac{\begin{array}{c} n+1=f(x),\\ n+1=f(g(x))\vdash\end{array} \qquad \dfrac{\dfrac{\varphi(n,g(x),X\circ f(g(x))<n+1\vdash)}{f(g(x))<n+1\vdash}\qquad \begin{array}{c}\vdash n+1=f(x),\\ f(x)<n+1\end{array}}{\vdash n+1=f(g(x))}}{\begin{array}{c} n+1=f(x)\vdash\Uparrow\\ \mu(n+1,x,X)\\ (1)\end{array}}$$

$$\dfrac{\dfrac{\begin{array}{c}(1)\\ n+1=f(x)\vdash\end{array}\qquad \begin{array}{c}\vdash f(x)<n+1,\\ n+1=f(x)\end{array}}{\vdash f(x)<n+1}\qquad \dfrac{\varphi(n,x,X\circ f(x)<n+1\vdash)}{f(x)<n+1\vdash}}{\begin{array}{c}\vdash\\ \Uparrow\\ \psi(n+1,x,X)\end{array}}$$

$$\psi(n+1,x,X)\Rightarrow res(res(\mu(n+1,x,X);\vdash f(x)<n+1,n+1=f(x);n+1=f(x));$$
$$\varphi(n,x,X\circ f(x)<n+1\vdash);f(x)<n+1)$$
$$\mu(n+1,x,X)\Rightarrow res(res(\vdash f(x)<n+1,n+1=f(x);\varphi(n,g(x),X\circ f(g(x))<n+1\vdash)$$
$$;f(x)<n+1);n+1=f(x),n+1=f(g(x))\vdash;n+1=f(x))$$

ψ basecase

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}0=f(g(x)),\\ 0=f(x)\vdash\end{array}\qquad \dfrac{f(x)<0\vdash\qquad \begin{array}{c}\vdash 0=f(x),\\ f(x)<0\end{array}}{\vdash 0=f(g(x))}}{0=f(x)\vdash}\qquad \begin{array}{c}\vdash f(x)<0,\\ 0=f(x)\end{array}}{\vdash f(x)<0}\qquad f(x)<0\vdash}{\begin{array}{c}\vdash\\ \Uparrow\\ \psi(0,x,X)\end{array}}$$

$$\psi(0,x,X)\Rightarrow res(res(\mu(0,x,X);\vdash f(x)<0,0=f(x);0=f(x));$$
$$f(x)<0\vdash;f(x)<0)$$

$$\mu(0,x,X)\Rightarrow res(res(\vdash f(x)<0,0=f(x);f(x)<0\vdash;f(x)<0);$$
$$0=f(x),0=f(g(x))\vdash;0=f(x))$$

φ **stepcase**

$$\dfrac{\dfrac{\vdash t(x) \leq g(t(x)) \qquad \begin{array}{c} X \circ x \leq y \vdash \\ n+1 = f(y), \\ f(y) < n+1 \end{array}}{\begin{array}{c} X \circ \vdash \\ n+1 = f(g(t(x))), \\ f(g(t(x))) < n+1 \end{array}} \qquad \dfrac{\varphi(n, g(t(x)), X \circ f(g(t(x))) < n+1 \vdash)}{f(g(t(x))) < n+1 \vdash}}{\begin{array}{c} X \circ \vdash \\ n+1 = f(g(t(x))) \\ (3) \end{array}}$$

$$\dfrac{\begin{array}{c} (3) \\ X \circ \vdash n+1 = f(g(t(x))) \end{array} \qquad n+1 = f(g(t(x))), n+1 = f(t(x)) \vdash}{\begin{array}{c} X \circ n+1 = f(t(x)) \vdash \\ (2) \end{array}}$$

$$\dfrac{\begin{array}{c} (2) \\ X \circ n+1 = f(t(x)) \vdash \end{array} \qquad \dfrac{\vdash t(x) \leq t(x) \qquad \begin{array}{c} X \circ x \leq y \vdash \\ n+1 = f(y), \\ f(y) < n+1 \end{array}}{\begin{array}{c} X \circ \vdash \\ n+1 = f(t(x)), \\ f(t(x)) < n+1 \end{array}}}{\begin{array}{c} X \circ \vdash f(t(x)) < n+1 \\ (1) \end{array}}$$

$$\dfrac{\begin{array}{c} (1) \\ X \circ \vdash f(t(x)) < n+1 \end{array} \qquad \dfrac{\varphi(n, t(x), X \circ f(t(x)) < n+1 \vdash)}{f(t(x)) < n+1 \vdash}}{\begin{array}{c} X \circ \vdash \\ \Uparrow \\ \varphi(n+1, t(x), X) \end{array}}$$

$$\varphi(n+1, t(x), X) \Rightarrow res(res(\mu(n+1, t(x), X);$$

$$res(\vdash \alpha \leq \alpha; X \circ \alpha \leq \beta \vdash f(\beta) < n+1, n+1 = f(\beta); \alpha \leq \beta); n+1 = f(t(x)));$$

$$\varphi(n, t(x), X \circ f(t(x)) < n+1 \vdash); f(t(x)) < n+1)$$

φ **basecase**

$$\dfrac{\begin{matrix}0 = f(g(t(x))),\\ 0 = f(t(x)) \vdash\end{matrix} \qquad \dfrac{f(g(t(x))) < 0 \vdash \qquad \dfrac{\vdash t(x) \leq g(t(x)) \qquad \begin{matrix}X \circ x \leq y \vdash 0 = f(y),\\ f(y) < 0\end{matrix}}{\begin{matrix}X\circ \vdash 0 = f(g(t(x))),\\ f(g(t(x))) < 0\end{matrix}}}{X\circ \vdash 0 = f(g(t(x)))}}{\begin{matrix}X \circ 0 = f(t(x)) \vdash\\ (1)\end{matrix}}$$

$$\begin{matrix}\dfrac{\dfrac{\begin{matrix}(1)\\ X \circ 0 = f(t(x)) \vdash\end{matrix} \qquad \dfrac{\vdash t(x) \leq t(x) \qquad \begin{matrix}X \circ x \leq y \vdash f(y) < 0,\\ 0 = f(y)\end{matrix}}{\begin{matrix}X\circ \vdash f(t(x)) < 0,\\ 0 = f(t(x))\end{matrix}}}{X\circ \vdash f(t(x)) < 0} \qquad f(t(x)) < 0 \vdash}{X\circ \vdash}\\ \Uparrow\\ \varphi(0, t(x), X)\end{matrix}$$

$$\varphi(0, t(x), X) \Rightarrow res(res(\mu(0, t(x), X);$$

$res(\vdash \alpha \leq \alpha; X\circ\alpha \leq \beta \vdash f(\beta) < 0, 0 = f(\beta); \alpha \leq \beta); 0 = f(t(x))); f(t(x)) < 0 \vdash; f(t(x)) < 0)$

Thus, the resolution refutation, when written in call order will be as follows:
$\psi(n+1, x, X) \Rightarrow res(res(\mu(n+1, x, X); \vdash f(x) < n+1, n+1 = f(x); n+1 = f(x));$
$\varphi(n, x, X \circ f(x) < n+1 \vdash); f(x) < n+1)$

$\psi(0, x, X) \Rightarrow res(res(\mu(0, x, X); \vdash f(x) < 0, 0 = f(x); 0 = f(x));$
$f(x) < 0 \vdash; f(x) < 0)$

$\varphi(n+1, t(x), X) \Rightarrow res(res(\mu(n+1, t(x), X);$
$res(\vdash \alpha \leq \alpha; X \circ \alpha \leq \beta \vdash f(\beta) < n+1, n+1 = f(\beta); \alpha \leq \beta); n+1 = f(t(x)));$
$\varphi(n, t(x), X \circ f(t(x)) < n+1 \vdash); f(t(x)) < n+1)$

$\varphi(0, t(x), X) \Rightarrow res(res(\mu(0, t(x), X);$
$res(\vdash \alpha \leq \alpha; X \circ \alpha \leq \beta \vdash f(\beta) < 0, 0 = f(\beta); \alpha \leq \beta); 0 = f(t(x))); f(t(x)) < 0 \vdash$
$; f(t(x)) < 0)$

$\mu(n+1, x, X) \Rightarrow res(res(\vdash f(x) < n+1, n+1 = f(x); \varphi(n, g(x), X \circ f(g(x)) < n+1 \vdash)$
$; f(x) < n+1); n+1 = f(x), n+1 = f(g(x)) \vdash; n+1 = f(x))$

$\mu(0, x, X) \Rightarrow res(res(\vdash f(x) < 0, 0 = f(x); f(x) < 0 \vdash; f(x) < n + 1);$
$0 = f(x), 0 = f(g(x)) \vdash; 0 = f(x))$

A fully unrolled resolution refutation of the clause set $C^{ECS}(3)$ can be found in A.2. It is easy to write a second-order unifier for this schematic resolution refutation as required by the resolution calculus found in Sec. 4.6, namely:

$$\lambda x.(h(n - x, \alpha))$$

where $h(n + 1, \alpha)$ is defined as follows:

$$h(n + 1, \alpha) \longrightarrow g(h(n, \alpha))$$

$$h(0, \alpha) \longrightarrow \alpha$$

5.5 Projections

Using the first-order CERES method of projection extraction (Sec. 3.4 for more detail see [13] we end up with the following projection derived from the formal proof found in A.1. It is enough for use to use the first-order CERES method's projection extraction because the projections do not need induction. If the projections included induction as well, then we would need to use the projection extraction defined in [30]. Though, as mentioned earlier, it is very hard to use the projection extraction found in [30], and of which it is one of the principal reasons why the concept was dropped from [31]. We do not show the precise machinations needed to derive these projections. Instead we just provide the projection in a standard sequent calculus format.

projection to $(C^{ECS}3_i)$

$$\dfrac{\dfrac{\dfrac{\begin{array}{c} i = f(\alpha)^*,\\ i = f(g(\alpha))^* \vdash\\ f(\alpha) = f(g(\alpha)) \end{array}}{\begin{array}{c} i = f(\alpha)^*,\\ i = f(g(\alpha))^*,\\ \alpha \leq g(\alpha) \vdash\\ f(\alpha) = f(g(\alpha)) \end{array}}\ w : r}{\begin{array}{c} i = f(\alpha)^*,\\ i = f(g(\alpha))^* \vdash\\ \alpha \leq g(\alpha) \rightarrow f(\alpha) = f(g(\alpha)) \end{array}}\ \rightarrow : r}{\begin{array}{c} i = f(\alpha)^*,\\ i = f(g(\alpha))^* \vdash\\ \exists x(x \leq g(x) \rightarrow f(x) = f(g(x))) \end{array}}\ \exists : r$$

projection to $(C^{ECS}0)$

The sequent $\bigvee_{i=0}^{n} i = f(\alpha) \vdash f(\alpha) < n + 1^*$ can be unfolded to a schematic sequence of axioms of the form $i = f(\alpha) \vdash f(\alpha) < n + 1$ for $0 \leq i \leq n$. For space reasons we leave this in schematic form.

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\begin{array}{c}\bigvee_{i=0}^{n} i = f(\alpha) \vdash \\ f(\alpha) < n+1^*\end{array}
}{
\begin{array}{c}\bigvee_{i=0}^{n} i = f(\alpha), \\ f(\alpha) = f(\alpha) \vdash \\ f(\alpha) < n+1^*\end{array}
}\, w:l
\qquad f(\alpha) < f(\alpha) \vdash
}{
\begin{array}{c}\bigvee_{i=0}^{n} i = f(\alpha), \\ f(\alpha) \leq f(\alpha) \vdash \\ f(\alpha) < n+1^*\end{array}
}\, \vee:l
\qquad
\begin{array}{c} n+1 = f(\alpha) \vdash \\ n+1 = f(\alpha)^*\end{array}
}{
\begin{array}{c}\bigvee_{i=0}^{n+1} i = f(\alpha), \\ f(\alpha) \leq f(\alpha) \vdash \\ n+1 = f(\alpha)^*, f(\alpha) < n+1^*\end{array}
}\, \vee:l
\qquad \vdash \alpha \leq \alpha
}{
\begin{array}{c}\bigvee_{i=0}^{n+1} i = f(\alpha), \\ \alpha \leq \alpha \to f(\alpha) \leq f(\alpha) \vdash \\ n+1 = f(\alpha)^*, f(\alpha) < n+1^* \\ (1)\end{array}
}\, \to: l
}{
\begin{array}{c}\forall x \bigvee_{i=0}^{n+1} i = f(x), \\ \alpha \leq \alpha \to f(\alpha) \leq f(\alpha) \vdash \\ n+1 = f(\alpha)^*, f(\alpha) < n+1^* \\ (1)\end{array}
}\, \forall:l
}{
\begin{array}{c}\forall x \bigvee_{i=0}^{n+1} i = f(x), \\ \forall y(\alpha \leq y \to f(y) \leq f(\alpha)) \vdash \\ n+1 = f(\alpha)^*, f(\alpha) < n+1^* \\ (1)\end{array}
}\, \forall:l
}{
\begin{array}{c}\forall x \bigvee_{i=0}^{n+1} i = f(x), \\ \forall x \forall y(x \leq y \to f(y) \leq f(x)) \vdash \\ n+1 = f(\alpha)^*, f(\alpha) < n+1^*\end{array}
}\, \forall:l
$$

5.6 Herbrand Sequent for the schematic ACNF of the proof found in A.1

Luckily in the case of our formal proof of ECS, we where able to extract a schematic Herbrand sequent . This was possible because the term language is quite simple, i.e. only monadic functions are used. We just follow the unifications up the resolution refutation to the projections and collect the necessary terms. We will do something similar for a more complex proof, however, in that case, the construction of the Herbrand sequent was extremely difficult. If we make the assumption that $g^0 = 0$, we can write the Herbrand sequent for the cut free proof as follows:

$$\bigwedge_{i=0}^{n+1} \bigvee_{j=0}^{n} j = f(g^i(0)), \bigwedge_{i=0}^{n} g^i(0) \leq g^{i+1}(0) \rightarrow f(g^{i+1}(0)) \leq f(g^i(0)) \vdash$$

$$\bigvee_{i=0}^{n} g^i(0) \leq g^{i+1}(0) \rightarrow f(g^i(0)) = f(g^{i+1}(0))$$

This schematic Herbrand sequent was not algorithmically extracted but rather by analysis of the complete ANCF for instances of the schematic proof. If we simplify the above sequent by introducing natural numbers rather than sequences of g functions we can provide a natural interpretation:

$$\bigwedge_{i=0}^{n+1} \bigvee_{j=0}^{n} j = f(i), \bigwedge_{i=0}^{n} i \leq (i+1) \rightarrow f(i+1) \leq f(i) \vdash$$

$$\bigvee_{i=0}^{n} i \leq (i+1) \rightarrow f(i) = f(i+1)$$

Which essentially states a weaker version of the pigeonhole principle , namely, if we are given $n+1$ pigeons and have n holes such that pigeons can only be assigned the current hole or some hole lower in the ordering, then at some point two pigeons will be assigned to the same hole. We call this version of the pigeonhole principle, the *monotonic* pigeonhole principle .

CHAPTER 6

Beyond Ceres$_{S}$: the NiA-Schema

In this chapter we provide an analysis of a proof formalized in the schematic sequent calculus $\mathbf{LKS}$ introduced in Ch. 4. The original goal was to analyse this proof using the schematic CERES method. However, an integral part of the method is not expressive enough to handle the cut-free version of our formal proof, namely, the schematic resolution calculus . Though, we do not have a proof that the clause set extracted from the formal proof does not have a schematic refutation, of which the schematic resolution refutation calculus of Ch. 4 could express, results providing us with a lower bound for the size of the minimal schematic refutation elude to a minimal schematic refutation outside the expressive power of the calculus. Also analogously, work done on CTL^{+} expressivity [2] has shown that the smallest formula in CTL^{+} expressing a property very similar to that of our schematic clause set has size $O(n!)$. A resolution derivation expressing an object $O(n!)$ in size (the size of the smallest refutation we have found) has a shape which cannot be expressed in the calculus of Ch. 4. We are referring to the breath of the tree rather than the length of the branches. The tree is too wide to be expressed in the calculus, the width is dependent on the free parameter. Using the resolution calculus of Ch. 4 it is possible to represent resolution refutations with non-elementary length, however, the trees do not have schematic growth of the width.

Lacking a schematic resolution refutation which fits in the expressive power of the calculus required us to prove that a refutation we construct for the characteristic clause set is indeed a refutation for every instance of the free parameter, i.e. mathematically showing that the refutation is a refutation for every value of the free parameter. Using this proof, it is possible to create a cut-free proof of each instance, but not algorithmically as was possible in the formalization of Ch. 4. Also, such a proof informs us as to what a more general language, powerful enough to express the refutation of the clause set of our formal proof, needs to contain.

We analyse a formal proof of the statement that total functions with domain $\mathbb{N}$ with a finite range cannot be one to one. One can write this assertion as follows:

Statement 2 (Non-injectivity Assertion (NiA))**.** *Let $f : \mathbb{N} \rightarrow [0, \cdots, n]$, where $n \in \mathbb{N}$, be total, then there exists $i, j \in \mathbb{N}$ such that $i < j$ and $f(i) = f(j)$.*

NiA is actually, a generalization of the pigeonhole principle (PHP), which has been heavily studied in literature [17, 22–24, 44, 50, 51]. However, the more important part of the defined schema are the lemmata used in the formalization. We will provide a formal proof using a particularly nice Π_2 cut. In particular we will use the following schema of lemmata to prove the end sequent of the NiA schema:

Lemma 6.0.1 (Infinity Lemma)**.** *Given a total function $f : \mathbb{N} \rightarrow \mathbb{N}_{n+1}$ then either for all $x \in \mathbb{N}$ there exist a $y \in \mathbb{N}$ such that $x \leq y$ and $f(y) = i$ where $i \in \mathbb{N}_n$, or for all $x \in \mathbb{N}$ there exist a $y \in \mathbb{N}$ such that $x \leq y$ and $f(y) = n + 1$.*

We would like to note that the goal of this chapter is not to formalize the NiA-schema, nor is it necessary to focused on performing cut-elimination on the NiA-schema, but rather the main goal initially was to test what the schematic CERES method of Ch. 4 could handle and cannot handle. We show that the NiA-schema can be formalized in the $\mathbf{LKS}$ calculus, but cannot be cut-eliminated back to an $\mathbf{LKS}$ proof using the method of Ch. 4. Though there is still the open point that one does not know if our formalization of the NiA-schema only has refutations of the characteristic clause set outside the expressive power of the resolution refutation calculus, it is already interesting to see that a relatively simple refutation of the characteristic clause set is outside the calculus. This point remains open, though it seems rather uninteresting to solve, in that another formalized proof will likely be out of the expressive power of the language, i.e. the gNiA-schema.

The rest of this work will be as follows:

- In Sec. 6.1 we provide an informal "mathematical" proof of NiA-schema's end sequent using the desired lemmata in $\mathbf{LKS}$.
- In B.1 we give a formal proof of NiA-schema's end sequent using the desired lemmata in $\mathbf{LKS}$.
- In Sec. 6.3 we extract the schematic clause set and projections (using the CERES method from Ch. 3) from the formal proof outlined in B.1.
- In Sec. 6.4 we prove some lower bounds for the minimum refutation in terms of the number of schematic clauses used. At the end of the section we give a very rough outline on how to show that a factorial number of clauses is necessary.
- In Sec. 6.5 we extract the clause set and give a mathematical refutation of the clause set, and an unrolled example of the resolution refutation.
- In B.2 we give an example of the resolution refutation for the clause set instantiated with two.

6.1 Informal Mathematical Proof of the NiA Schema

This section provides an Informal Mathematical Proof of the $\mathbf{LKS}$ proof found in Appendix B.1.This proof is not required for the rest of the material in this chapter, and stands more as a reference, that is a reference used for understanding the **LKS** proof found in Appendix B.1. We attempt to follow the arguments of the proof found in Appendix B.1 as closely as possible.

Lemma 6.1.1. *Given a total function $f : \mathbb{N} \to \{0, \cdots m-1\}$, where $1 \leq m$, then for all $x \in \mathbb{N}$ there exists a $y \in \mathbb{N}$ such that $x \leq y$ and $f(n) = z$ for $z \in \{0, \cdots m-1\}$.*

<u>Proof.</u>
If we assume the contrary, that is there exists an $x \in \mathbb{N}$ for all $y \in \mathbb{N}$ such that $x \leq y$, then $f(n) \neq z$ for $z \in \{0, \cdots m-1\}$ we reach a contradiction with the definition of f.
$\square$

Lemma 6.1.2. *Let f be a function as defined in Lem. 6.1.1, then for $1 \leq k \leq m$ either for all $x_1 \in \mathbb{N}$ there exists a $y_1 \in \mathbb{N}$ such that $x_1 \leq y_1$ and $f(n) = z$ for $z \in \{0, \cdots k-1\}$, or for all $x_2 \in \mathbb{N}$ there exists a $y_2 \in \mathbb{N}$ such that $x_2 \leq y_2$ and $f(n) = z$ for $z \in \{k, \cdots m-1\}$.*

<u>Proof.</u>
Assume the contrary, that is there exists a $x_1 \in \mathbb{N}$ for all $y_1 \in \mathbb{N}$ such that when $x_1 \leq y_1$, then $f(n) \neq z$ for $z \in \{0, \cdots k-1\}$, and there exists a $x_2 \in \mathbb{N}$ for all $y_2 \in \mathbb{N}$ such that when $x_2 \leq y_2$ then $f(n) \neq z$ for $z \in \{k, \cdots m-1\}$. The former part of the above statement implies that for all y_1, such that $x_1 \leq y_1$, $f(n) = z$, where $z \in \{k, \cdots m-1\}$. The latter part of the above statement implies that for all y_2, such that $x_2 \leq y_2$, $f(n) = z$, where $z \in \{0, \cdots k-1\}$. However, this implies that the following must hold, for $\gamma = \max(x_1, x_2)$, it is the case that $f(\gamma) = z$ for $z \in \{0, \cdots k-1\}$ and $z \in \{k, \cdots m-1\}$, which violates the fact that f is a function. Thus, we have a contradiction.
$\square$

Lemma 6.1.3. *Given a total function $f : \mathbb{N} \to \mathbb{N}$, such that for all $x \in \mathbb{N}$ there exists a $y \in \mathbb{N}$ where $x \leq y$ and $f(y) = z$, where $z \in \mathbb{N}$, then there exists $p, q \in \mathbb{N}$ such that $p < q$ and $f(p) = f(q)$.*

<u>Proof.</u>
Let us choose two numbers α_0 and α_1 such that $\alpha_0 < \alpha_1$. If we choose two more numbers n_0 and n_1 such that $\alpha_0 \leq n_0$ and $\alpha_1 \leq n_1$ then it is the case that $f(n_0) = f(n_1) = x$.
$\square$

Lemma 6.1.4. *Let f be a function as defined in Lem. 6.1.1, then either there exists $p, q \in \mathbb{N}$ such that $p < q$ and $f(p) = f(q)$, or for all $x \in \mathbb{N}$ there exists a $y \in \mathbb{N}$ such that $x \leq y$ and $f(n) = z$ for $z \in \{0, \cdots, m-2\})$.*

<u>Proof.</u>
Instantiating the free variable k of Lem. 6.1.2 with $m-1$ we get the following statement: either for all $x_1 \in \mathbb{N}$ there exists a $y_1 \in \mathbb{N}$ such that $x_1 \leq y_1$ and $f(n) = z$ for $z \in \{0, \cdots m-2\}$, or for all $x_2 \in \mathbb{N}$ there exists a $y_2 \in \mathbb{N}$ such that $x_2 \leq y_2$ and $f(n) = m-1$. Focusing on the latter part of this statement and using Lem. 6.1.3 we can derive the statement required by the lemma.
$\square$

From now on we will refer to the statement of Lem. 6.1.4, namely, either there exists $p, q \in \mathbb{N}$ such that $p < q$ and $f(p) = f(q)$, or for all $x \in \mathbb{N}$ there exists a $y \in \mathbb{N}$ such that $x \leq y$ and $f(n) = z$ for $z \in \{0, \cdots, m-2\}$), as $rr(m)$. And, in general we use $rr(k)$ when $1 \leq k \leq m$.

Lemma 6.1.5. *Let f be a function as defined in Lem. 6.1.1, and $1 < k < m$, then if $rr(k)$ holds, then so does $rr(k-1)$.*

Proof.
By Lem. 6.1.4 we know that $rr(m)$ holds. we can consider this a basecase for an induction. For the induction hypothesis, let us assume that the theorem holds for all w such that $k < w + 1 \leq m$ and we now show that it hold for w. Taking the right hand side of the statement $rr(w+1)$ and applying Lem. 6.1.2 for $k = w$, we can apply the same procedure that was used in Lem. 6.1.4 resulting in $rr(w)$ and thus proving the theorem.
$\square$

Theorem 6.1.1. *Let f be a function as defined in Lem. 6.1.1, if $rr(m)$ holds then so does $rr(1)$.*

Proof.
This is a result of simply chaining together the implications of Lem. 6.1.5.
$\square$

Corollary 6.1.1. *Let f be a function as defined in Lem. 6.1.1, then $\exists p, q \in \mathbb{N}(p < q \wedge f(p) = f(q))$*

Proof.
Take the result of Thm. 6.1.1 and apply Lem. 6.1.3 to the right hand side of $rr(1)$ once.
$\square$

6.2 A Short Introduction to Formalization

Moving from the informal proof in Sec. 6.1 to the formal proof used in this section required several intermediate proofs to be construct. Actually, the proof found in Sec. 6.1 is one of the intermediate steps. A mathematician attempting to prove the end sequent of the NiA-schema would never prove it using the method of Sec. 6.1. One can already see in Lem. 6.1.4 of the informal proof the schematic sequence of cuts. Maybe, not as explicitly as the formal proof illustrates the sequence, but more explicitly then a mathematical proof would.

Though, in terms of formalizing a schematic proof in the **LKS** calculus, the hardest part is going from the formal proof for an instance to the formal proof for the inductive step. We mentioned in the introduction to this dissertation that the NiA-schema is based on the tape proof found in [8, 9, 55]. This tape proof is essentially the NiA-schema for two symbols rather than n symbols. Because the number of symbols used in the tape proof is finite and fixed, there is a trick used in this proof which cannot be used in the NiA-schema because it would lead to an eigenvariable violation . Namely, in the tape proof there is a sequence of formulae which have an eigenvariable from a proof symbol lower in the ordering which must remain unquantified. In the finite and fixed version there are no proof symbols and the proof is finite, thus, it is ok to leave eigenvariables unquantified and quantify them all at the end of the proof. This problem requires the proof links to have the eigenvariables explicitly stated in the proof links. If one was to remove these eigenvariables, then the leaves of the proof would not be of the form $A \vdash A$. One way which this problem could be resolved was by adding bounded parameters to the **LKS** calculus, but this seemed to be overkill. In the end it was solved by changing the cut formulae, i.e. making them stronger.

This change of cut formulae might explain the situation found in this chapter, namely that the NiA-schema can be formalized in the **LKS** calculus but its resolution refutation, so far, cannot be formalized within the schematic resolution calculus of ch. 4. The schematic resolution calculus was designed using the same concepts as **LKS** calculus.

6.3 Clause set Extraction & Projection Construction

Clause set Extraction

The work outlined in this section heavily relies on the **LKS** proof of Appendix B.1. For those who would like to look into the details of the formal proof please see Appendix B.1. As mentioned at the beginning of the Sec. 6.1. Extracting the clause sets from the base cases of $\omega(0)$ and $\varphi(0)$, the following two sets result:

$$CS(\omega, 0, \emptyset) \equiv ((((s(\beta) \leq \alpha \vdash) \otimes (f(\alpha) \sim 0, f(\beta) \sim 0 \vdash)) \oplus \vdash \alpha \leq \alpha) \oplus \vdash f(\alpha) \sim 0)$$

$$CS(\varphi, 0, \emptyset) \equiv (s(\beta) \leq \alpha \vdash) \otimes (f(\alpha) \sim 0, f(\beta) \sim 0 \vdash)$$

By $CS(\omega, 0, \emptyset)$ we are referring to the clause set for proof ω given the value 0 from the schematic sort and configuration $\emptyset$. Extracting the clause set from the step-case of φ will require us to use a single configuration , namely, $\Omega_{n+1} = \left\{ \forall x \exists y (x \leq y \wedge \bigvee_{i=0}^{n+1} f(y) \sim i)^{**} \vdash \right\}$. This is the cut formula passed through the proof links .

$$CS(\omega, n+1, \emptyset) \equiv (((\vdash \alpha \leq \alpha) \oplus (\vdash \bigvee_{i=0}^{n+1} f(\alpha) \sim i)) \oplus CS(\varphi, n+1, \Omega_{n+1}))$$

$$CS(\varphi, n+1, \Omega_{n+1}) \equiv (((((s(\beta) \leq \alpha \vdash) \otimes (f(\alpha) \sim n+1, f(\beta) \sim n+1 \vdash)) \oplus$$
$$(max(\alpha, \beta) \leq \gamma \vdash \alpha \leq \gamma)) \oplus (max(\alpha, \beta) \leq \gamma \vdash \beta \leq \gamma)) \oplus CS(\varphi, n, \Omega_n))$$

When we simplify this representation and remove redundancy, we get the following clause set $C(n)$:

$$\begin{array}{ll}
(C1) & \vdash \alpha \leq \alpha \\
(C2) & max(\alpha, \beta) \leq \gamma \vdash \alpha \leq \gamma \\
(C3) & max(\alpha, \beta) \leq \gamma \vdash \beta \leq \gamma \\
(C4_0) & f(\beta) \sim 0, f(\alpha) \sim 0, s(\beta) \leq \alpha \vdash \\
\vdots & \vdots \\
(C4_n) & f(\beta) \sim n, f(\alpha) \sim n, s(\beta) \leq \alpha \vdash \\
(C5) & \vdash f(\alpha) \sim 0, \cdots, f(\alpha) \sim n
\end{array}$$

The clause set $C(n)$ is the set which we will prove unsatisfiable for all instantiations of the free parameter n. Being that the language for resolution refutations presented in the original paper on schematic CERES Ch. 4 does not have enough expressive power, we will present the refutation using a mathematical meta-language.

Projections for the Formal proof of NiA

An interesting result pertaining to our formal proof of NiA is the simplicity of the projections . There are $n + 1$ projections in the clause set $C(n)$, one for each of the $C4$ clause and one for the clause $C5$. The rest of the projections are trivial being that they are only axioms.

Projection for (C4$_i$)

$$\dfrac{\dfrac{\dfrac{s(\alpha) \leq \beta \vdash \alpha < \beta \qquad \begin{array}{c} f(\alpha) \sim i, f(\beta) \sim i \vdash \\ f(\alpha) \sim f(\beta) \end{array}}{\begin{array}{c} f(\alpha) \sim i, f(\beta) \sim i, s(\alpha) \leq \beta \vdash \\ \alpha < \beta \wedge f(\alpha) \sim f(\beta) \end{array}}\wedge : r}{\begin{array}{c} f(\alpha) \sim i, f(\beta) \sim i, s(\alpha) \leq \beta \vdash \\ \exists x(x < \beta \wedge f(x) \sim f(\beta) \end{array}}\exists : r}{\begin{array}{c} f(\alpha) \sim i, f(\beta) \sim i, s(\alpha) \leq \beta \vdash \\ \exists x \exists y(x < y \wedge f(x) \sim f(y)) \end{array}}\exists : r$$

Projection for (C5)

$$\dfrac{\bigvee_{i=0}^{n} f(\alpha) \sim i \vdash \bigvee_{i=0}^{n} f(\alpha) \sim i}{\forall x \bigvee_{i=0}^{n} f(x) \sim i \vdash \bigvee_{i=0}^{n} f(\alpha) \sim i}\forall : l$$

6.4 Lower Bounds for Size of the schematic refutation of $C(n)$

In this section, we show that a refutation of the clause set $C(n)$ cannot be polynomial in size where the metric is the number of schematic length clauses used. We conjecture that there lower bound can be pushed up to $O(f(n)^n)$ (where $f(n)$ is a polynomial), but we where not able to show that this is the case. We provide proof that the number of schematic length clauses used is more than $O(2^n)$, and show that based on this results we can extrapolate the necessary structure for a refutation of this clause set. If all the proof steps in this extrapolation were worked out the result would be that at least $O(n!)$ schematic length clauses are needed. At the end of the section we outline the proof steps needed for proving the extrapolated results.

$O(2^n)$ lower bound for the Clause Set $C(n)$

Definition 6.4.1. *The set $der(C(n))$ is the set of all clauses derivable from the clauses in $C(n)$.*

Definition 6.4.2. *Given that* $\mathbf{c}' \in der(C(n))$*, let* $res(\mathbf{c}', C(n))$ *be the set of all resolution derivations deriving* $\mathbf{c}'$ *from* $C(n)$*.*

Definition 6.4.3. *Let* $Occ(x, r)$ *be defined as the number of times the clause* x *is used in the refutation* r*.*

Definition 6.4.4. *Given a resolution derivation r, a sub-resolution derivation* $r' \subseteq r$ *is a sub-tree whose leaves are instances of the clauses in the clause set.*

Lemma 6.4.1. *Let* $\mathbf{c}' \in der(C(n))$ *for* $n \in \mathbb{N}$ *be such that*

$$res(\mathbf{c}', \{C1, C2, C3, C4_i, C4_j)\}) \not\equiv \emptyset$$

, such that $i \neq j$*, and*

$$res(\mathbf{c}', \{C1, C2, C3\}) \equiv \emptyset$$

, then either

$$res(\mathbf{c}', \{C1, C2, C3, C4_i)\}) \not\equiv \emptyset$$

or

$$res(\mathbf{c}', \{C1, C2, C3, C4_j)\}) \not\equiv \emptyset$$

, but not both.

Proof. First, it is obvious that for all clauses $\mathbf{c}$, $res(\mathbf{c}, \{C4_i, C4_j\}) \equiv \emptyset$ unless $\mathbf{c} \in \{C4_i, C4_j\}$. Second, let c be such that $res(\mathbf{c}, \{C1, C2, C3, C4_j\}) \not\equiv \emptyset$, then it can be shown (though derivation is difficult and not worth the space) that given any clause $\mathbf{c}'$, $res(\mathbf{c}', \{\mathbf{c}, C4_i\}) \not\equiv \emptyset$ only when $\mathbf{c}' \in \{\mathbf{c}, C4_i\}$ or $res(\mathbf{c}, \{C1, C2, C3\}) \not\equiv \emptyset$. Thus, when there exist a $r \in res(\mathbf{c}', \{C1, C2, C3, C4_i, C4_j\})$, r cannot contain both $C4_i, C4_j$.
□ □

Lemma 6.4.2. *Given the clause set* $C(n)$*,* $res(\perp, \{C1, C2, C3, C4_0, \ldots, C4_n)\}) \equiv \emptyset$

Proof. No positive occurrences of $f(\alpha) \sim i$, where $0 \leq i \leq n$, occurs in these clauses, and $res(\perp, \{C1, C2, C3\}) \equiv \emptyset$ is obvious, because the set contains no completely negative clauses.
□ □

Lemma 6.4.3. $res(\vdash s(\beta) \leq \beta, C(n))) \equiv \emptyset$*.*

Proof. This statement is the same as $res(\vdash s(\beta) \leq \beta, \{C1, C2, C3\}) \equiv \emptyset$ because these are the only clauses with positive occurrences of the $\leq$ predicate. Also, for either of the clauses $C2, C3$ to derive a positive clause their antecedent must be unified such that $max(\alpha, \beta) \equiv \gamma$ and then it can be resolved with $C1$. However, this implies a max function will occur placed in the consequent. This Essentially this is due to $\Delta, s(\beta) \leq \beta \vdash$ and $max(a, b) \leq d \vdash b \leq d$ not being unifiable.
□ □

Lemma 6.4.4. *For all* $\mathbf{c}' \in der(C(n))$ *either* $res(\mathbf{c}', \{C1, C2, C3, C4_i\}) \not\equiv \emptyset$ *for some* i *or the the minimal set of clauses* $C^{min}(n) \subseteq C(n)$ *such that* $res(\mathbf{c}', C^{min}(n)) \not\equiv \emptyset$ *contains* $C5$*.*

Proof. By Lem. 6.4.1 we know that if $res(\mathbf{c}', \{C1, C2, C3, C4_0, \ldots, C4_n\}) \not\equiv \emptyset$, then for some i,
$res(\mathbf{c}', \{C1, C2, C3, C4_i\}) \not\equiv \emptyset$. Thus, we define

$$D_{C5} = der(C(n)) - \left\{\mathbf{c}' \mid res(\mathbf{c}', \{C1, C2, C3, C4_0, \ldots, c4_n\}) \not\equiv \emptyset\right\}$$

using Lem. 6.4.2 and the fact that it is known that $\bot \in der(C(i))$ for $i \in \mathbb{N}$ (this is known by the fact that $C(n)$ for each n is a characteristic clause set. Characteristic clause sets are by definition refutable), we get that D_{C5} is not empty, it must at least contain $\bot$. Also, being that we have used every clause within $C(n)$ the case distinction is strict.
□ □

Theorem 6.4.1. *For every $r \in res(\bot, C(n))$, such that the clauses $C4_i$ occur in branches of r in monotonically increasing order, $2^{n+1} \leq occ(C5_n, r)$*

Proof. Lem 6.4.3 essentially states that when the clauses $C4_0$-$C4_n$ are used to derive bottom they are never instantiated such that $f(\beta) \not\sim i$ and $f(\alpha) \not\sim i$ factor, where $0 \leq i \leq n$. Using this fact and Lem 6.4.4, we can set up the following induction proof:

<u>Basecase</u>

Let us assume we have $r \in res(\bot, C(0))$ and we would like to show that $2 \leq occ(C5, r)$. We know by Lem. 6.4.3 that every clause such that

$$\mathbf{c} \in \left\{\mathbf{c}' \mid res(\mathbf{c}', \{C1, C2, C3, C4_0\}) \wedge \exists r(r \in res(\bot, C(0)) \wedge occ(c', r) \neq 0\right\}$$

will have two instances of $f(\alpha) \not\sim 0$. Thus, the minimal refutation of $C(0)$ would be

$$\dfrac{\dfrac{\begin{array}{c} r \in res(\mathbf{c}, \{C1, C2, C3, C4_0\}) \\ \vdots \\ \mathbf{c} \end{array} \qquad \vdash f(\alpha) \sim 0}{\mathbf{c}'} \qquad \vdash f(\alpha) \sim 0}{\vdash}$$

implying that $2 \leq occ(C5, r)$ holds.

<u>Stepcase</u>

Let use assume that for all $i \leq n$ and all $r \in res(\bot, C(i))$ that $2^{i+1} \leq occ(C5, r)$, then we would like to show that the same holds for $n + 1$. Let us consider the case when we take an arbitrary resolution refutation $r \in res(\bot, C(n))$, such that the clauses $C4_i$ occur in branches of r in monotonically increasing order, and replace every instance of $C5$ in r with $C5^{(n+1)}$, the clause $C5$ from the clause set $C(n + 1)$. We could assume without loss of generality that r is minimal in terms of $occ(C5, r)$. We will call the resulting resolution derivation r'. Assuming that every instance of the atom $f(\alpha) \sim (n + 1)$ at the leaves will factor with each other at some point in the derivation, then r' will derive a clause consisting of only a single $f(\alpha) \sim (n+1)$. As in the basecase, we choose a clause

$$\mathbf{c} \in \left\{ \mathbf{c}' \mid res(\mathbf{c}', \{C1, C2, C3, C4_{n+1}\}) \wedge \exists r (r \in res(\bot, C(n+1)) \wedge occ(c', r) \neq 0 \right\}$$

and get the following refutation r'':

$$\dfrac{\dfrac{\begin{array}{c}\vdots \\ \mathbf{c}\end{array} \qquad r'}{\mathbf{c}'} \qquad\qquad r'}{\vdash}$$

we know by the induction hypothesis that $2^{n+1} \leq occ(C5, r)$, where r is the resolution refutation r' was derived from, thus, for r' it is also the case that $2^{n+1} \leq occ(C5^{(n+1)}, r')$. We know that r'' consist of two occurrences of the resolution derivation r' and thus, $occ(C5^{(n+1)}, r'') = 2 \cdot occ(C5^{(n+1)}, r')$. Thus we can finally conclude that $2^{n+2} \leq occ(C5^{(n+1)}, r'')$. If we use less occurrences then 2^{n+2} we will end up not being able to refute $C(n)$.

□ □

Definition 6.4.5. *Let $res_m(\bot, C(n))$ for $0 \leq n < m$, be the set of resolution derivations such that we take a refutation $r \in res(\bot, C(n))$ and replace every instance of $C5 \in C(n)$ with an instance of $C5 \in C(m)$.*

Lemma 6.4.5. *Given a resolution derivation $r \in res_{n+1}(\bot, C(n))$, let r' be a sub-resolution derivation such that it contains only two occurrences of $C5^{n+1}$. Then the root of r' must contain two positive occurrences of the literal $f(\cdot) \sim n+1$, for two distinct terms t and t′ which cannot be factored.*

Proof. Consider the following resolution derivation r_p deriving the clause C'_n,we can safely assume $r_p \subseteq r$:

$$\underline{r_p}$$

$$\dfrac{\begin{array}{c} r_c \\ \vdots \\ f(\alpha) \sim i, f(t(\alpha, \overline{x_m})) \sim i \vdash \end{array} \qquad C5^{n+1}}{\begin{array}{c} f(t'[\alpha]) \sim i \vdash f(\overline{t'}[\alpha]) \sim 0, \cdots, f(\overline{t'}[\alpha]) \sim i-1, \\ f(\overline{t'}[\alpha]) \sim i+1, \cdots, f(\overline{t'}[\alpha]) \sim n+1 \end{array}}\,\text{res}$$

$$C'_n$$

Where $r_c \in res(c, \{C1, C2, C3, C4_i\})$ and $c = f(\alpha) \sim i, f(t(\alpha, \overline{x_m})) \sim i \vdash$, $t(\alpha, \overline{x_m})$ is some arbitrary term. We assume that
$res(c, \{C1, C2, C3, C4_i\}) \not\equiv \emptyset$. Also, t' is a term with a single hole such $t' \in \{\bullet, t(\bullet, \overline{x_m})\}$ and $\overline{t'}$ is defined as, if $t' = \bullet$ then $\overline{t'} = t(\bullet, \overline{x_m})$ and vice versa. If we extend the derivation r_p to the following derivation r':

$$\underline{r'}$$

$$\dfrac{\dfrac{\begin{array}{c} r_c \\ \vdots \\ f(\alpha) \sim i, f(t(\alpha, \overline{x_m})) \sim i \vdash \end{array} \qquad C0_{n+1}}{\begin{array}{c} f(t'[\alpha]) \sim i \vdash f(\overline{t'}[\alpha]) \sim 0, \cdots, f(\overline{t'}[\alpha]) \sim i-1, \\ f(\overline{t'}[\alpha]) \sim i+1, \cdots, f(\overline{t'}[\alpha]) \sim n+1 \end{array}} \qquad C0_{n+1}}{\begin{array}{c} \vdash f(\overline{t'}[\alpha]) \sim 0, \cdots, f(\overline{t'}[\alpha]) \sim i-1, \\ f(\overline{t'}[\alpha]) \sim i+1, \cdots, f(\overline{t'}[\alpha]) \sim n+1 \\ f(t'[\alpha]) \sim 0, \cdots, f(t'[\alpha]) \sim i-1, \\ f(t'[\alpha]) \sim i+1, \cdots, f(t'[\alpha]) \sim n+1 \\ C''_n \end{array}}$$

we can see that the C''_n must contain two instances of $f(\cdot) \sim n+1$ because $f(t'[\alpha]) \sim n+1$ and $f(\overline{t'}[\alpha]) \sim n+1$ are not factorable. During the induction step we will only apply rules to one of the instances of $C5^{n+1}$ without loss of generality because we know that the resolution refutation we have chosen is for the instance n of the clause set, thus none of the resolution steps can be applied to literals with an $n+1$ schematic term. Thus, showing that two instances of a literal with an $n+1$ schematic term don't factor without applying any rules to the second instance of the $C5^{n+1}$ clause will be enough. We can show this using induction, but all we need from the second instance of $C5^{n+1}$ is the fact that $f(\cdot) \sim n+1$ will not factor when introduced through resolution.

So far we have proven the basecase for an induction over the number of clauses which have C'_n as their predecessor before resolving with the second instance of $C5^{n+1}$, we will abbreviate this by $\#p(C'_n)$. Let us assume that the lemma holds for $\#p(C'_n) \leq k$,$k \in \mathbb{N}$, let us show for $\#p(C'_n) = k+1$. We will assume W.L.O.G that C'_n was resolved with $C5^{n+1}$ using the i^{th} schematic term, thus, from now on when we use i we are referring to this specific position. we know that every one of the successors of C'_n and is derived from resolving C'_n or one of its successors with a clause derived from $\{C1, C2, C3, C4_j\}$ for some $j \in [0, \cdots, n]$. Also, $wlog$ we will assume that the first k successors of C'_n where derived from a resolution refutation found in the following sets:

$$res(C'_n(1), \{C1, C2, C3, C4_0, C'_n\})$$

$$\vdots$$

$$res(C'_n(i-1), \{C1, C2, C3, C4_{i-1}, C'_n(i-2)\})$$
$$res(C'_n(i+1), \{C1, C2, C3, C4_{i+1}, C'_n(i-1)\})$$

$$\vdots$$

$$res(C'_n(k), \{C1, C2, C3, C4_k, C'_n(k-1)\})$$

where $C'_n(w)$ for $1 \leq w \leq k$ means the w^{th} successor of C'_n. We can do this being that in this particular case the resolution procedure is invariant of which term in the ω sort is chosen in which order, we just need to make sure they match when performing a resolution step.

Depending on the assignment to $\overline{t'}$ in C'_n, either all the atoms in the consequent of $C'_n(k)$ have the free variable α or at some point α is replaced with some β and all the atoms are dependent on this β. Nonetheless, there is a free variable which all the atoms in $C'_n(k)$ depend on. Thus, adding one more successor of C'_n, $C'_n(k+1)$, which is derived from a resolution refutation in the set $res(C'_n(k+1), \{C1, C2, C3, C4_k, C'_n(k)\})$, does not change the fact that every literal in the consequent of $C'_n(k+1)$ will have a variable which they share. Thus resolving $C'_n(k+1)$ with $C5^{n+1}$, assuming the new free variable is β, will result in the following two atoms $f(T'[\beta]) \sim n+1$ and $f(\overline{T'}[\beta]) \sim n+1$ where T' is defined as t', but for a term which is the result of the extra resolution steps with all the successors of C'_n. Thus, by induction the lemma holds.

□ □

Theorem 6.4.2. *For every $r \in res(\bot, C(n))$, such that the clauses $C4_i$ occur in branches of r in monotonically increasing order, $2^{n+1} < occ(C5, r)$*

Proof. Applying Lem. 6.4.5 to the assumption in the proof of Thm. 6.4.1 that all the occurrences of $f(\cdot) \sim n+1$ factor.

□ □

At this point we have shown that the clause set needs an exponential number of schematic length clauses to refutate it, however, this is not as strong of a lower bound as we would like. Lem. 6.4.5 eludes to a problem with resolving two clauses derived using $C5$ such that both have a non-empty consequent, essentially, the values in the consequent which have the same value in the schematic sort will not factor because they all share at least one free variable and have different terms.

This leads to to intuitive fact that a clause which is derived from two instances of $C5$ and has an empty consequent is optimal because anything in the consequent is essentially doubled and will have to be dealt with further down the tree. Also, something that was not pointed out in the proof of Lem. 6.4.5 is that even though the literals of consequent share variables it need not be the case that literals of the antecedent share variables with other literals of the antecedent.

If we follow with this train of thought we see that given two instances of $C5$, one of the derivable clause with the least number of literals is as follows:

$$\bigwedge_{i=0}^{k-1} f(x_i) \sim i, \bigwedge_{i=k+1}^{n} f(x_i) \sim i \vdash \tag{6.1}$$

If we continue this pattern we see in Eq. 6.1 we see that one of the clauses with the least number of literals which can be constructed from five instances of $C5$ is as follows:

$$\bigwedge_{i=0}^{k-1} f(x_i) \sim i, \bigwedge_{i=k+1}^{w-1} f(x_i) \sim i, \bigwedge_{i=w+1}^{n} f(x_i) \sim i \vdash \tag{6.2}$$

These facts are provable, but we would have to go into great detail to prove them, which is not worth the time or effort in that Sec. 6.5 will use exactly these clauses to produce a resolution refutation of the clause set, which is exactly what we need. But, if we where to prove these facts, the resulting proof structure would be first creating the set of all the clause which can be derived from m instances of the clause $C5$. Then we would put an ordering on these clauses based on the number of literals the clauses have. Finally, we would have to show that there exists a bottom element in the ordering such that there exist a refutation using that element, of which uses less occurrences of $C5$ than any refutation which uses an element which is not a bottom element. This is quite obvious considering Lem. 6.4.5. The final step is to do an induction on the number of members in the antecedent of the bottom elements in the ordering to show that we can get to one element in the antecedent, which implies we can refute the clause set with one more instance of $C5$.

Such a proof would show that the number of occurrences of the clause $C5$ would have to be $O(n!)$. But, as mentioned before every step in this proof is extremely intuitive and very tedious, thus, the mathematical gain is quite limited for a proof which would take many pages.

Short explanation of bounding results

The idea behind this section was to find a lower bound for the size of the resolution refutation of the NiA-schema's characteristic clause set. In the end we where only able to show a slight lower bound, no where close to the bound we where hoping for. Even though such a bound does not really help one in understanding the following section, the tools and methods developed in this section can aid future work analysing the refutations of schematic clause sets.

6.5 Mathematical proof for the refutation of $C(n)$

In this section we give a proof that the clause set derived from B.1, $C(n)$, is refutable for all values of n. We prove this result by first deriving a set of clauses which we will consider the least elements of a well ordering. Then we show how resolution can be applied to this least elements to derive clauses of the form $f(\alpha) \sim i$ for $0 \leq i \leq n$. The last step is simply take the clause $(C5)$ from the clause set $C(n)$ and resolve it with each of the $f(\alpha) \sim i$ clauses. The heart of this set is the well ordering needed to prove that the given refutation is a refutation of the clause set.

Even though the results in this section only provide us with a proven minimum upper bound for the number of schematic length clauses (clause C5) needed to refute the clause set, it is the intuition built while finding a lower bounding for the number of instances of clause C5 needed to refute the clause set that lead to these results. In B.2 we provide a fully unrolled example of the refutation for the clause set $C(2)$.

Schematic Resolution refutation for $C(n)$

Definition 6.5.1. *Let the* variable symbol set $\mathcal{V}$ *denote a countably infinite set of variable symbols.*

Definition 6.5.2. *Let* $\Sigma = \{\max(\cdot,\cdot), s(\cdot), 0\}$*, our* term language over Σ *is defined by the following grammar:*

$$TERM := \max(TERM, TERM)|s(TERM)|0|x$$

where $x \in \mathcal{V}$. The set of all terms will be denoted by T_Σ.

Definition 6.5.3. *we define the primitive recursive term $m(k+1, x_1, \cdots, x_{k+1}, t)$ with arity $k+2$ for $k \in \mathbb{N}$, $x_1, \cdots, x_k \in \mathcal{V}$, and $t \in T_\Sigma$ be defined as follows:*

$$m(k+1, x_1, \cdots, x_{k+1}, t) \Rightarrow m(k, x_1, \cdots, x_k, max(s(x_{k+1}), t)) \tag{6.3a}$$

$$m(0, t) \Rightarrow t \tag{6.3b}$$

We will use the abbreviation $\bar{x}_k$ for the list of parameters $x_1, \cdots, x_k$.

The following definition is introduced to simplify the proofs in this chapter.

Definition 6.5.4. *We define the resolution rule $res(\sigma, P)$ where σ is a unifier and P is a predicate as follows:*

$$\frac{\Pi \vdash P^*, \Delta \qquad \Pi', P^{**} \vdash \Delta'}{\Pi\sigma, \Pi'\sigma \vdash \Delta\sigma, \Delta'\sigma}\, res(\sigma, P)$$

The predicates P^* and P^{**} are defined such that $P^{**}\sigma = P^*\sigma = P$. Also, there are no occurrences of P in $\Pi'\sigma$ and P in $\Delta\sigma$.

Lemma 6.5.1. *Given $0 \leq k$ and $0 \leq n$, the clause $\vdash t \leq m(k, \overline{x}_k, t)$, where $t \in T_\sigma$, is derivable by resolution from $C(n)$.*

Proof. Let us consider the case when $k = 0$, the clause we would like to show derivability of is $\vdash t \leq m(0, t)$, which is equivalent to the clause $\vdash t \leq t$, an instance of (C1). Assuming the lemma holds for all $m < k+1$, we show that the lemma holds for $k+1$. By the induction hypothesis, the instance $\vdash max(s(x_{k+1}), t) \leq m(k, \overline{x}_k, max(s(x_{k+1}), t))$ is derivable. Thus, the following derivation proves that the clause $\vdash t \leq m(k+1, \overline{x}_{k+1}, t)$, where $t = max(s(x_{k+1}), t')$ for some term t' is derivable:

$$\cfrac{\cfrac{\begin{array}{c}(IH)\\ \vdash P\end{array} \qquad \begin{array}{c}(C3)\\ max(\beta, \delta) \leq \gamma \vdash \delta \leq \gamma\end{array}}{\vdash t \leq m(k, \overline{x}_k, max(s(x_{k+1}), t))}\, res(\sigma, P)}{\vdash t \leq m(k+1, \overline{x}_{k+1}, t)}\, \twoheadrightarrow$$

$$P = max(s(x_{k+1}), t) \leq m(k, \overline{x}_k, max(s(x_{k+1}), t))$$

$$\sigma = \{\beta \leftarrow s(x_{k+1}), \gamma \leftarrow m(k, \overline{x}_k, max(s(x_{k+1}), t)), \delta \leftarrow t\}$$

□ □

Corollary 6.5.1. *Given $0 \leq k$ and $0 \leq n$, the clause $\vdash s(x_{k+1}) \leq m(k, \overline{x}_k, max(s(x_{k+1}), t))$ is derivable by resolution from $C(n)$.*

Proof.
$$\cfrac{\overset{(Lem.6.5.1)}{\vdash P} \qquad \overset{(C2)}{max(\beta,\delta) \leq \gamma \vdash \beta \leq \gamma}}{\vdash s(x_{k+1}) \leq m(k, \overline{x}_k, max(s(x_{k+1}), t))}\; res(\sigma, P)$$

$$P = max(s(x_{k+1}), t) \leq m(k, \overline{x}_k, max(s(x_{k+1}), t))$$

$$\sigma = \{\beta \leftarrow s(x_{k+1}), \gamma \leftarrow m(k, \overline{x}_k, max(s(x_{k+1}), t)), \delta \leftarrow t\}$$

□ □

Corollary 6.5.2. *Given* $0 \leq k$ *and* $0 \leq n$*, the clause* $f(x_{k+1}) \sim i, f(m(k, \overline{x}_k, max(s(x_{k+1}), t))) \sim i \vdash$ *for* $0 \leq i \leq n$ *is derivable by resolution from* $C(n)$*.*

Proof.
$$\cfrac{\overset{(Cor.6.5.1)}{\vdash P} \qquad \overset{(C4_i)}{f(\alpha) \sim i, f(\beta) \sim i, s(\alpha) \leq \beta \vdash}}{f(x_{k+1}) \sim i, f(m(k, \overline{x}_k, max(s(x_{k+1}), t))) \sim i \vdash}\; res(\sigma, P)$$

$$P = s(x_{k+1}) \leq m(k, \overline{x}_k, max(s(x_{k+1}), t))$$

$$\sigma = \{\alpha \leftarrow x_{k+1}, \beta \leftarrow m(k, \overline{x}_k, max(s(x_{k+1}), t))\}$$

□ □

Corollary 6.5.3. *Given* $0 \leq k$ *and* $0 \leq n$*, the clause* $f(x_{k+1}) \sim i, f(m(k, \overline{x}_k, s(x_{k+1}))) \sim i \vdash$ *for* $0 \leq i \leq n$ *is derivable by resolution from* $C(n)$*.*

Proof.
$$\cfrac{\overset{(Lem.6.5.1)}{\vdash P} \qquad \overset{(C4_i)}{f(\alpha) \sim i, f(\beta) \sim i, s(\alpha) \leq \beta \vdash}}{f(x_{k+1}) \sim i, f(m(k, \overline{x}_k, s(x_{k+1}))) \sim i \vdash}\; res(\sigma, P)$$

$$P = s(x_{k+1}) \leq m(k, \overline{x}_k, s(x_{k+1}))$$

$$\sigma = \{\alpha \leftarrow x_{k+1}, \beta \leftarrow m(k, \overline{x}_k, s(x_{k+1})))\}$$

□ □

So far the steps taken in the refutation of the clause set $C(n)$ have been straight forward and have not required much additional machinery to derive. However, in the next lemma we need to add bijective functions to the schematic sort to show the derivability of certain clauses. The problem we need to get around is that every permutation of the numbers from 0 to n (i.e. $0231456 \cdots n$) will be needed to refute the clause set. Thus, we cannot directly prove the properties of the refutation using a simple linear induction. We replace the numbers in the numeric sort with the bijective function in order to simplify the problem enough to prove the derivability of some clauses without using a complex ordering. However, even this simplification was not enough for all the clauses needed in the refutation, and thus, to derive $\vdash$ we still need to construct a special ordering. Also to know, we add an additional meta-level function symbol in order to simplify the proof. In the rest of this section the function b is a bijection which is not in the term language of the logic, but rather represents an arbitrary term from the schematic sort, i.e. a numeral.

Definition 6.5.5. *Given* $0 \leq n$*,* $-1 \leq k \leq j \leq n$*,* $z \in \mathcal{V}$ *and a bijective function* $b : \mathbb{N}_n \rightarrow \mathbb{N}_n$ *we define the following formulae:*

$$c_b(k, j, z) = \bigwedge_{i=0}^{k} f(x_{b(i)}) \sim b(i) \vdash \bigvee_{i=k+1}^{j} f(m(k+1, \bar{x}_{k+1}, z)) \sim b(i).$$

The formulae $c_b(-1, -1, z) \equiv \vdash$ *for all values of* n *and* $c_b(-1, n, z) \equiv \vdash \bigvee_{i=0}^{n} f(z) \sim i$ *for all values of* n *.*

Lemma 6.5.2. *Given* $0 \leq n$*,* $-1 \leq k \leq n$ *and for all bijective functions* $b : \mathbb{N}_n \rightarrow \mathbb{N}_n$*. the formula* $c_b(k, n, z)$ *is derivable by resolution from C(n).*

Proof. We prove this lemma by induction on k and a case distinction on n. When $n = 0$ there are two possible values for k, $k = 0$ or $k = -1$. When $k = -1$ the clause is an instance of (C5). When $k = 0$ we have the following derivation:

$$\dfrac{\begin{array}{c}(C5)\\ c_b(-1,1,y)\end{array} \qquad \begin{array}{c}(Cor.6.5.2[i \leftarrow b(0), k \leftarrow 0])\\ f(x_1) \sim b(0), f(max(s(x_1), z)) \sim b(0) \vdash\end{array}}{c_b(0,1,z)}\ res(\sigma, P)$$

$$P = f(max(s(x_1), z)) \sim b(0)$$

$$\sigma = \{y \leftarrow max(s(x_1), z)\}$$

By $(Cor.6.5.2[i \leftarrow b(0), k \leftarrow 0])$ we mean take the clause that is proven derivable by Cor. 6.5.2 and instantiate the free parameters of Cor. 6.5.2, i.e. i and k, with the given terms, i.e. $b(0)$ and 0. Remember that $b(0)$ can be either 0 or 1. We will use this syntax through out the dissertion. When $n > 0$ and $k = -1$ we again trivially have (C5). When $n > 0$ and $k = 0$, the following derivation suffices:

$$\dfrac{\begin{array}{c}(C5)\\ c_b(-1,n,y)\end{array} \qquad \begin{array}{c}(Cor.6.5.2[i \leftarrow b(0), k \leftarrow 0])\\ f(x_1) \sim b(0), f(max(s(x_1), z)) \sim b(0) \vdash\end{array}}{c_b(0,n,z)}\ res(\sigma, P)$$

$$P = f(max(s(x_1), z)) \sim b(0)$$

$$\sigma = \{y \leftarrow max(s(x_1), z)\}$$

The main difference between the case for $n = 1$ and $n > 1$ is the possible instantiations of the bijection at 0. In the case of $n > 1$, $b(0) = 0 \ \vee \cdots \vee \ b(0) = n$. Now we assume that for all $w < k + 1 < n$ and $n > 0$ the theorem holds, we proceed to show that the theorem holds for $k + 1$. The following derivation will suffice:

$$\dfrac{\begin{matrix}(IH) & (Cor.6.5.2[i \leftarrow b(k+1)]) \\ c_b(k,n,y) & f(x_{k+1}) \sim b(k+1), P \vdash\end{matrix}}{c_b(k+1,n,z)}\, res(\sigma, P)$$

$$P = f(m(k, \overline{x}_k, max(s(x_{k+1}), t))) \sim b(k+1)$$

$$\sigma = \{y \leftarrow max(s(x_{k+1}), z)\}$$

□ □

Definition 6.5.6. *Given* $0 \leq n$*,* $0 \leq k \leq j \leq n$*, and a bijective function* $b : \mathbb{N}_n \to \mathbb{N}_n$ *we define the following formulae:*

$$c_b'(k,j) = \bigwedge_{i=0}^{k} f(x_{i+1}) \sim b(i) \vdash \bigvee_{i=k+1}^{j} f(m(k, \bar{x}_k, s(x_{k+1})) \sim b(i).$$

Lemma 6.5.3. *Given* $0 \leq n$*,* $0 \leq k \leq n$ *and for all bijective functions* $b : \mathbb{N}_n \to \mathbb{N}_n$*. the formula* $c_b'(k,n)$ *is derivable by resolution from C(n).*

Proof. We prove this lemma by induction on k and a case distinction on n. When $n = 0$ it must be the case that $k = 0$. When $k = 0$ we have the following derivation :

$$\dfrac{\begin{matrix}(C5) & (Cor.6.5.3[i \leftarrow 0, k \leftarrow 0]) \\ c_b(-1,0,y) & f(x_1) \sim 0, f(s(x_1)) \sim 0 \vdash\end{matrix}}{c_b'(0,0)}\, res(\sigma, P)$$

$$P = f(s(x_1)) \sim 0$$

$$\sigma = \{y \leftarrow s(x_1)\}$$

Remember that $b(0)$ can only be mapped to 0. When $n > 0$ and $k = 0$, the following derivation suffices:

$$\dfrac{\begin{matrix}(C5) & (Cor.6.5.3[i \leftarrow b(0), k \leftarrow 0]) \\ c_b(-1,n,y) & f(x_1) \sim b(0), f(s(x_1)) \sim b(0) \vdash\end{matrix}}{c_b'(0,n)}\, res(\sigma, P)$$

$$P = f(s(x_1)) \sim b(0)$$

$$\sigma = \{y \leftarrow s(x_1)\}$$

The main difference between the case for $n = 0$ and $n > 0$ is the possible instantiations of the bijection at 0. In the case of $n > 0$, $b(0) = 0 \ \vee \cdots \vee \ b(0) = n$. Now we assume that for all $w \leq k$ the theorem holds, we proceed to show that the theorem holds for $k + 1$. The following derivation will suffice:

$$\dfrac{\begin{array}{cc}(IH) & (Cor.6.5.2[i \leftarrow b(k+1)]) \\ c_b(k,n,y) & f(x_{k+1}) \sim b(k+1), P \vdash \end{array}}{c_b(k+1,n,z)}\ res(\sigma, P)$$

$$P = f(m(k, \overline{x}_k, max(s(x_{k+1}), t))) \sim b(k+1)$$
$$\sigma = \{y \leftarrow max(s(x_{k+1}), z)\}$$

□ □

Definition 6.5.7. *Given* $0 \leq n$ *we define the ordering relation* $\lessdot_n$ *over* $A_n = \{(i,j) | i \leq j \wedge 0 \leq i,j \leq n \wedge i,j \in \mathbb{N}\}$ *s.t. for* $(i,j),(l,k) \in A_n$*,* $(i,j) \lessdot_n (l,k)$ *iff* $i,k,l \leq n$*,* $j < n$*,* $l \leq i$*,* $k \leq j$*, and* $i = l \leftrightarrow j \neq k$ *and* $j = k \leftrightarrow i \neq l$*.*

The ordering defined above, when drawn as a tree, essentially maps out the branching of the resolution refutation. It is a complex and strange ordering, and will only play the role of an intermediary allowing us to show that the clause set $C(n)$ is refutable for every n. After showing that the clause set is refutable, we will develop a new schematic language for iterating through resolution refutations which is expressive enough to handle the aforementioned refutation. This language is based on vectors of natural numbers rather than singleton numbers. Once this language has been developed we will show how one can convert the resolution refutation built in this section into a resolution refutation using the newly constructed language.

Lemma 6.5.4. *The ordering* $\lessdot_n$ *over* A_n *for* $0 \leq n$ *is a complete well ordering.*

Proof. Every chain has a greatest lower bound, namely, one of the members of A_n, (i,n) where $0 \leq i \leq n$, and it is transitive, anti-reflexive, and anti-symmetric. □

The clauses proved derivable by Lem. 6.5.3 can be paired with members of A_n as follows, $c'_b(k,n)$ is paired with (k,n). Thus, each $c'_b(k,n)$ is essentially the greatest lower bound of some chain in the ordering $\lessdot_n$ over A_n.

Lemma 6.5.5. *Given* $0 \leq k \leq j \leq n$*, for all bijective functions* $b : \mathbb{N}_n \rightarrow \mathbb{N}_n$ *the clause* $c'_b(k,j)$ *is derivable from C(n).*

Proof. We will prove this lemma by induction over A_n. The basecases are the clauses $c'_b(k,n)$ from Lem. 6.5.3. Now let us assume that the lemma holds for all clauses $c'_b(k,i)$ pairs such that, $0 \leq k \leq j < i \leq n$ and for all clauses $c'_b(w,j)$ such that $0 \leq k < w \leq j \leq n$, then we want to show that the lemma holds for the clause $c'_b(k,j)$. We have not made any restrictions on the bijections used, we will need two different bijections to prove the theorem. The following derivation provides proof:

$$\dfrac{\dfrac{\begin{array}{cc}(IH[k,j+1]) & (IH[k+1,k+1]) \\ \Pi_b(k), \vdash \Delta_b(k,j), P_b(j+1) & \Pi_{b'}(k), f(x_{b'(k+1)}) \sim b'(k+1) \vdash \end{array}}{\Pi_b(k), \Pi_{b'}(k) \vdash \Delta_b(k,j)}\ res(\sigma, P)}{\begin{array}{c}\Pi_b(k) \vdash \Delta_b(k,j) \\ c'_b(k,j)\end{array}}\ c:l$$

$$P_b(k+1) = f(m(k,\bar{x}_k, s(x_{k+1}))) \sim b(k+1)$$

$$\Pi_b(k) \equiv \bigwedge_{i=0}^{k} f(x_{b(i)}) \sim b(i)$$

$$\Delta_b(k,j) \equiv \bigvee_{i=k+1}^{j} f(m(k,\bar{x}_k, s(x_{k+1}))) \sim b(i)$$

$$\sigma = \left\{ x_{b'(k+1)} \leftarrow m(k,\bar{x}_k, s(x_{k+1})) \right\}$$

We assume that $b'(k+1) = b(j+1)$ and that $b'(x) = b(x)$ for $0 \leq x \leq k$. □

Theorem 6.5.1. *Given $n \geq 0$, $C(n)$ derives $\vdash$.*

Proof. By Lem. 6.5.5, The clauses $f(x) \sim 0 \vdash, \cdots, f(x) \sim n \vdash$ are derivable. Thus, we can prove the statement by induction on the instantiation of the clause set. When $n = 0$, the clause (C5) is $\vdash f(x) \sim 0$ which resolves with $f(x) \sim 0 \vdash$ to derive $\vdash$. Assuming that for all $n' \leq n$ the theorem holds we now show that it holds for $n+1$. The clause (C5) from the clause set $C(n+1)$ is the clause (C5) from the clause set $C(n)$ with the addition of a positive instance of $\vdash f(\alpha) \sim (n+1)$. Thus, by the induction hypothesis we can derive the clause $\vdash f(\alpha) \sim (n+1)$. By Lem. 6.5.5 we can derive $f(x) \sim (n+1) \vdash$, and thus, resolving the two derived clauses results in $\vdash$.

□

A simple way to understand this proof is by first categorizing the clauses proved derivable by Lem. 6.5.3 in sets based on the number of literals they have in their consequent. The pattern that emerges is that the number of clauses with k literals in the consequent is exactly $\binom{n}{k}$. And thus, the number of clauses proven derivable by Lem. 6.5.3 is

$$\sum_{k=1}^{n} \binom{n}{k} = 2^n.$$

We do not have to substract one for $\binom{n}{n}$ because it is essentially interpreted as (C5), though it is not included in the set of clauses derivable from Lem. 6.5.3. Each one of these clauses essentially states that, the symbols in the antecedent show up on the tape, then the symbols in the consequent are the symbols which can show up in the "next" position such that up to this point no repetition of a symbol occurs. By next position we mean the maximum of all currently chosen positions plus one. Thus, if all symbols occur in the antecedent then there can be no symbol in the "next" position without a repeated symbol. Thus, starting from the clause with no consequent we can show how to resolve each of the other clauses (when the consequent is none empty) such that they have an empty consequent as well, and thus refute the clause set. We end this section with a theorem pertaining to the number of schematic length clauses used to refute this clause set.

Theorem 6.5.2. *Let r be the resolution refutation of Thm. 6.5.1 for the clause set $C(n)$, then $Occ(C5, r)$ is the result of the following recurrence relation $a(n+1) = (n+1) * a(n) + 1$ and $a(0) = 1$.*

Proof. Let us consider the case for the clause set $C(0)$. This is the case when we have only one symbol in the function's range. If we compute the recurrence we get $a(1) = a(0) + 1 = 2$ Now let us assume it holds for all $m \leq n$ and show it hold for $n + 1$. In the proof of Lem. 6.5.5, when deriving $c_b'(0, 0)$ the literal $f(\alpha) \sim b(0)$ is in the antecedent for every clause higher in the resolution derivation and it is never used in a resolution step . If we remove this clause from the antecedent then we have a resolution refutation for the clause $C(n)$, only if we rename the schematic sort terms accordingly. To refute $C(n + 1)$ we need to derive $n + 1$ distinct $c_b'(0, 0)$ clauses and resolve them with a single instance of $(C5)$. Thus, we have the equation, $Occ(C5^{n+1}, r_{n+1}) = (n + 1) * Occ(C5^n, r_n) + 1$ where r_{n+1} is the resolution refutation of Thm. 6.5.1 for the clause set $C(n + 1)$ and r_n is the resolution refutation of Thm. 6.5.1 for the clause set $C(n)$. Thus, the theorem holds by induction.

□ □

Corollary 6.5.4. *The recurrence relation $a(n) = n \cdot a(n - 1) + 1$ and $a(0) = 1$ is equivalent to the equation:*

$$f(n) = n! \cdot \sum_{i=0}^{n} \frac{1}{i!}$$

Proof. If we unroll the relation one we get,

$$a(n) = n \cdot (n - 1) \cdot a(n - 2) + n + 1 = n \cdot (n - 1) \cdot a(n - 2) + \frac{n!}{(n - 1)!} + \frac{n!}{n!}$$

Thus, unrolling the function k times results in the following:

$$a(n) = \left(\prod_{i=n-k+1}^{n} i \right) \cdot a(n - k) + \sum_{i=n-k+1}^{n} \frac{n!}{i!}$$

Now when we set $k = n$ we get,

$$a(n) = n! + \sum_{i=1}^{n} \frac{n!}{i!} = \frac{n!}{0!} + \sum_{i=1}^{n} \frac{n!}{i!} = \sum_{i=0}^{n} \frac{n!}{i!}$$

□ □

Remark on the Second-order Unifier for the Resolution Refutation

Even though we have outlined a resolution refutation of the clause set and gave an example of how it would looked unrolled, we have not provided a second order unifier as required in Ch. 4. The reason why is that the second-order unifier of Ch. 4 was based on the assumption that there will be unique states for each call to the induction invariant of the resolution refutation. What we mean by this is that in the resolution refutation outlined above, the only difference between two calls to the induction invariant can be a switch of variable names and a switch of the schematic sort terms. However, these changes influence the unification lower in the tree near to the roots. Thus, given an accurate and easy to read second-order unifier will require us to first construct a more powerful language, essentially one which can handle these subtle changes.

CHAPTER 7

Extension of Schematic Resolution Calculus

The schematic resolution refutation calculus Ch. 4 used non-nested primitive recursion to build the refutations. A major downside to using non-nested primitive recursion is that the shape of the resolution refutations expressible in the calculus is significantly reduced. However, allowing greater nesting depths for the primitive recursion operator does not fix all of our problems, in that, expressing resolution refutations can be cumbersome and at times impossible without extending the language. Rather than dealing directly with the messy world of nesting primitive recursion operators, we instead change the ordering over which the recursion is defined. Standard primitive recursion is defined over the order of the natural numbers. In the coming sections we define *permutation vectors* , and a method to construct the vectors uniquely, as well as, complexity results. The purpose of this generalized ordering is to allow for more flexibility in defining recursions without the problems of having to keep track of bound parameters and the like. An example of our reason for this generalization would be the factorial. If one wants to construct a function to compute the factorial using primitive recursion, this would be a very simple exercise, however, if one wants to instead to construct all factorial arrangements of n numbers, then a single primitive recursion operator would not be enough. The problem becomes even more difficult when we are looking to construct a sub-set or super-set of the arrangements.

Our goal is to formalize the refutation of the clause set derived from our formal proof of the Non-injectivity Assertion using our generalized recursion. This will require updating Def. 7.16 of [30] or definition 5.8 of [31]. For this work we will work with the definition Def. 7.16 of [30]. Other then updating Def. 7.16 of [30], we have to assure that our new language for resolution refutation has unique states, this will be guaranteed by the ordering we construct in the following section. These unique state will be used to construct a second-order unifier for the resolution refutation. We will also provide an upper bound for the expressiveness of this well ordering, specifically, how many unique states does it have given n, the size of the permutations . The number of unique states in this system has a nice combinatorial interpretations, Namely, the number of objects in all permutations of n objects where each permutation has two labels which are positioned by a set of rules. This work provides a previously unknown (to the best of our knowledge) enumeration function.

7.1 Well-ordering Construction

Definition 7.1.1. *Given $n \in \mathbb{N} \setminus \{0\}$, we define $\langle x_1, \cdots, x_n \rangle$, where $x_1, \cdots, x_n \in \mathbb{N}$, as a* n-ary vector. *The set of all n-ary vectors is $\mathcal{V}^{\langle n \rangle}$.*

Example 7.1.1. *An example of the considered n-ary vectors, where $n = 4$ is $\langle 10, 3, 4, 2 \rangle$.*

Definition 7.1.2. *Given $v \in \mathcal{V}^{\langle n \rangle}$ and $i \in [1, \cdots, n]$, the* ith projection of *v is $\pi_i(v) = x_i$.*

Example 7.1.2. *Using the vector from Ex. 7.1.1, $\pi_2(\langle 10, 3, 4, 2 \rangle) = 3$.*

Definition 7.1.3. *Given $v \in \mathcal{V}^{\langle n \rangle}$, v is a n*-ary permutation vector *if the following holds:*

- $\forall i \in [1, \cdots, n]\ 0 \leq x_i < n$
- $\forall i, j \in [1, \cdots, n]\ i \neq j \rightarrow x_i \neq x_j$

The set of all n-ary permutation vectors is $\mathcal{V}_p^{\langle n \rangle}$.

Example 7.1.3. *The following vector is a permutation vector: $\langle 3, 2, 0, 1 \rangle$.*

Definition 7.1.4. *Given $n, m \in \mathbb{N} \setminus \{0\}$ and $v \in \mathcal{V}^{\langle n \rangle}$, a (n, m)*-labelling of *v $l_v^{(n,m)} : [1, \cdots, n] \times [1, \cdots, m] \rightarrow \{0, 1\}$ is an assignment of labels to the positions of the vector such that*

$$\prod_{j=1}^{m} \left(\sum_{i=1}^{n} l_v^{(n,m)}(i, j) \right) = 1$$

and if $l_v^{(n,m)}(i, j) = 1$ then $\left(\sum_{l=j+1}^{m} \sum_{k=1}^{i-1} l_v^{(n,m)}(k, l) \right) = 0$. The set of all (n,m)-labellings of v is $v^{l(n,m)}$.

Example 7.1.4. *Using the permutation vector from Ex. 7.1.3 $v = \langle 3, 2, 0, 1 \rangle$, we can apply a (4,2)-labelling of v in various ways (i.e. applying two labels to permutation vectors of size 4): $l_v^{(4,2)}(1, 1) = 1$ and $l_v^{(4,2)}(2, 2) = 1$, or $l_v^{(4,2)}(2, 1) = 1$ and $l_v^{(4,2)}(3, 2) = 1$. However, $l_v^{(4,2)}(2, 1) = 1$ and $l_v^{(4,2)}(1, 2) = 1$ is not valid. The labelling $l_v^{(4,2)}(1, 1) = 1$ and $l_v^{(4,2)}(1, 2) = 1$ is also valid. We can represent these labellings directly on the permutation vectors using underscores:*
$l_v^{(4,2)}(1, 1) = 1$ *and* $l_v^{(4,2)}(2, 2) = 1 \quad \equiv \langle \underline{3}_1, \underline{2}_2, 0, 1 \rangle$
$l_v^{(4,2)}(2, 1) = 1$ *and* $l_v^{(4,2)}(3, 2) = 1 \quad \equiv \langle 3, \underline{2}_1, \underline{1}_2, 0, 1 \rangle$
$l_v^{(4,2)}(1, 1) = 1$ *and* $l_v^{(4,2)}(1, 2) = 1 \quad \equiv \left\langle \underline{3}_{\{1,2\}}, 2, 0, 1 \right\rangle$

Definition 7.1.5. The set of (n,m)-labelled permutation vectors $\mathcal{V}_p^{\langle n,m\rangle}$ *is defined as follows:*

$$\mathcal{V}_p^{\langle n,m\rangle} = \bigcup_{v \in \mathcal{V}_p^{\langle n\rangle}} v^{l(n,m)}$$

Definition 7.1.6. *Given* $v \in \mathcal{V}_p^{\langle n,m\rangle}$*, for* $i \in [1, \cdots, m]$ *let* $\rho_i(v)$ *be* the position of the ith label.

Example 7.1.5. *Given the vector* $v = \langle 3, \underline{2}_1, \underline{1}_2, 0, 1\rangle$*,* $\rho_1(v) = 2$ *and* $\rho_2(v) = 3$*.*

From now on we will work with the set $\mathcal{V}_p^{\langle n,2\rangle}$ and will refer to the labels as *top* (for the label 1) and *bottom* (for the label 2). Instead of writing $\left\langle \underline{3}_{\{1,2\}}, 2, 0, 1\right\rangle$ we will write $\left\langle \underline{\overline{3}}, 2, 0, 1\right\rangle$, where the overline refers to top and the underline refers to bottom. Instead of $\rho_1(v)$ and $\rho_2(v)$, we write $\rho_t(v)$ and $\rho_b(v)$.

Definition 7.1.7. *We define the* (n,2)-labelled permutation vector rewrite rules *as follows:* shift $\mapsto^{sh}$: $\mathcal{V}_p^{\langle n,2\rangle} \to \mathcal{V}_p^{\langle n,2\rangle}$, shift top $\mapsto^{sht}$: $\mathcal{V}_p^{\langle n,2\rangle} \to \mathcal{V}_p^{\langle n,2\rangle}$ *and* switch $\mapsto^{sw}$: $\mathcal{V}_p^{\langle n,2\rangle} \to \mathcal{V}_p^{\langle n,2\rangle}$. *For all of the rewrite rule definitions, we will refer to the vector to the left of the arrow as* v *and the vector to the right of the arrow as* w*.*

- *For* $1 \leq i < n$ *and* $\rho_t(v) = \rho_b(v) = i$ *,*
  $\left\langle x_1, \cdots, \underline{\overline{x_i}}, x_{i+1}, \cdots, x_n\right\rangle \mapsto^{sh} \left\langle x_1, \cdots, x_i, \underline{\overline{x_{i+1}}}, \cdots, x_n\right\rangle$
- *For* $1 \leq i < j \leq n$*,* $x_j = max_{\{j|i<j\}}\{x_j < x_i\}$*, and* $\rho_t(v) = \rho_b(v) = i$ *,*
  $\left\langle x_1, \cdots, \underline{\overline{x_i}}, \cdots, x_j, \cdots, x_n\right\rangle \mapsto^{sw} \left\langle x_1, \cdots, \underline{\overline{x_j}}, \cdots, x_i, \cdots, x_n\right\rangle$*, i.e.* $\pi_i(w) = x_j$ *and* $\pi_j(w) = x_i$
- *For* $1 < i \leq j \leq n$*,* $\rho_t(v) = i$ *and* $\rho_b(v) = j$ *,*
  $\left\langle x_1, \cdots, x_{i-1}, \overline{x_i}, \cdots, \underline{x_j}, \cdots, x_n\right\rangle \mapsto^{sht} \left\langle x_1, \cdots, \overline{x_{i-1}}, x_i, \cdots, \underline{x_j}, \cdots, x_n\right\rangle$

Definition 7.1.8. *A* (n,2)-labelled permutation vector chain c of length $1 \leq m$ *is an ordered sequence of vectors* $v_0, \cdots v_{m-1} \in \mathcal{V}_p^{\langle n,2\rangle}$ *such that* $c \equiv v_0 \mapsto^{x_1} v_1 \mapsto^{x_2} \cdots, \mapsto^{x_{m-1}} v_{m-1}$ *where,* $x_1, \cdots x_{m-1} \in \{sh, sw, sht\}$*.*

Example 7.1.6. *The following is a (n,2)-labelled permutation vector chain of length 4:*

$$\left\langle \underline{\overline{3}}, 2, 1, 0\right\rangle \mapsto^{sw} \left\langle \underline{\overline{2}}, 3, 1, 0\right\rangle \mapsto^{sh} \left\langle 2, \underline{\overline{3}}, 1, 0\right\rangle \mapsto^{sw} \left\langle 2, \underline{\overline{1}}, 3, 0\right\rangle$$

Definition 7.1.9. *The* length *of a (n-2)-labelled permutation vector chain c is* $l(c)$*.*

Definition 7.1.10. *A* sub-chain *of a (n,2)-labelled permutation vector chain* $c \equiv v_0 \mapsto^{x_1} v_2 \mapsto^{x_2}, \cdots, \mapsto^{x_{m-1}} v_{m-1}$ *is a (n,2)-labelled permutation vector chain* $c' \equiv w_0 \mapsto^{y_1} w_1 \mapsto^{y_2}, \cdots, \mapsto^{y_{k-1}} w_{k-1}$ *such that* $l(c') \leq l(c)$ *and for some* $0 \leq i \leq j \leq m$*, where* $|j - i| = k$*,* $w_z = v_{i+z}$ *and* $y_{1+l} = x_{i+l}$ *for* $0 \leq z \leq k$ *and* $0 \leq l < k$*. A* proper sub-chain c' *of* c *is one such that* $l(c') < l(c)$ *.*

Example 7.1.7. *The following are sub-chains of the chain in Ex. 7.1.6:*
$c_1 : \left\langle \overline{\underline{3}}, 2, 1, 0 \right\rangle \mapsto^{sw} \left\langle \overline{\underline{2}}, 3, 1, 0 \right\rangle \mapsto^{sh} \left\langle 2, \overline{\underline{3}}, 1, 0 \right\rangle \mapsto^{sw} \left\langle 2, \overline{\underline{1}}, 3, 0 \right\rangle$
$c_2 : \left\langle \overline{\underline{3}}, 2, 1, 0 \right\rangle \mapsto^{sw} \left\langle \overline{\underline{2}}, 3, 1, 0 \right\rangle \mapsto^{sh} \left\langle 2, \overline{\underline{3}}, 1, 0 \right\rangle$
$c_3 : \left\langle \overline{\underline{2}}, 3, 1, 0 \right\rangle \mapsto^{sh} \left\langle 2, \overline{\underline{3}}, 1, 0 \right\rangle \mapsto^{sw} \left\langle 2, \overline{\underline{1}}, 3, 0 \right\rangle$
$c_4 : \left\langle \overline{\underline{2}}, 3, 1, 0 \right\rangle \mapsto^{sh} \left\langle 2, \overline{\underline{3}}, 1, 0 \right\rangle$
Chain c_1 is not a proper sub-chain.

Definition 7.1.11. *An* extension *of a (n,2)-labelled permutation vector chain $c \equiv v_0 \mapsto^{x_1} v_2 \mapsto^{x_2} \cdots, \mapsto^{x_{m-1}} v_{m-1}$ is $c' \equiv c \mapsto^{x_m} v_m$, where $v_m \in \mathcal{V}_p^{\langle n,2 \rangle}$, $v_{m-1} \mapsto^{x_m} v_m$ and $x_m \in \{sh, sw, sht\}$.*

Example 7.1.8. *Using the chains in Ex. 7.1.7, the chain c_3 is a extension of the chain c_4 by the vector $\left\langle 2, \overline{\underline{1}}, 3, 0 \right\rangle$.*

Definition 7.1.12. *Given a $(n, 2)$-labelled permutation vector chain $c \equiv v_0 \mapsto^{x_1} v_2 \mapsto^{x_2} \cdots, \mapsto^{x_{m-1}} v_{m-1}$,* the i^{th} of *c will be denoted by $c(i)$.*

Definition 7.1.13. *A* complete (n,2)-labelled permutation vector chain *c is a labelled permutation vector chain $c \equiv v_0 \mapsto^{x_1} v_2 \mapsto^{x_2} \cdots, \mapsto^{x_{m-1}} v_{m-1}$ such that an extension by $c \mapsto^{x_m} v_m$, where $x_m \in \{sh, sw, sht\}$, would not be possible.*

Example 7.1.9. *Using the chains in Ex. 7.1.7 again, if we were to take the chain c_1 and extend it by the vector $\left\langle \overline{2}, \underline{1}, 3, 0 \right\rangle$ there are no more valid extensions to this chain.*

$$\left\langle \overline{\underline{3}}, 2, 1, 0 \right\rangle \mapsto^{sw} \left\langle \overline{\underline{2}}, 3, 1, 0 \right\rangle \mapsto^{sh} \left\langle 2, \overline{\underline{3}}, 1, 0 \right\rangle \mapsto^{sw} \left\langle 2, \overline{\underline{1}}, 3, 0 \right\rangle \mapsto^{sht} \left\langle \overline{2}, \underline{1}, 3, 0 \right\rangle$$

Lemma 7.1.1. *Every $(n, 2)$-labelled permutation vector chain is extendible to a complete $(n, 2)$-labelled permutation vector chain.*

Proof. By the definition of the rewrite rule $\mapsto^{sht}$, once this rule is applied to a vector no other rewrite rule may be applied to that vector. Thus, if we have a $(n, 2)$-labelled permutation vector chain c such that c is extended by a vector v_0 using the rule $\mapsto^{sht}$ (we assume the c is not already complete and can be extended by $\mapsto^{sht}$), we know that the only rules that can further extend this chain are $\mapsto^{sht}$. Also, because the vectors are of finite size, we know that at some point $c \mapsto^{sht} v_0 \cdots v_k \mapsto^{sht} \left\langle \overline{x_1}, \cdots, \underline{x_j}, \cdots, x_n \right\rangle$, where $1 < j \leq n$ and $0 \leq k$. No rewrite rules can be applied to the vector $\left\langle \overline{x_1}, \cdots, \underline{x_j}, \cdots, x_n \right\rangle$ and therefore we have completed the chain c. However, if we assume that c is a chain such that the $\mapsto^{sht}$ rule cannot be applied, then there are two possible chains. The first possibility is $c \mapsto^{sw} \left\langle \overline{\underline{x_1}}, \cdots, x_n \right\rangle$, which can be extended to a complete (n,2)-labelled permutation vector chain as follows (we refer to $\left\langle \overline{\underline{x_1}}, \cdots, x_n \right\rangle$ as v):

$$c \mapsto^{sw} v \mapsto^{sh} v' \mapsto^{sht} v''$$

The second possibility is $c \mapsto^{sh} \left\langle x_1, \overline{\underline{x_2}}, \cdots, x_n \right\rangle$, which can be extended to a complete (n,2)-labelled permutation vector chain as follows (we refer to $\left\langle x_1, \overline{\underline{x_2}}, \cdots, x_n \right\rangle$ as v):

$$c \mapsto^{sh} v \mapsto^{sht} v'$$

Thus, every labelled permutation vector chain can be extended to a complete labelled permutation vector chain.

□ □

Theorem 7.1.1. *For all $n \in \mathbb{N} \setminus \{0\}$, There are only a finite number of (n,2)-labelled permutation vector chains constructible from the vectors of $\mathcal{V}_p^{\langle n,2\rangle}$ and the rewrite rules $\mapsto^{sh}, \mapsto^{sw}, \mapsto^{sht}$.*

Proof. When $n = 1$, there is only one vector in $\mathcal{V}_p^{\langle 1\rangle}$ and none of the rewrite rules can be applied to it. Let us assume that for all $m \leq n$ the theorem holds and show that it holds for $n+1$, i.e. vectors in the set $\mathcal{V}_p^{\langle n+1,2\rangle}$. It is quite obvious that the rewrite rules $\mapsto^{sh}$ and $\mapsto^{sht}$ can only be applied a finite number of times, i.e. $\mapsto^{sh}$ can only be applied to every vector position once except for the last position in the vector and $\mapsto^{sht}$ can be applied to every position once (given that enough $\mapsto^{sh}$ applications were used) except for the first position. Thus, the only problematic rule is $\mapsto^{sw}$. W.l.o.g we assume that both labels are in the first position of the vector $v_m = \left\langle \underline{\overline{x_1}}, \cdots, x_{n+1} \right\rangle$. If $\rho_t(v) \neq \rho_b(v)$ then the $\mapsto^{sw}$ rule cannot be applied, If $\rho_t(v) = \rho_b(v) = i$ such that $1 < i$ then the maximum number of shift rules that can be applied is reduced by at least one and the problem is equivalent to an instance of the induction hypothesis. Also, we assume that v_m is the last position in the chain c. Now we apply a switch rule to the chain and get $c \mapsto^{sw} \left\langle \cdots, \underline{\overline{x_j}}, \cdots, x_{n+1} \right\rangle$ for some j. This implies that to the right of the first position there are at most $n-1$ values less than the value of the first position which is always the case for vectors in the set $\mathcal{V}_p^{\langle n,2\rangle}$. We know, by the induction hypothesis, that the number of chains is finite for the n case and, thus must also be finite for the case $n+1$.

□ □

Lemma 7.1.2. *Given $v \in \mathcal{V}_P^{\langle n,2\rangle}$, the maximum number of $\mapsto^{sw}$ rules that can be applied at position $i \in [1, \cdots, n]$ without applying a $\mapsto^{sh}$ rule is $n - i$.*

Proof. If $\rho_t(v) \neq \rho_b(v)$, then the number of $\mapsto^{sw}$ rules that can be applied to v is zero at any position because $\mapsto^{sw}$ can only be applied to a vector if $\rho_t(v) = \rho_b(v)$. Thus, we assume that $\rho_t(v) = \rho_b(v) = i$. To maximize the number of $\mapsto^{sw}$ rules applied to v it ought to be the case that for all j such that $i < j \leq n$ it is the case that $\pi_j(v) < \pi_i(v)$. If this is not the case then at some point $\pi_i(v) < \pi_j(v)$ for all j, $i < j \leq n$, and the number of $\mapsto^{sw}$ rules applied will be less than $(n-i)$. Otherwise, one can apply the $\mapsto^{sw}$ rule once to each position j such that $i < j$; remember, there are $n - i$ such positions j. After applying the $(n-i)$ $\mapsto^{sw}$ rules it is the case that $\pi_i(v) < \pi_j(v)$ for all j such that $i < j$, thus no more $\mapsto^{sw}$ rules may be applied. We cannot apply more than $(n-i)$ $\mapsto^{sw}$ rules because the number of position after position i is $n - i$.

□ □

Theorem 7.1.2. *For every $v \in \mathcal{V}_P^{\langle n\rangle}$, there exists at least one $v' \in v^{l(n,2)}$ ($v^{l(n,2)}$ is the set of possible labellings of v) such that there exist a (n,2)-labelled permutation vector chain c ending at v' and starting with $\left\langle \underline{\overline{n-1}}, \cdots, 0 \right\rangle$. We will refer to the vector $\left\langle \underline{\overline{n-1}}, \cdots, 0 \right\rangle$ as $\mathbf{I}^n$.*

Proof. For the case of $n = 1$ there is only one member of $\mathcal{V}_P^{\langle n\rangle}$, namely $\langle 0 \rangle$, and only one possible labelling $\langle \underline{\overline{0}} \rangle$ in $\langle 0 \rangle^{l(n,2)}$. Also, all chains c starting at the vector $\langle \underline{\overline{0}} \rangle$ are of length

$l(c) = 1$ and end at v'. Assume the theorem holds for all $m \leq n$, we now show it holds for $n+1$. If $v = \langle n, \cdots, 0\rangle$ then we choose the labelling $\langle \underline{\overline{n}}, \cdots, 0\rangle$ and we are done, else we can consider the chains containing only the switch operation starting at $\langle \underline{\overline{n}}, \cdots, 0\rangle$:
$c_1 : \mathbf{I}^{n+1} \mapsto^{sw} \langle \underline{\overline{n-1}}, n \cdots, 0\rangle$
$c_2 : \mathbf{I}^{n+1} \mapsto^{sw} v_1 \mapsto^{sw} \langle \underline{\overline{n-2}}, n \cdots, 0\rangle$
$\vdots$
$c_n : \mathbf{I}^{n+1} \mapsto^{sw} v_1 \cdots v_n \mapsto^{sw} \langle \underline{\overline{0}}, n \cdots, 1\rangle$
If v is the permutation of the last vector of any of these chains then we are done. The last vector of every one of the chains, we will refer to them as w_j for $1 \leq j \leq n$, is such that $\pi_2(w_j) = n$ and $\pi_i(w_j) < \pi_2(w_j)$ for $2 < i \leq n+1$. If we apply a shift rule to anyone of the w_j vectors and then map n to $n-1$ then the chosen w_j is equivalent to $\mathbf{I}^n$ modulo the extra $\mapsto^{sht}$ rule that can be applied when the top label is in the second position. However, prior to applying this extra $\mapsto^{sht}$ the derived vector is a labelling of some vector in $\mathcal{V}_P^{\langle n\rangle}$. Thus, even if the extra $\mapsto^{sht}$ rule is not applied we have reduced the problem to a case of the induction hypothesis. Thus, by the the induction hypothesis and extending each chain c_i by a shift rule the theorem holds.
□ □

7.2 Analysis of the Well-ordering

Definition 7.2.1. *Let the set $\mathcal{V}(I^n)$ be the set all vectors which can be found on a (n,2)-labelled permutation vector chain starting at I^n.*

Definition 7.2.2. *Given a vector $v \in \mathcal{V}(I^n)$,* the projection of the top $\pi_t(v)$ *and* the projection of the bottom $\pi_b(v)$ *are projections providing the value stored at the location of the top label and bottom label. Also, let* the position of the top $\rho_t(v)$ *and* the position of the bottom $\rho_b(v)$ *be the position in the vector of the top label and the bottom label.*

Theorem 7.2.1. *Given $v^* \in \mathcal{V}_p^{\langle n,2\rangle}$, if*

$$c \equiv v_0 \mapsto^{x_1} v_1 \cdots v_{m-2} \mapsto^{x_{m-1}} v^* \text{ and } c' \equiv w_0 \mapsto^{y_1} w_1 \cdots w_{k-2} \mapsto^{y_{k-1}} v^*$$

for $l(c') \leq l(c)$ and there exists (n,2)-labelled permutation vector chains c^{v_0}, c^{w_0} starting with $\mathbf{I}^n$ and ending at v_0,w_0 ,then c' must be a sub-chain of c.

Proof. Let us assume for the basecase that $l(c) = 1$. If $l(c') = 1$ as well we know that c' must be a sub-chain of c because $c \equiv c' \equiv v^*$. If $l(c') = 1$ and $1 < l(c)$ then $c' \equiv v^*$ and again and it is a sub-chain of c because c ends in v^*. Given some chain c_2' ending in v^*, where the initial vector of c_2' is derivable from $\mathbf{I}^n$, and $l(c_2') < l(c)$, we assume that all chains c_1' ending in v^*, where the initial vector of c_1' is derivable from $\mathbf{I}^n$, and $l(c_1') \leq l(c_2')$, are sub-chains of c. Using this assumption we show that chains c' ending in v^* and starting with an initial vector derivable from $\mathbf{I}^n$, such that $l(c') = l(c_2') + 1$ are also sub-chains of c. Let c'' be the sub-chain of c' such that the first vector in the chain is removed, i.e. w_0. By the induction hypothesis, the chain c'' is a sub-chain of c. Now we only need to show that the chain $w_0 \mapsto^{y_1} w_1$ is also a sub-chain of some sub-chain of c. By the definition of a sub-chain and the induction hypothesis,

we know that $v_{l(c)-1-k} = w_1$ and that c has a sub-chain c^* whose last vector is $v_{l(c)-1-k}$. Using the induction hypothesis again we know that $w_0 \mapsto^{y_1} w_1$ is a sub-chain of c^*. If we now extend both $w_0 \mapsto^{y_1} w_1$ and c^* by the rewrite rules and vectors in the order they appear in the chain c, The extensions of $w_0 \mapsto^{y_1} w_1$ will remain sub-chains of the extensions of c^*. When both are fully extended we end up with the chains c' and c with c' being a sub-chain of c.
□ □

Corollary 7.2.1. *Given a vector $v \in \mathcal{V}_p^{\langle n,2\rangle}$ such that there exists a (n,2)-labelled permutation vector chain c ending at v and starting with $\mathbf{I}^n$, then c is the unique derivation of v from $\mathbf{I}^n$.*

Proof. By Thm. 7.1.2 and Thm. 7.2.1.
□ □

Corollary 7.2.2. *There exists $v \in \mathcal{V}_p^{\langle n,2\rangle}$ such that a (n,2)-labelled permutation vector chain c starting with $\mathbf{I}^n$ cannot derive v.*

Proof. As an example, $\left\langle \overline{\underline{3}}, 1, 2, 0 \right\rangle$.
□ □

Theorem 7.2.2. *Using the rewrite rules $\mapsto^{sh}$, $\mapsto^{sw}$,and $\mapsto^{sht}$, the number of (n,2)-labelled permutation vector chains starting with the vector $\mathbf{I}^n$, where $0 < n$, is*

$$f(n) = n \cdot f(n-1) + \sum_{k=0}^{n-1} \frac{n!}{k!} \tag{7.1a}$$

$$f(0) = 0 \tag{7.1b}$$

Proof. If $n = 1$ then $\mathbf{I}^1 = \langle \overline{\underline{0}} \rangle$ which cannot have any of the rewrite rules applied to it. Thus, there is only one (1,2)-labelled permutation vector chain starting with $\mathbf{I}^1$, namely the chain of length zero $\langle \overline{\underline{0}} \rangle$. The value for the function f at one is as follows:

$$f(1) = 1 \cdot f(0) + \sum_{k=0}^{0} \frac{1!}{k!} = 0 + \frac{1!}{0!} = 1$$

Let us now assume that for $m \leq n$ the theorem holds, we now show that the theorem holds for $n+1$. The initial vector of any chain is $\mathbf{I}^{n+1} = \langle \overline{\underline{n}}, \cdots, 0 \rangle$. Let us consider the case when we apply the shift rule to $\mathbf{I}^{n+1}$, the resulting chain c^0 is $\mathbf{I}^{n+1} \mapsto^{sh} \left\langle n, \overline{\underline{n-1}}, \cdots, 0 \right\rangle$. If we ignore the first position of the resulting vector, the vector is equivalent to the vector $\mathbf{I}^n$. By the induction hypothesis, the number of (n,2)-labelled permutation vector chains starting with $\mathbf{I}^n$ is $f(n)$. As in Thm. 7.1.2 we will consider the following derivations starting at $\mathbf{I}^{n+1}$:
$c_1 : \mathbf{I}^{n+1} \mapsto^{sw} v_1 \mapsto^{sh} \langle n-1, \overline{\underline{n}} \cdots, 0 \rangle$
$c_2 : \mathbf{I}^{n+1} \mapsto^{sw} v_1 \mapsto^{sw} v_2 \mapsto^{sh} \langle n-2, \overline{\underline{n}} \cdots, 0 \rangle$
$\vdots$
$c_n : \mathbf{I}^{n+1} \mapsto^{sw} v_1 \cdots v_n \mapsto^{sw} \langle 0, \overline{\underline{n}} \cdots, 1 \rangle$
The last vector of every one of the chains, we will refer to it as w, is such that $\pi_2(w) = n$ and

$\pi_i(w) < \pi_2(w)$ for $2 < i \leq n+1$. If we map n to $n-1$ then w is equivalent to $\mathbf{I}^n$. By the induction hypothesis, the number of labelled permutation vector chains starting with $\mathbf{I}^n$ is $f(n)$. Adding up the number of calls to $f(n)$ we get $(n+1) \cdot f(n)$. So far we have not applied the rule $\mapsto^{sht}$ to any chain, however, we can extend each of the chains c_i for $0 \leq i \leq n$ by a $\mapsto^{sht}$ rule application. Thus, we have an additional $n+1$ labelled permutation vector chains giving us a total of $(n+1) \cdot f(n) + (n+1)$ chains. Given the above derivation, we know that the derived formula, $(n+1) \cdot f(n) + (n+1)$ counts the number of chains ending in an application of $\mapsto^{sht}$ which puts the top label into the first position of the vector, i.e. $f'(n+1) = (n+1) \cdot f'(n) + (n+1)$ and $f(0) = 0$. If we unroll the recurrence $f'(n+1)$ we get $\sum_{k=0}^{n} \frac{(n+1)!}{k!}$. Putting the two parts together we get

$$f(n) = n \cdot f(n-1) + \sum_{k=0}^{n-1} \frac{n!}{k!}$$

□ □

Theorem 7.2.3. *The recurrence relation:*

$$f(n) = n \cdot f(n-1) + \sum_{k=0}^{n-1} \frac{n!}{k!}$$

$$f(0) = 0$$

is equivalent to the following summation:

$$f(n) = \sum_{k=0}^{n} k \cdot k! \cdot \binom{n}{k}$$

Proof. Replacing $f(n-1)$ by the next iteration of the relation we get the following formulation:

$$\begin{aligned}
f(n) &= n \cdot \left((n-1) \cdot f(n-2) + \sum_{k=0}^{n-2} \frac{(n-1)!}{k!} \right) + \sum_{k=0}^{n-1} \frac{n!}{k!} \\
&= \left(\frac{n!}{(n-2)!} \cdot f(n-2) + \sum_{k=0}^{n-2} \frac{n!}{k!} \right) + \sum_{k=0}^{n-1} \frac{n!}{k!} \\
&= \left(\frac{n!}{(n-2)!} \cdot f(n-2) + \sum_{k=0}^{n-2} \frac{n!}{k!} \right) + \sum_{k=0}^{n-2} \frac{n!}{k!} + \frac{n!}{(n-1)!} \\
&= \frac{n!}{(n-2)!} \cdot f(n-2) + 2 \cdot \sum_{k=0}^{n-2} \frac{n!}{k!} + \frac{n!}{(n-1)!} \\
&= \frac{n!}{(n-2)!} \cdot f(n-2) + 2 \cdot \sum_{k=0}^{n-2} \frac{n!}{k!} + \sum_{k=1}^{n-(n-1)} \frac{n!}{(n-k)!}
\end{aligned}$$

Now replacing the $f(n-2)$ by the next iteration of the relation we get the following formulation:

$$f(n) = \frac{n!}{(n-2)!} \cdot \left((n-2) \cdot f(n-3) + \sum_{k=0}^{n-3} \frac{(n-2)!}{k!} \right) + 2 \cdot \sum_{k=0}^{n-2} \frac{n!}{k!} + \sum_{k=1}^{n-(n-1)} \frac{n!}{(n-k)!}$$

$$= \left(\frac{n!}{(n-3)!} \cdot f(n-3) + \sum_{k=0}^{n-3} \frac{n!}{k!} \right) + 2 \cdot \sum_{k=0}^{n-2} \frac{n!}{k!} + \sum_{k=1}^{n-(n-1)} \frac{n!}{(n-k)!}$$

$$= \frac{n!}{(n-3)!} \cdot f(n-3) + \sum_{k=0}^{n-3} \frac{n!}{k!} + 2 \cdot \sum_{k=0}^{n-3} \frac{n!}{k!} + 2 \cdot \frac{n!}{(n-2)!} + \sum_{k=1}^{n-(n-1)} \frac{n!}{(n-k)!}$$

$$= \frac{n!}{(n-3)!} \cdot f(n-3) + 3 \cdot \sum_{k=0}^{n-3} \frac{n!}{k!} + \sum_{k=1}^{n-(n-2)} k \cdot \frac{n!}{(n-k)!} \tag{7.2}$$

We can now write Eq. 7.2 in a form for any value less than n.

$$f(n) = \frac{n!}{(n-(m+1))!} \cdot f(n-(m+1)) + (m+1) \cdot \sum_{k=0}^{n-(m+1)} \frac{n!}{k!} + \sum_{k=1}^{n-(n-m)} k \cdot \frac{n!}{(n-k)!}$$

where $0 \leq m < n$. When $m = n-1$ we get the following formula:

$$f(n) = \frac{n!}{0!} \cdot f(0) + n \cdot \sum_{k=0}^{0} \frac{n!}{k!} + \sum_{k=1}^{n-1} k \cdot \frac{n!}{(n-k)!}$$

$$= 0 + n \cdot \frac{n!}{0!} + \sum_{k=1}^{n-1} k \cdot \frac{n!}{(n-k)!}$$

$$= n \cdot \frac{n!}{(n-n)!} + \sum_{k=1}^{n-1} k \cdot \frac{n!}{(n-k)!} = \sum_{k=1}^{n} k \cdot \frac{n!}{(n-k)!} = \sum_{k=0}^{n} k \cdot k! \cdot \binom{n}{k}$$

□ □

7.3 A First-order Theory of Permutation Vectors

In this section, we develop a first order theory of permutation vectors using the construction provided by [20] for first Order array theory . The first-order array theory is quite an expressive theory and will not be used to its full extent in this chapter. However, we rely on some of the results provided in [20] for this theory and thus, use this very expressive framework. Our goal concerning the use of first-order array theory as defined in [20] is to write constraint formulae which are only satisfied by some of the permutation vectors of a given length n. These constraint formulae will be used to guide a the computation tree of a recursive function. The idea is that this recursive function, using the constraint formulae defined using first-order array theory will be used to guide the construction of the resolution refutation tree of a given characteristic clause set, i.e. the resolution refutation of the NiA-schema's characterisitic clause set. This is similar to the work of ch. 4 where primitive recursion is used. In the case of primitive recursion the constraint formula $F(n)$ would be $\exists x(n = s(x)) \vee n = 0$. Though, it is not necessary for the free variable n to be a numeral, i.e. it can also be a defined function symbol, it will also evaluate to a numeral and the domain of the model modelling the constraint formula will also be the natural numbers.

Instead of considering our vectors as they have been considered in the previous sections, we will instead consider the permutation vectors as infinite sequences of integers with a specific form as follows:

$$[(p_1, \cdots, p_n, 0, 0, 0, \cdots), n]$$

where $p \in \mathfrak{S}_n$ and $n \in \mathbb{N}$. We will label the set of objects of this form as $A(\mathfrak{S})_n$. It probably seems as if these changes are unnecessary, in that they are mostly cosmetic, however our goal is to use the same framework as was used in [20]. In [20], the semantics for the first order array theory allowed for arbitrary access of locations in a given array, i.e. one cannot have an out of bounds error. This was mostly likely done to simplify the theory, and in no way stops us from having bounded access. We can formally define $A(\mathfrak{S})_n$ as follows:

Definition 7.3.1. *We define $A(\mathfrak{S})_n$ as the set of all pairs $[a, n]$, such that $n \in \mathbb{N}$ and $a = (p_1, p_2, \cdots)$ such that for all y, where $n < y$, $p_y = 0$, and for all x, where $1 \leq x \leq n$, $0 \leq p_x < n$ and given $1 \leq x_1 < x_2 \leq n$, $p_{x_1} \neq p_{x_2}$.*

Though $A(\mathfrak{S})_n$ is closer to our usage of arrays, it does not have any representation of the top and bottom labels; $A(\mathfrak{S})_n$ only captures the members of the symmetric group of size n and not the permutation vectors. To capture the permutation vectors we need to add the top and bottom positions, but only for permutation vectors which are derivable from $\mathbf{I}^n$ because these are the only ones we are interested in. We can simply extend the objects outlined above to the following:

$$[(p_1, \cdots, p_n, 0, 0, 0, \cdots), t, b, n]$$

where $1 \leq t, b, \leq n$ and the permutation vector it represents is derivable from $\mathbf{I}^n$. We will label the set of objects of this form $A(\mathbf{I}^n)$. Again, using the previous definition, we can define $A(\mathbf{I}^n)$ as follows:

Definition 7.3.2. *We define $A(\mathbf{I}^n)$ as the set of all 4-tuples $[a, t, b, n]$, such that $n, t, b \in \mathbb{N}$ and $1 \leq t, b \leq n$, and $a = (p_1, p_2, \cdots)$ such that for all y, where $n < y$, $p_y = 0$, and for all x, where $1 \leq x \leq n$, $0 \leq p_x < n$ and given $1 \leq x_1 < x_2 \leq n$, $p_{x_1} \neq p_{x_2}$. Also, we assume that, when written as a permutation vector, i.e. $v = \left\langle P_1, \cdots \overline{P_i}, \cdots, \underline{P_j}, \cdots P_n \right\rangle$, where $t = i$ and $b = j$, that there exists a (n,2)-labelled permutation vector chain starting at $\mathbf{I}^n$ and ending at v.*

The objects of $A(\mathbf{I}^n)$ happen to be a bit more complex than the sequences originally considered in [20] being that the objects of $A(\mathbf{I}^n)$ are a 4-tuple rather than a sequence. Though, we can define a projection $\pi_i(\cdot)$ where $i \in [s, t, b, l]$ which gives access to the individual parts.

In the case of gaining access to a particular part of the sequence given $\mathbf{a} \in A(\mathbf{I}^n)$, we do the following, $\pi_s(a)\,[i]$ where $i \in \mathbb{N}$. In the case when we want to gain access to the top element we do the following $\pi_s(a)\,[\pi_t(a)]$. The projection $\pi_l(\cdot)$, returns the size of the permutations in the considered symmetric group. When it is understood which part of the tuple one is attempting to access, it is not necessary to indicate the projections and we adopt the same syntax found in [20]. Our goal is to use $A(\mathbf{I}^n)_n$ to define a new type of recursion for use with schematic CERES, and to do so we need to make a distinction between $A(\mathbf{I}^n)_n$ and $A(\mathbf{I}^n)_n$ without the 4-tuple representing $\mathbf{I}^n$. We will refer to the latter as $A(\mathbf{I}^n)^-$.

Thus, what we will use in this work is the array property fragment of T_A over a model extended by $\pi_i(\cdot)$ where $i \in [s,t,b,l]$ with a restricted domain $A(\mathbf{I}^n)^-$. We will use these formulae as guards for an extension of primitive recursion we will call permutation vector recursion. Essentially we want to restrict the array property fragment to formulae which are valid given the domain $A(\mathbf{I}^n)_n^-$ and the interpretation of the projections $\pi_i(\cdot)$ given above. We will call this fragment $APF(A(\mathbf{I}^n)^-)$. The syntax of this fragment is essentially the same as the syntax found in [20] for the array property fragment. We now provide a modified version of that syntax to include the additional constants. In [20] they use Pressburger arithmetic with the additional array constants and a syntax theory for the elements of the arrays. In our case we do not need these additional element theories being that the elements of our arrays are exactly the same as the indexing language. Thus, we can construct the signature for our theory T_V as follows:

$$\Sigma_V = \Sigma_{\mathbb{N}} \cup \{\cdot\,[\cdot]\,,\cdot\,\{\cdot \leftarrow \cdot\}\}$$

where $\Sigma_{\mathbb{N}}$ is the signature of Pressburger arithmetic, i.e. $\{0,1,=,<,+\}$. The two additional members of Σ_V are *array access* $\cdot\,[\cdot]$ (for example, given $a\,[i]$ we should read this as, given array a return the value stored at i) and array assignment $\cdot\,\{\cdot \leftarrow \cdot\}$ (for example, given $a\,\{i \leftarrow e\}$ we should read this as, given array a store the value e at position i. We can extend this language a little further to accommodate the additional semantics of $A(\mathbf{I}^n)^-$ outlines above.

$$\Sigma_{A(\mathbf{I}^n)^-} = \Sigma_V \cup \{t,b,n\}$$

where t, b, and n are constants which stand for the top position, bottom position and vector size. We only consider one dimensional arrays in this work thus, the term language is quite simplified in comparison to [20]. We will call the theory using the signature $\Sigma_{A(\mathbf{I}^n)^-}$, $T_{A(\mathbf{I}^n)^-}$.

All the literals of our theory $T_{A(\mathbf{I}^n)^-}$ are constructed using the signature outlined above, i.e. atoms are either $=$ or $<$. Formulae are boolean combinations of these literals.

Definition 7.3.3 ([20]). *An* array property *is a formula with the following form:*

$$\forall \bar{i}\,\left(\varphi_I(\bar{i}) \rightarrow \varphi_V(\bar{i})\right)$$

where $\bar{i}$ is a vector of index variables and $\varphi_I(\bar{i})$ and $\varphi_V(\bar{i})$ are the index guards *and* value constraints , *respectively. It is not necessary for the index variables to be bounded by the vector size n, but in most cases they will be. The* height *of the property is the number of quantified index variables in the formula.*

The form of the index guard is constrained by the following grammar [20]:

$$\textbf{\textit{iguard}} \rightarrow \neg\textbf{\textit{iguard}}|\textbf{\textit{iguard}} \vee \textbf{\textit{iguard}}|\textbf{\textit{iguard}} \wedge \textbf{\textit{iguard}}|\textbf{\textit{iguard}}|\textbf{\textit{atom}}$$

$$\textbf{\textit{atom}} \rightarrow \textbf{\textit{expr}} < \textbf{\textit{expr}}|\textbf{\textit{expr}} = \textbf{\textit{expr}}$$

$$\textbf{\textit{expr}} \rightarrow \textit{uvar}|t|b|n|\textbf{\textit{pexpr}}$$

$$\textbf{\textit{pexpr}} \rightarrow \mathbb{N}|\mathbb{N} \cdot \textit{evar}|\textbf{\textit{pexpr}} + \textbf{\textit{pexpr}}$$

where $uvar$ is a universally quantified variable and $evar$ is existentially quantified variable. The form of the value constraint is also constrained, in that any occurrence of a quantified index variable from the quatifier prefix must be used for an array read.

Definition 7.3.4. *The fragment of first order array theory $APF(A(\mathbf{I}^n)^-)$ is essentially all formulae with the array property form or which are part of $T_{A(\mathbf{I}^n)^-}$ and do not contain quantifiers. Also, it must be the case that the semantic constraints outlined above concerning $A(\mathbf{I}^n)^-$ are respected.*

Recursion Iterated by Permutation Vectors

In this section we use $APF(A(\mathbf{I}^n)^-)$ to define the guards of the components of permutation vector recursive functions which we will define shortly. Unlike primitive recursion which does not need a logic for guarding the individual components, i.e. either we have a successor in the object defining the iteration or we are in the zero case, the objects defining the iteration in our case are very complex. Another idiosyncrasy of having such a complex iterating object is the need to have an initializing call for a permutation vector recursive function. This is necessary because iteration of the entire set is only possible when one starts with $\mathbf{I}^n$, thus, we have to enforce this constraint. Also, the end of the recursion is not as clear as it is for primitive recursion in that the recursion stops when we reach a guard guarding a statement that does not recurse or a rewrite rule cannot be applied to the current vector.

To clarify the necessity of this type of recursion, it has so far been a recurrent pattern that proof analysis of complex schematic proofs requires recursion beyond singly nested primitive recursion, but usually not by a second free parameter, but rather an assortment of bounded parameters. Writing such recursion using the standard primitive recursion syntax is extremely cumbersome and veers away from the structure found in the resolution refutations of the extracted characteristic clause sets. Thus, what we are attempting in this work is to use the combinatoric structure derived from the clause set to build a recursion which mimics the structure of the resolution refutation of the clause set. Getting the precise shape of the resolution refutation out of a nested primitive recursion is extremely difficult. The hope is that this method can be generalized to more complex combinatoric sets and provide a structure for representing refutations of schematic clause sets. Even more so, we would like this method to be generalized in the future to handle a variety of refutations, being that we only have the NiA-schema formalized in this language.

Before we define the Permutation Vector Recursive functions (PVR) we need to give syntax representation to the semantic notions developed in the previous sub-section. For example, we defined three rewrite rules on permutation vectors in the previous sub-section, namely $\mapsto^{sw}$, $\mapsto^{sh}$ and $\mapsto^{sht}$. We will add three functions to the syntax of our PVR functions in order to represent these rules, namely, $W(\cdot)$, $H(\cdot)$, and $L(\cdot)$. Given a 4-tuple $v \in A(\mathbf{I}^n)^-$ such $v = (a, t, b, n)$ we define $W(v) = (a\{t \leftarrow a[i]\}\{i \leftarrow a[t]\}, t, b, n)$ where $i = max_j \{a[j] < a[t] \wedge t < j \leq n\}$ and $i = t$ if no value is smaller than the value at $a[t]$. Given a 4-tuple $v \in A(\mathbf{I}^n)^-$ such that $v = (a, t, b, n)$ we define $H(v) = (a, t+1, b+1, n)$,if $t+1, b+1 \leq n$. Given a 4-tuple $v \in A(\mathbf{I}^n)^-$ such that $v = (a, t, b, n)$ we define $L(v) = (a, t-1, b, n)$,if $1 < t$. Also, we define the initial vector representing $\mathbf{I}^n$ as $\mathfrak{I}^n \equiv ((n-1, n-2, \cdots, 1, 0, 0, \cdots), 1, 1, n)$ Now without further ado we provide a definition for Permutation Vector Recursive functions (PVR):

Definition 7.3.5. *Let $n \in \mathbb{N}$ be the free parameter for our recursion and $x_1, \cdots x_e$ are first order variables (we will abbreviate as $\bar{x}$). We define the initial function for the PVR function g as follows:*

$$it_g(\mathfrak{I}^n, \bar{x}) \rightarrow r(\mathfrak{I}^n, \bar{x}) \tag{7.3a}$$

$$it_g(\mathfrak{I}^0, \bar{x}) \rightarrow h(\bar{x}) \tag{7.3b}$$

such that r and h are primitive recursive and r contains at least one instance of $g(v, \bar{x})$ in context, where $v \in \{W(\mathfrak{I}^n), H(\mathfrak{I}^n), L(\mathfrak{I}^n)\}$.

Now we define the secondary part of the function g, let $r^1, \cdots, r^m$ be primitive recursive functions and $j^1, \cdots, j^k$ be in PVR, then we have the following:

$$g(v, \bar{x}) \rightarrow r^1(v, \bar{x}) : \varphi^1 \tag{7.4a}$$

$$\vdots$$

$$g(v, \bar{x}) \rightarrow r^m(v, \bar{x}) : \varphi^m \tag{7.4b}$$

where each r^i can contain an instance of any one of the $j^l(w_l, \bar{x})$ functions (or none at all), $1 \leq l \leq k$, where $w_l \in \{W(v), H(v), L(v), v\}$. It is possible for any of the j^l functions to be g. Also, $\varphi^1, \cdots \varphi^m \in APF(A(\mathbf{I}^n)^-)$ (They are the guarding constraints) where the following holds:

$$APF(A(\mathbf{I}^n)^-) \models \forall (array\ v) \left(\left(\bigvee_{i=1}^{m} \varphi^i(v) \right) \wedge \neg \left(\bigvee_{i=1}^{m} \bigvee_{j=i+1}^{m} \varphi^i(v) \leftrightarrow \varphi^j(v) \right) \right)$$

Example 7.3.1. *The following function computes $n!$:*

$$it_g(\mathfrak{I}^n) \rightarrow (\mathfrak{I}^n[t] + 1) * g(H(\mathfrak{I}^n))$$
$$it_g(\mathfrak{I}^0) \rightarrow 1$$

$$g(v) \rightarrow (v[t] + 1) * g(H(v)) : \varphi_1$$
$$g(v) \rightarrow 1 : \varphi_2$$
$$g(v) \rightarrow 1 : \varphi_3$$

where $\varphi_1(v) \equiv t = b \wedge t < l \wedge 1 < t$,
$\varphi_2(v) \equiv t = b \wedge t = l$,
and $\varphi_3(v) \equiv \neg\varphi_1(v) \wedge \neg\varphi_2(v)$

Notice that the φ_3 case is necessary to capture all the possible vectors because φ_1 and φ_2 are not enough to capture the vector where the top is not equal to the bottom.

Example 7.3.2. *A more complex function computation of $n!$:*

$$it_g(\mathfrak{I}^n) \to g(H(\mathfrak{I}^n))$$
$$it_g(\mathfrak{I}^0) \to 1$$

$$g(v) \to g(H(v)) : \varphi_1$$
$$g(v) \to (v[t] + 1) * h(L(v)) : \varphi_2$$
$$g(v) \to 1 : \varphi_3$$

$$it_h(\mathfrak{I}^n) \to 1$$
$$it_h(\mathfrak{I}^0) \to 1$$

$$h(v) \to (v[t] + 1) * h(L(v)) : \varphi_4$$
$$h(v) \to 1 : \varphi_5$$
$$h(v) \to 1 : \varphi_6$$

where $\varphi_1(v) \equiv t = b \wedge t < l \wedge 1 < t$,
$\varphi_2(v) \equiv t = b \wedge t = l$,
$\varphi_3(v) \equiv \neg\varphi_1(v) \wedge \neg\varphi_2(v)$,
$\varphi_4(v) \equiv b = l \wedge 1 < t \wedge t < l$,
$\varphi_5(v) \equiv b = l \wedge t = 1$,
$\varphi_6(v) \equiv \neg\varphi_4(v) \wedge \neg\varphi_5(v)$.

Example 7.3.3. *This function computes n^n:*

$$it_g(\mathfrak{I}^n) \to (\mathfrak{I}^n[t] + 1) * g(W(\mathfrak{I}^n))$$
$$it_g(\mathfrak{I}^0) \to 1$$

$$g(v) \to h(H(v)) : \varphi_1$$
$$g(v) \to v[t] + 1 : \varphi_2$$
$$g(v) \to 1 : \varphi_3$$

$$it_h(\mathfrak{I}^n) \to 1$$
$$it_h(\mathfrak{I}^0) \to 1$$

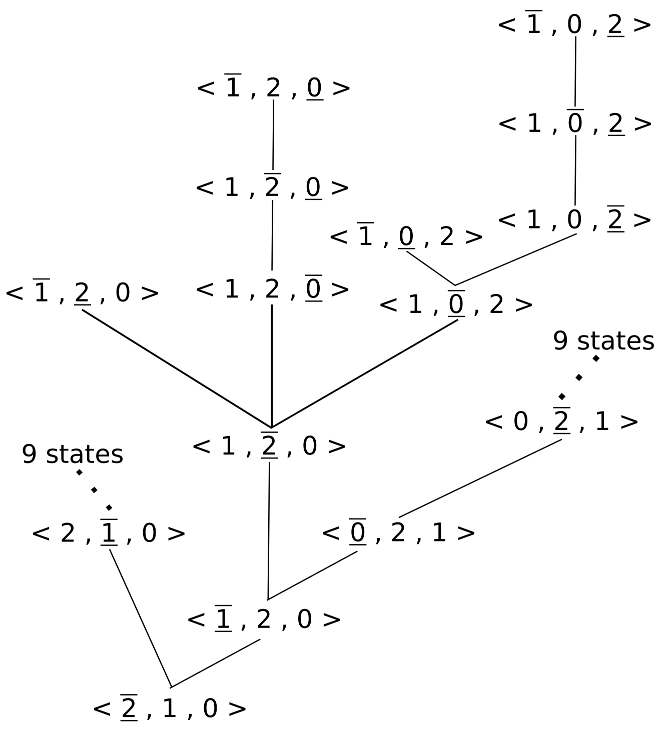


Figure 7.1: An outline of the computation tree for Ex. 7.3.4.

$$h(v) \to (v[t]+1) * g(W(v)) : \varphi_4$$
$$h(v) \to v[t] + 1\varphi_5$$
$$h(v) \to 1 : \varphi_6$$

where $\varphi_1(v) \equiv t = b \wedge v[t] \neq l \wedge t < l \wedge 1 < t$,
$\varphi_2(v) \equiv t = b \wedge t = l \wedge v[t] = l$,
$\varphi_3(v) \equiv \neg\varphi_1(v) \wedge \neg\varphi_2(v)$,
$\varphi_4(v) \equiv t = b \wedge 1 < t \wedge t < l \wedge v[t] = l$,
$\varphi_5(v) \equiv t = b \wedge t = l \wedge v[t] = l$,
$\varphi_6(v) \equiv \neg\varphi_4(v) \wedge \neg\varphi_5(v)$.

Example 7.3.4. *This function follows all paths starting from $\mathfrak{I}^n$ and adds them together:*

$$it_g(\mathfrak{I}^n) \to g(W(\mathfrak{I}^n)) + g(H(\mathfrak{I}^n)) + 1$$
$$it_g(\mathfrak{I}^0) \to 1$$

$$g(v) \to g(W(v)) + g(H(v)) + 1 : \varphi_1$$

$$g(v) \to g(W(v)) + 1 : \varphi_2$$

$$g(v) \to g(W(v)) + g(L(v)) + g(H(v)) + 1 : \varphi_3$$

$$g(v) \to g(L(v)) + g(H(v)) + 1 : \varphi_4$$

$$g(v) \to g(L(v)) + 1 : \varphi_5$$

$$g(v) \to 1 : \varphi_6$$

where $\varphi_1(v) \equiv t = b \wedge t = 1 \wedge v[t] \neq 1$,
$\varphi_2(v) \equiv t = b \wedge t = 1 \wedge v[t] = 1$,
$\varphi_3(v) \equiv t = b \wedge 1 < t \wedge t < l \wedge v[t] \neq t$,
$\varphi_4(v) \equiv t = b \wedge 1 < t \wedge t < l \wedge v[t] = t$,
$\varphi_5(v) \equiv (t = b \wedge t = l) \vee (\equiv t < b \wedge 1 < t)$, *and*
$\varphi_6(v) \equiv t = 1$

We would like to note that this function is computing the function found in Thm. 7.2.3.

Example 7.3.5. *We will use the following function to implement a resolution refutation recursion :*

$$it_g(\mathfrak{I}^n) \to g(W(\mathfrak{I}^n)) + g(H(\mathfrak{I}^n)) + 1$$

$$it_g(\mathfrak{I}^0) \to 1$$

$$g(v) \to g(W(v)) + g(H(v)) + 1 : \varphi_1$$

$$g(v) \to g(H(v)) + 1 : \varphi_2$$

$$g(v) \to g(L(v)) + g(H(v)) + 1 : \varphi_3$$

$$g(v) \to g(L(v)) + 1 : \varphi_4$$

$$g(v) \to g(L(v)) + 1 : \varphi_5$$

$$g(v) \to 1 : \varphi_6$$

$$g(v) \to 1 : \varphi_7$$

where $\varphi_1(v) \equiv t = b \wedge 1 \leq t \wedge t < n \wedge (W(v))[t] \neq v[t]$,
$\varphi_2(v) \equiv t = b \wedge t = 1 \wedge (W(v))[t] = v[t]$,
$\varphi_3(v) \equiv t = b \wedge 1 \leq t \wedge t < l \wedge (W(v))[t] = v[t]$,
$\varphi_4(v) \equiv t = b \wedge t = l$,
$\varphi_5(v) \equiv t \neq b \wedge 1 < t \wedge t < l$,
$\varphi_5(v) \equiv (t = b \wedge t = l) \vee (\equiv t < b \wedge 1 < t)$, *and*
$\varphi_6(v) \equiv t \neq b \wedge t = 1$
$\varphi_7(v) \equiv \bigwedge_{i=0}^{6} \neg\varphi_i(v)$

7.4 Resolution Proof Schema using PVR Functions

In this section we put all the parts so far constructed together to replace Def. 7.16 [30]. In Def. 7.3.5 We limited the number of PVR functions a given PVR function g can call, we call this set of functions $\mathbf{J}^k$ (i.e. the function $j^1, \cdots, j^k$ from Def. 7.3.5), where k is the size. This set also includes the function g itself. To define the resolution proof schema we need to define an ordering on the cross product of $\mathbf{J}^k$ and $A(\mathbf{I}^n)$. We will simple compose an ordering on the members of $\mathbf{J}^k$, namely the ordering $<_{\mathbf{J}}$ and an ordering on $A(\mathbf{I}^n)$ based on the length of the permutation vector chains starting at $\mathbf{I}^n$ (the smaller the chain length the closer to the least element of the ordering $<_{\mathbf{A}}$, i.e.$\mathbf{I}^n$ is the least element). The ordering $<_{\mathbf{J}\times\mathbf{A}}$ is as follows: if $f, g \in \mathbf{J}^k$ and $v, w \in A(\mathbf{I}^n)$, then if $f <_{\mathbf{J}} g$, then $(f, v) <_{\mathbf{J}\times\mathbf{A}} (g, v)$, and if $v <_{\mathbf{A}} w$, then , $(f, v) <_{\mathbf{J}\times\mathbf{A}} (f, w)$. The ordering $<_{\mathbf{J}\times\mathbf{A}}$ is primarily used to mark which is the outer most function and which functions can be called from the current function. Our definition of resolution proof schema will use the same resolution terms which are defined in [30] (Def. 7.10). Also, to simplify the definition we will use the syntactic sugar that given a PVR function g, one could write the function as follows:

$$g^n(\bar{x}) \approx \left\{t^{(it,n)}(\bar{x}) : it^n, t^{(it,0)}(\bar{x}) : it^0, t^1(\bar{x}) : \varphi^1, \cdots, t^m(\bar{x}) : \varphi^m\right\}$$

where the φ^i are the guards and the t_i are the various function bodies, the first two pairs being the initial function components.

Definition 7.4.1 (resolution proof schema). *A resolution proof schema over the variables $x_1, \cdots, x_\alpha \in \mathcal{V}$, $X_1, \cdots, X_\beta \in V_{\text{clset}}$ (abbreviated as $\bar{x}$ and $\bar{X}$), given the finite set of PVR functions $\mathbf{J}^k$, the free parameter n, the set $A(\mathbf{I}^n)$ and the ordering $<_{\mathbf{J}\times\mathbf{A}}$, is a structure $\left(\mathbf{J}^k, A(\mathbf{I}^n), <_{\mathbf{J}\times\mathbf{A}}\right)$ where the functions in $\mathbf{J}^k$ are as follow*

$$\mathbf{J}^k \equiv$$

$$\left\{\begin{array}{c} j_1^n(\bar{x}, \bar{X}) \approx \left\{t_1^{(it,n)}(\bar{x}, \bar{X}) : it_1^n, t_1^{(it,0)}(\bar{x}, \bar{X}) : it_1^0, t_1^1(\bar{x}, \bar{X}) : \varphi_1^1, \cdots, t_1^{m_1}(\bar{x}, \bar{X}) : \varphi_1^{m_1}\right\} \\ \vdots \\ j_k^n(\bar{x}, \bar{X}) \approx \left\{t_k^{(it,n)}(\bar{x}, \bar{X}) : it_k^n, t_k^{(it,0)}(\bar{x}, \bar{X}) : it_k^0, t_k^1(\bar{x}, \bar{X}) : \varphi_k^1, \cdots, t_k^{m_k}(\bar{x}, \bar{X}) : \varphi_k^{m_k}\right\} \end{array}\right\},$$

and assuming that the e^{th} function was given the permutation vector v and uses the l^{th} component, then the term t_e^l is a resolution term which may or may not contain functions $g_1, \cdots, g_p \in \mathbf{J}^k$ where $1 \leq p \leq k$, however when t_e^l contains the said functions the following holds, $(g_1, w) <_{\mathbf{J}\times\mathbf{A}} (j^e, v)$, $\cdots$, $(g_p, w) <_{\mathbf{J}\times\mathbf{A}} (j^e, v)$ where $w \in \{H(v), W(v), L(v), v\}$.

7.5 Using Def. 7.4.1 to construct the refutation for the NiA schema

To make this definition of the resolution refutation work we need a second-order variable which we can instantiate with pairs to get unique first-order variables. Let α be this second order variable. We also need to redefine the iterated max term of Def. 6.5.3 such that it takes a permutation vector and a second-order variable of the form above.

Definition 7.5.1. *We define the primitive recursive term $m(k+1, v, \alpha, t)$ with arity four for $k \in \mathbb{N}$, $\alpha \in \mathcal{V}_2$, and $t \in \mathcal{V}$ and $v \in A(\mathbf{I}^n)$ for $n \geq k$ be defined as follows:*

$$m(k+1, v, \alpha, t) \Rightarrow m(k, v, \alpha, max(s(\alpha(\pi_s(v)\,[k]\,, \pi_t(v))), t)) \tag{7.5a}$$

$$m(0, t) \Rightarrow t \tag{7.5b}$$

We will abbreviate the $\pi_s(v)\,[\pi_t(v)]$ by the expression tv_v or *top value*, where the subscript v is the vector the value is coming from, and $\pi_t(v)$ by t_v for the *top position*, where the subscript v is the vector the value is coming from. We will use the following function to implement a resolution refutation recursion:

$$\begin{gathered} it_g(\mathfrak{I}^n, \alpha, X, Y) \rightarrow \\ res(g(H(\mathfrak{I}^n), \alpha, X', Y \circ f(\alpha(tv_{\mathfrak{I}^n}, t_{\mathfrak{I}^n})) \sim tv_{\mathfrak{I}^n} \vdash); \\ g(W(\mathfrak{I}^n), \alpha, X \circ \vdash f(m((\pi_t(\mathfrak{I}^n) - 1), \mathfrak{I}^n, \alpha, s(\alpha(tv_{\mathfrak{I}^n}, t_{\mathfrak{I}^n})))) \sim tv_{\mathfrak{I}^n}, Y); \\ f(\alpha(tv_{\mathfrak{I}^n}, t_{\mathfrak{I}^n})) \sim tv_{\mathfrak{I}^n}) \\ it_g(\mathfrak{I}^0) \rightarrow Simple\, refutation,\ not\, of\ interest. \end{gathered}$$

$$\begin{gathered} g(v, \alpha, X, Y) \rightarrow \\ res(g(H(v), \alpha, X', Y \circ f(\alpha(tv_v, t_v)) \sim tv_v \vdash); \\ g(W(v), \alpha, X \circ \vdash f(m((\pi_t(v) - 1), v, \alpha, s(\alpha(tv_v, t_v)))) \sim tv_v, Y); \\ f(\alpha(tv_v, t_v)) \sim tv_v) : \varphi_1 \end{gathered}$$

$$\begin{gathered} g(v, \alpha, X, Y) \rightarrow \\ res(g(H(v), \alpha, X', Y \circ f(\alpha(tv_v, t_v) \sim tv_v \vdash); \\ X \circ \vdash f(\alpha(tv_v, t_v)) \sim tv_v; \\ f(\alpha(tv_v, t_v)) \sim tv_v) : \varphi_2 \end{gathered}$$

$$\begin{gathered} g(v, \alpha, X, Y) \rightarrow \\ res(h(v, \alpha, X \circ \vdash f(m((\pi_t(v) - 1), v, \alpha, s(\alpha(tv_v, t_v)))) \sim tv_v, Y); \\ g(H(v), \alpha(tv_w, t_w), X', Y \circ f(\alpha(tv_v, t_v)) \sim tv_v \vdash); \\ f(\alpha(tv_v, t_v)) \sim tv_v)) : \varphi_3 \end{gathered}$$

$$g(v, \alpha, X, Y) \rightarrow h(v, \alpha, X, Y) : \varphi_4$$

$$g(v) \rightarrow \vdash \varphi_5$$

where $\varphi_1(v) \equiv t = b \wedge 1 \leq t \wedge t < n \wedge W(v)[t] \neq v[t]$,
$\varphi_2(v) \equiv t = b \wedge t = 1 \wedge W(v)[t] = v[t]$,
$\varphi_3(v) \equiv t = b \wedge 1 \leq t \wedge t < l \wedge W(v)[t] = v[t]$,
$\varphi_4(v) \equiv t = b \wedge t = l$,
$\varphi_5(v) \equiv \bigwedge_{i=1}^{4} \neg\varphi_i(v)$

This function takes care of Lem. 6.5.2, 6.5.3,6.5.5, & 6.5.1, from the proof of refutation of the NiA schema. The following function h is lower in the ordering than g and covers the refutation down to Cor. 6.5.3. The q function defined in h, finishes the refutation by dealing with the max function nesting.

$$it_h(\mathfrak{I}^n, \alpha, X, Y) \rightarrow \vdash$$
$$it_h(\mathfrak{I}^0) \rightarrow \vdash$$

$$h(v, \alpha, X, Y) \rightarrow$$
$$res(q(v, \alpha, X' \circ f(\alpha(tv_v, t_v)) \sim tv_v, f(m((\pi_t(v) - 1), v, \alpha, s(tv_v, t_v)))) \sim tv_v \vdash);$$
$$h(L(v), \alpha, X \circ \vdash f(m((\pi_t(v) - 1), L(v), \alpha, x)) \sim tv_v, Y);$$
$$f(m((\pi_t(v) - 1), L(v), \alpha, x)) \sim tv_v) : \varphi_1$$

$$h(v, \alpha, X, Y) \rightarrow res($$
$$q(v, \alpha, X' \circ f(\alpha(\pi_s(v)\,[\pi_t(v)]\,, \pi_t(v))) \sim \pi_s(v)\,[\pi_t(v)]\,, f(m((\pi_t(v) - 1), v, \alpha, x)) \sim \pi_s(v)\,[\pi_t(v)] \vdash);$$
$$h(L(v), \alpha, X \circ \vdash f(m((\pi_t(L(v)) - 1), L(v), \alpha, x)) \sim \pi_s(v)\,[\pi_t(v)]\,, Y);$$
$$f(m((\pi_t(L(v)) - 1), L(v), \alpha, x)) \sim \pi_s(v)\,[\pi_t(v)]) : \varphi_2$$

$$h(v, \alpha, X, Y) \rightarrow X : \varphi_3$$

$$h(v, \alpha, X, Y) \rightarrow \vdash : \varphi_4$$

where $\varphi_1(v) \equiv t = b \wedge t = l$,
$\varphi_2(v) \equiv t \neq b \wedge b = l \wedge 1 < t < l$,
$\varphi_3(v) \equiv t = 1 \wedge b = l$,
$\varphi_4(v) \equiv \bigwedge_{i=1}^{3} \neg\varphi_i(v)$,

Definition of q:

$$it_q(\mathfrak{I}^n, \alpha, X) \rightarrow \vdash$$
$$it_q(\mathfrak{I}^0) \rightarrow \vdash$$

$$q(v, \alpha, X) \rightarrow$$
$$res(\alpha \leq \beta \circ X \vdash; p(L(v), \alpha, X' \circ \vdash \alpha(tv_v, t_v) \leq m((t_v - 1), v, \alpha, s(\alpha(tv_v, t_v)));$$
$$\alpha \leq \beta) : \varphi_1$$

$$q(v, \alpha, X) \rightarrow$$
$$res(\alpha \leq \beta \circ X \vdash; p(L(v), \alpha, X' \circ \vdash \alpha(tv_v, t_v) \leq m((t_v - 1), v, \alpha, x); \alpha \leq \beta) : \varphi_2$$

$$q(v, \alpha, X) \rightarrow \vdash : \varphi_2$$

where $\varphi_1(v) \equiv t = b \wedge t = l$,
$\varphi_2(v) \equiv t \neq b \wedge b = l \wedge 1 < t < l$,
$\varphi_3(v) \equiv \bigwedge_{i=1}^{1} \neg\varphi_i(v)$,

The final function p takes care of the unrolling of the nested max function to axioms.

CHAPTER 8

Extraction of Schematic Herbrand Sequents: NiA Schema

In this chapter we show how, using a variation of the resolution refutation found in Chapter 6, one could extract the schematic Herbrand sequent from the proof. This variation is necessary because we would like to make a generalization over the term language used in the refutation. This generalization is based on an iterated max function which has a fixed arity (fixed in the sense that its arity is equivalent to the value of the free parameter unlike in the previous case when we used a set of functions with decreasing arity). We also need to use what we will refer to as *quasi-projections* to reach our goal. These are essentially projections with only propositional rules, no weak quantifier rules. The idea is, we replace the clauses in the clause set which have projections to them by the clauses plus the end sequent formulae. Then we analyse how the local unifiers change the end sequent formulae. As we evaluate the local unifiers we put the term instantiations, based on term depth, into equivalence classes. The idea here is if two formulae which belong to the end sequent have the exact same term instantiations after a unification we can contract the two instances. At the end of the refutation, Some information will be obfuscated, but we will derive Herbrand sequent with term equivalence classes rather than terms such that each of the equivalence classes (the equivalence classes generalize the structure of the terms in the proof) will contain at least one satisfying assignment.

Extraction of a schematic Herbrand sequent is motivated by the fact that using schematic CERES one cannot construct a schematic ACNF , but rather, after instantiating the free parameter, one can take the projections and attach them to the now grounded refutation . On top of this already annoying problem, the schematic ACNF tend to be very complex and very large. We can see from Ch. 6 that even a relatively simple ACNF can have factorial growth. If we are to imagine the growth rate of the ACNF of a much more complex proof, i.e. The Fürstenberg's proof of the infinitude of primes [11], the ACNF for the schematic version would be, for large instantiations of the free parameter, unimaginably large. Extraction of a Herbrand sequent [38] turns out to be a convenient method of retaining some of the information stored in the proof while reducing the size of the output. However, no algorithm exist (has been discovered) for extraction of schematic

Herbrand sequents from schematic cut-free proofs. Partially this is due to the non-existence of a method for attaining a schematic ACNF. What we show in this section is a method of extracting a schematic Herbrand sequents from the refutation of the NiA-schema which can possibly be generalized for other cases as well. The method is specifically devised for refutations using a complex ordering to iterate through the resolution steps, for example refutations formalized using the language found in Ch. 7.

The rest of this work is as follows: first we provide the variation of the refutation of the clause set $C(n)$, then the construction of the equivalence classes, then the construction of the quasi-projections and the quasi-clause set , and finally we walk through the local unification of 8.1 and build the Herbrand sequent.

8.1 Variation on Refutation of $C(n)$ for Herbrand Sequent Extraction

In this section we use a slight variation on the iterated max function used in previous work. The purpose of this variant is that it makes the definition of the equivalence classes much easier. The reason for not using the new definition earlier in this work is that the definition found in this section makes the language for the resolution refutation calculus harder to define. Annoyingly, enough the opposite is true when it comes to extracting the Herbrand sequent. Thus, at least for the concerns of this work, it is better to use these two definitions and two refutations rather than a single definition. The only change we make is to the term language used in the refutation. Being that later on in this chapter we will be constructing equivalence classes based on the term structure used in the refutation, we needed a term structure which allows for this property without the addition of even more complex structure. For a better explanation of the resolution refutation then what is provided in this section, see Sec. 6.5.

Definition 8.1.1. *The primitive recursively defined term $m_n(k, x_0, \cdots, x_n)$ with arity $n+1$ for $k, n \in \mathbb{N}$, $x \in \mathcal{V}$ is defined as follows:*

$$When\, n < k+1, \quad m_n(k+1, x_0, \cdots, x_n) \Rightarrow m_n(k, x_0, \cdots, x_n) \tag{8.1a}$$

$$\begin{aligned} When\, k+1 \leq n, \qquad & m_n(k+1, x_0, \cdots, x_n) \Rightarrow \\ & max(m_n(k, x_0, \cdots, x_n), s(x_{k+1})) \end{aligned} \tag{8.1b}$$

$$m_n(0, x_0, \cdots, x_n) \Rightarrow max(s(x_0), s(x_0)) \tag{8.1c}$$

We will use the abbreviation $\bar{x}_n$ for the list of parameters $x_0, \cdots, x_n$. Also, for simplicity we will always write the term $m_n(k, x_0, \cdots, x_n)$ as $m_n(k, \bar{x}_n)$ where $k \leq n$, being that the function skips evaluation for values higher than n.

Lemma 8.1.1. *Given $0 \leq k \leq n$, the clause $\vdash m_n(k, \bar{x}_n) \leq m_n(n, \bar{x}_n)$ is derivable from (C1),(C2),and (C3).*

Proof. We prove this lemma by induction on the difference between n and k. When $n = k$, the only possibility is $\vdash m_n(n, \bar{x}_n) \leq m_n(n, \bar{x}_n)$, an instance of $(C1)$. Assuming for all differences $u' \leq u$ the lemma holds, we now show for $u+1$. This leaves two possibilities, namely, $u+1 = n-(k-1)$ for $k > 1$ and $u+1 = (n+1)-k$. First we derive the clause $\vdash m_n(k-1, \bar{x}_n) \leq m_n(n, \bar{x}_n)$ using Def. 6.5.4.

$$\dfrac{\begin{matrix} (IH) \\ \vdash P \end{matrix} \qquad \begin{matrix} (C2) \\ max(\beta, \delta) \leq \gamma \vdash \beta \leq \gamma \end{matrix}}{\vdash m_n(k-1, \bar{x}_n) \leq m_n(n, \bar{x}_n)}\ res(\sigma, P)$$

$$P = m_n(k, \bar{x}_n) \leq m_n(n, \bar{x}_n)$$

$$\sigma = \{\beta \leftarrow m_n(k-1, \bar{x}_n), \gamma \leftarrow m_n(n, \bar{x}_n), \delta \leftarrow s(x_k)\}$$

Now we derive the clause $\vdash m_{n+1}(k+1, \bar{x}_{n+1}) \leq m_{n+1}(n+1, \bar{x}_{n+1})$

$$\dfrac{\begin{matrix} (IH) \\ \vdash P \end{matrix} \qquad \begin{matrix} (C2) \\ max(\beta, \delta) \leq \gamma \vdash \beta \leq \gamma \end{matrix}}{\vdash m_{n+1}(k, \bar{x}_{n+1}) \leq m_{n+1}(n+1, \bar{x}_{n+1})}\ res(\sigma, P)$$

$$P = m_{n+1}(k+1, \bar{x}_{n+1}) \leq m_{n+1}(n+1, \bar{x}_{n+1})$$

$$\sigma = \{\beta \leftarrow m_{n+1}(k, \bar{x}_{n+1}), \gamma \leftarrow m_{n+1}(n+1, \bar{x}_{n+1}), \delta \leftarrow s(x_{k+1})\}$$

Thus, by induction the lemma holds.

□ □

Lemma 8.1.2. *Given* $0 \leq k \leq n$*, the clause* $\vdash s(x_k) \leq m_n(n, \bar{x}_n)$ *is derivable from (C1),(C2),and (C3).*

Proof. We know that for every n and k, $\vdash m_n(k, \bar{x}_n) \leq m_n(n, \bar{x}_n)$ is derivable by Lem. 8.1.1. Thus the following derivations show that $\vdash s(x_k) \leq m_n(n, \bar{x}_n)$ is derivable:

$$\dfrac{\begin{matrix} (Lem.8.1.1) \\ \vdash P \end{matrix} \qquad \begin{matrix} (C3) \\ max(\beta, \delta) \leq \gamma \vdash \delta \leq \gamma \end{matrix}}{\vdash s(x_k) \leq m_n(n, \bar{x}_n)}\ res(\sigma, P)$$

$$P = m_n(k, \bar{x}_n) \leq m_n(n, \bar{x}_n)$$

$$\sigma = \{\beta \leftarrow m_n(k-1, \bar{x}_n), \gamma \leftarrow m_n(n, \bar{x}_n), \delta \leftarrow s(x_k)\}$$

□ □

Lemma 8.1.3. *Given* $0 \leq i \leq n$*, the clause* $f(m_n(n, \bar{x}_n)) \sim i, f(x_i) \sim i \vdash$ *is derivable from (C1),(C2),(C3),and (C4$_i$).*

Proof.

$$\dfrac{\overset{(Lem.8.1.2)}{\vdash P} \qquad \overset{(C4_i)}{f(\alpha) \sim i, f(\beta) \sim i, s(\alpha) \leq \beta \vdash}}{f(m_n(n, \bar{x}_n)) \sim i, f(x_i) \sim i \vdash} \; res(\sigma, P)$$

$$P = \vdash s(x_i) \leq m_n(n, \bar{x}_n)$$

$$\sigma = \{\beta \leftarrow m_n(n, \bar{x}_n), \alpha \leftarrow s(x_i)\}$$

□ □

Definition 8.1.2. *Given* $0 \leq n$*,* $-1 \leq k \leq j \leq n$*, and a bijective function* $b : \mathbb{N}_n \to \mathbb{N}_n$ *we define the following formulae:*

$$c_b(k, j, n) = \bigwedge_{i=0}^{k} f(x_{b(i)}) \sim b(i) \vdash \bigvee_{i=k+1}^{j} f(m_n(n, \bar{x}_n)) \sim b(i).$$

The formulae $c_b(-1, -1, n) \equiv \vdash$ *for all values of* n *and* $c_b(-1, n, n) \equiv \vdash \bigvee_{i=0}^{n} f(z) \sim i$ *for all values of* n *.*

Lemma 8.1.4. *Given* $0 \leq n$*,* $-1 \leq k \leq n$ *and for all bijective functions* $b : \mathbb{N}_n \to \mathbb{N}_n$*. the formula* $c_b(k, n, n)$ *is derivable from C(n).*

Proof. We prove this lemma using an induction on k and a case distinction on n When $n = 0$ there are two possible values for k, $k = 0$ or $k = -1$. When $k = -1$ the clause is an instance of (C5). When $k = 0$ we have the following derivation (remember that the bijective function must be $b(0) = 0$):

$$\dfrac{\overset{(C5)}{c_b(-1, 0, 0)} \qquad \overset{(Lem.8.1.3[i \leftarrow 0, k \leftarrow 0])}{f(x_0) \sim 0, f(max(s(x_0), s(x_0))) \sim 0 \vdash}}{c_b(0, 0, 0)} \; res(\sigma, P)$$

$$P = f(max(s(x_0), s(x_0))) \sim 0$$

$$\sigma = \{\alpha \leftarrow f(max(s(x_0), s(x_0))) \sim 0\}$$

When $n > 0$ and $k = -1$ we again trivially have (C5). Now we assume that for all $w \leq k$ for $n > 0$ the theorem holds, we then proceed to prove the theorem holds for $k + 1$. We assume that $k < n$. The following derivation will suffice:

$$\dfrac{\overset{(IH)}{c_b(k, n, n)} \qquad \overset{(Lem.8.1.3[i \leftarrow b(k+1), k \leftarrow n])}{f(x_{k+1}) \sim b(k+1), f(m_n(n, \bar{x}_n)) \sim b(k+1) \vdash}}{c_b(k+1, n, n)} \; res(\sigma, P)$$

$$P = f(m_n(n, \bar{x}_n)) \sim b(k+1)$$
$$\sigma = \{\emptyset\}$$

□

□

The clauses proved derivable by Lem. 8.1.4 can be paired with members of A_n (See Def. 6.5.7) as follows, $c_b(k, n, n)$ is paired with (k, n). Thus, each $c_b(k, n, n)$ is essentially the greatest lower bound of some chain in the ordering $\lessdot_n$ over A_n.

Lemma 8.1.5. *Given* $0 \leq k \leq j \leq n$*, for all bijective functions* $b : \mathbb{N}_n \to \mathbb{N}_n$ *the clause* $c_b(k, j, n)$ *is derivable from C(n).*

Proof. We will prove this lemma by induction over A_n. The base cases are the clauses $c_b(k, n, n)$ from Lem. 8.1.4. Now let us assume that the lemma holds for all clauses $c_b(k, i, n)$ pairs such that, $0 \leq k \leq j < i \leq n$ and for all clauses $c_b(w, j, n)$ such that $0 \leq k < w \leq j \leq n$, then we want to show that the lemma holds for the clause $c_b(k, j, n)$. The following derivation provides proof:

$$\cfrac{\cfrac{\begin{array}{c}(IH[k,j+1])\\ \Pi_b(k), \vdash \Delta_b(k,j), P_b(j+1)\end{array} \qquad \begin{array}{c}(IH[k+1,k+1])\\ \Pi_{b'}(k), f(x_{b'(k+1)}) \sim b'(k+1) \vdash\end{array}}{\Pi_b(k), \Pi_{b'}(k) \vdash \Delta_b(k,j)}\ res(\sigma, P)}{\begin{array}{c}\Pi_b(k) \vdash \Delta_b(k,j)\\ c_b(k,j,n)\end{array}}\ c:l$$

$$P_b(k+1) = f(m_n(n, \bar{x}_n)) \sim b(k+1)$$

$$\Pi_b(k) \equiv \bigwedge_{i=0}^{k} f(x_{b(i)}) \sim b(i)$$

$$\Delta_b(k,j) \equiv \bigvee_{i=k+1}^{j} f(m_n(n, \bar{x}_n)) \sim b(i)$$

$$\sigma = \left\{x_{b'(k+1)} \leftarrow m_n(n, \bar{x}_n)\right\}$$

We assume that $b'(k+1) = b(j+1)$ and that $b'(x) = b(x)$ for $0 \leq x \leq k$.

□ □

Theorem 8.1.1. *Given* $n \geq 0$*,* $C(n)$ *derives* $\vdash$*.*

Proof. By Lem. 8.1.5, The clauses $f(x) \sim 0 \vdash, \cdots, f(x) \sim n \vdash$ are derivable. Thus, we can prove the statement by induction on the instantiation of the clause set. When $n = 0$, the clause (C5) is $\vdash f(x) \sim 0$ which resolves with $f(x) \sim 0 \vdash$ to derive $\vdash$. Assuming that for all $n' \leq n$ the theorem holds we now show that it holds for $n + 1$. The clause (C5) from the clause set $C(n+1)$ is the clause (C5) from the clause set $C(n)$ with the addition of a positive instance of

$\vdash f(\alpha) \sim (n+1)$. Thus, by the induction hypothesis we can derive the clause $\vdash f(\alpha) \sim (n+1)$. By Lem. 8.1.5 we can derive $f(x) \sim (n+1) \vdash$, and thus, resolving the two derived clauses results in $\vdash$.

$\square$

8.2 Natural-like Numbers

In this section we create a class of equivalence classes based on the term language found in the resolution refutation of SubSec. 8.1. The idea is to put the terms found in the resolution refutation into classes such that literals with terms from the same equivalence class can factor. We consider similarity to be having the same maximum term depth considering terms constructed using the pattern defined in this section. The idea is to stop cluttering of the end sequent of the proof with literals which are similar to each other, and thus do not carry any information. We prove that these equivalence classes can be mapped back to the natural numbers and that choosing certain unifiers one can transition through the classes in a similar way to addition transitions between natural numbers.

Definition 8.2.1. *Given a substitution σ, $dom(\sigma)$ is a set of variables $D \subset \mathcal{V}$ such that for every $x \in D$, σ does not map x to itself.*

Definition 8.2.2. *Given a substitution σ, $ran(\sigma)$ is a set of variables $D \subset \mathcal{V}$ such that exists $x \in dom(\sigma)$ for all $y \in D$ such that there exists a term t containing y and $x \leftarrow t[y]$ in σ.*

Definition 8.2.3. *We say that a substitution σ is an (x,k)-*substitution *for $x \in \mathcal{V}$ and $k \in \mathbb{N}$ if $\{x_i\}_{i=0}^{k} \subset \mathcal{V}$, if $\{x_i\}_{i=0}^{k} \not\subseteq dom(\sigma)$ and $ran(\sigma) \subseteq \{x_i\}_{i=0}^{k}$.*

Definition 8.2.4. *Let $\mathcal{N}^k$ for $k \in \mathbb{N}$ and $x \in \mathcal{V}$ be the class of all equivalences classes $\overline{\mathbf{n}}_{(x,k)}$ of the following form where σ is a (x,k)-substitution:*

$$\overline{\mathbf{0}}_{(x,k)} \equiv \{x_i | i \in [0, \cdots, k]\}$$

$$\overline{\mathbf{n}}_{(x,k)} \equiv \left\{ m_k(k, \bar{y}_k)\sigma \;\middle|\; \{y_i\}_{i=0}^{k} \in \mathcal{V}, y_i \neq x_i, i \in [0, \cdots, k], \right.$$
$$\sigma = \left\{ y_0 \leftarrow w_0, \cdots, y_{i-1} \leftarrow w_{i-1}, y_i \leftarrow w_i, y_{i+1} \leftarrow w_{i+1}, \cdots, y_k \leftarrow w_k \right\}$$
$$\left. \left(\bigwedge_{j \in [0,\cdots i-1, i+1, \cdots, k]} \varphi(j) \right), w_i \in \overline{\mathbf{n}-\mathbf{1}}_{(x,k)} \right\}$$

Where $\varphi(j) = \left(\bigvee_{v \in \overline{\mathbf{0}}_{(x,k)}} w_j =_{syn} v \right) \vee \cdots \vee \left(\bigvee_{v \in \overline{\mathbf{n}-\mathbf{1}}_{(x,k)}} w_j =_{syn} v \right)$ and by $=_{syn}$ we mean syntactically equivalent including the names of variables.

As one can see $\overline{\mathbf{1}}_{(x,k)} = m_k(k, \bar{x}_k)$. What is important to see about these equivalence classes is that applying the substitution

$$\sigma_x = \Big\{ y_0 \leftarrow x_0, \cdots, y_{i-1} \leftarrow x_{i-1}, y_i \leftarrow m_k(k, \bar{x}_k), y_{i+1} \leftarrow x_{i+1}, \cdots, y_k \leftarrow x_k \Big\}$$

to a member of the class $\overline{\mathbf{n}}_{(y,k)}$ gives you a member of the class $\overline{\mathbf{n}+\mathbf{1}}_{(x,k)}$. This is exactly the substitution used in the local unification of Lem. 8.1.5. We will use the equivalence classes in the resolution refutation to replace the terms in end-sequent formulae with the equivalence classes. When two formulae have different substitutions applied to them, but the resulting terms are in the same equivalence class, we contract the formulae. Also important to note is how to interpret a formula with equivalence classes replacing terms:

$$\vdash \varphi(\overline{\mathbf{n}}_{(x,k)}) \equiv \vdash \bigvee_{t \in \overline{\mathbf{n}}_{(x,k)}} \varphi(t) \tag{8.3a}$$

$$\varphi(\overline{\mathbf{n}}_{(x,k)}) \vdash \equiv \bigwedge_{t \in \overline{\mathbf{n}}_{(x,k)}} \varphi(t) \vdash \tag{8.3b}$$

The whole idea of the equivalence classes is to group together terms that state roughly the same thing.

The sequent proven derivable in Lem. 8.1.4 after adding the end-sequent formulae and equivalence classes is as follows:

$$\bigwedge_{i=0}^{k} f(x_{b(i)}) \sim b(i), \bigvee_{i=0}^{n} f(\overline{\mathbf{1}}_{(x,n)}) \sim i \vdash \bigvee_{i=0}^{k} (\overline{\mathbf{0}}^{b(i)}_{(x,n)} < \overline{\mathbf{1}}_{(x,n)} \wedge f(\overline{\mathbf{0}}^{b(i)}_{(x,n)}) \sim f(\overline{\mathbf{1}}_{(x,n)})),$$
$$\bigvee_{i=k+1}^{n} f(m_n(n, \bar{x}_n)) \sim b(i)$$

The superscript on the equivalence classes $\overline{\mathbf{0}}_{(x,n)}$ denotes the index of the variable, remember, Lem. 8.1.4 required a bijection.

From now on we assume that in $\mathcal{N}^k$, $k \in \mathbb{N}$.

Definition 8.2.5. *For $\overline{\mathbf{n}}_{(x,k)}, \overline{\mathbf{m}}_{(y,k)} \in \mathcal{N}^k$ we define $\overline{\mathbf{m}}_{(y,k)} \subseteq_k \overline{\mathbf{n}}_{(x,k)}$ (to be read as $\overline{\mathbf{m}}_{(y,k)}$ is contained in or equivalent to $\overline{\mathbf{n}}_{(x,k)}$) as for all $g \in \overline{\mathbf{n}}_{(x,k)}$ there exist $t \in \overline{\mathbf{m}}_{(y,k)}$ such that there exists a subterm g' of g and $t\sigma \equiv_{syn} g'$(syntactically equivalent) where σ is a (x,k)-substitution defined as $\sigma = \{y_0 \leftarrow x_0, \cdots, y_k \leftarrow x_k\}$. We define $\overline{\mathbf{m}} \subset_k \overline{\mathbf{n}}$, the same way except $g' \not\equiv_{syn} g$.*

Definition 8.2.6. *We define an ordering on $\mathcal{N}^k$, $<_k$ such that for $\overline{\mathbf{n}}, \overline{\mathbf{m}} \in \mathcal{N}^k$ $\overline{\mathbf{n}}_{(x,k)} <_k \overline{\mathbf{m}}_{(x,k)}$ iff $\overline{\mathbf{n}}_{(y,k)} \subset_k \overline{\mathbf{m}}_{(x,k)}$.*

Lemma 8.2.1. *The ordering $<_k$ is a linear ordering over $\mathcal{N}^k$ for $k \in \mathbb{N}$*

Proof. Anti-reflexivity is obvious be the definition of $\subset_k$. Anti-symmetry can be shown by the fact that the choice of sub-terms in the definition $\subset_k$ can be bidirectional only in the case when the two classes are equivalent. Thus, by the properties of $\subset_k$ the ordering is Anti-symmetric. For transitivity, it is enough to see that subterm containment is transitive.
□ □

Lemma 8.2.2. *The ordering $<_k$ is a well founded ordering over $\mathcal{N}^k$*

Proof. If we consider the set $\left\{\overline{\mathbf{m}}_{(x,k)}\right\} \subset \mathcal{N}^k$ it is obvious that $<_k$ has a minimal element namely $\overline{\mathbf{m}}_{(x,k)}$. Let us consider all sets $A_{(x,k)} \subset \mathcal{N}^k$ for $x \in \mathcal{V}$ which have cardinality $\leq n$ and assume they have a minimal element, we will now show that sets of cardinality $n+1$ have a minimal element. Let us name the new element $\overline{\mathbf{w}}_{(x,k)}$. Given a set $A_{(x,k)} \subset \mathcal{N}^k$ of cardinality n such that $\overline{\mathbf{l}}_{(x,k)} <_k \overline{\mathbf{w}}_{(x,k)}$ for all $\overline{\mathbf{l}}_{(x,k)} \in A_{(x,k)}$ then the new sub-ordering has the same minimal element as the sub-ordering over $A_{(x,k)}$. If $\overline{\mathbf{w}}_{(x,k)} <_k \overline{\mathbf{l}}_{(x,k)}$ for all $\overline{\mathbf{l}}_{(x,k)} \in A_{(x,k)}$, then $\overline{\mathbf{w}}_{(x,k)}$ is the new minimal element of the sub-ordering over $A_{(x,k)} \cup \left\{\overline{\mathbf{w}}_{(x,k)}\right\}$. If for some $\overline{\mathbf{l}}_{(x,k)} \in A_{(x,k)}$, $\overline{\mathbf{l}}_{(x,k)} <_k \overline{\mathbf{w}}_{(x,k)}$ and for some $\overline{\mathbf{s}}_{(x,k)} \in A_{(x,k)}$, $\overline{\mathbf{w}}_{(x,k)} <_k \overline{\mathbf{s}}_{(x,k)}$, then by transitivity of the ordering we get that $\overline{\mathbf{min}}_{(x,k)} <_k \overline{\mathbf{w}}_{(x,k)}$, where $\overline{\mathbf{min}}_{(x,k)}$ is the minimal element of the sub-ordering over $A_{(x,k)}$. Thus, the sub-ordering over $A_{(x,k)} \cup \{\overline{\mathbf{w}}_x\}$ has a minimal element. This completes the induction step and thus, $<_k$ is a well founded ordering over $\mathcal{N}^k$.
□ □

Lemma 8.2.3. *If $\overline{\mathbf{n}}_{(x,k)} \in \mathcal{N}^k$, then $\overline{\mathbf{n}}_{(x,k)}$ is of finite size.*

Proof. For the basecase, it is obvious that $\overline{\mathbf{0}}_{(x,k)}$ is finite, being that it contains the variables $x_0, \cdots, x_k$. Assuming that the statement holds for all $\overline{\mathbf{w}}_{(x,k)} <_k \overline{\mathbf{n}}_{(x,k)}$, we will show for $\overline{\mathbf{n}}_{(x,k)}$. According to the definition of $\overline{\mathbf{n}}_{(x,k)}$, $\overline{\mathbf{n}}_{(x,k)} = m_k(k, t_0, \cdots, t_k)$. The t_i's are defined such that if $t_i \in \overline{\mathbf{w}}_{(x,k)}$ then, $\overline{\mathbf{w}}_{(x,k)} <_k \overline{\mathbf{n}}_{(x,k)}$. By the induction hypothesis each t_i is chosen from a finite set, thus, $\overline{\mathbf{n}}_{(x,k)}$ must be finite as well. □

Definition 8.2.7. *Given $\overline{\mathbf{n}}_{(x,k)}, \overline{\mathbf{m}}_{(y,k)} \in \mathcal{N}^k$ such that $x, y \in \mathcal{V}$ and are distinct, we define $\overline{\mathbf{n}}_{(x,k)} +_k \overline{\mathbf{m}}_{(y,k)}$ as the equivalence class where all instances of the variables x_i for $i \in [0, \cdots, k]$ are replace with a member of $\overline{\mathbf{m}}_{(y,k)}$.*

Theorem 8.2.1. *Given $\overline{\mathbf{n}}_{(x,k)}, \overline{\mathbf{m}}_{(y,k)} \in \mathcal{N}^k$ such that for $x, y \in \mathcal{V}$ and are distinct , then $\overline{\mathbf{n}}_{(x,k)} +_k \overline{\mathbf{m}}_{(y,k)} = \overline{(\mathbf{n}+\mathbf{m})}_{(y,k)}$ where $+$ is the addition over natural numbers and $n, m \in \mathbb{N}$*

Proof. If we assume $\overline{\mathbf{n}}_{(x,k)} = \overline{\mathbf{0}}_{(x,k)}$, then $\overline{\mathbf{0}}_{(x,k)} +_k \overline{\mathbf{m}}_{(y,k)} \equiv \overline{\mathbf{0}}_{(x,k)}\sigma_y$ where $\sigma_y \equiv \left\{x_0 \leftarrow \overline{\mathbf{m}}_{(y,k)}, \cdots, x_k \leftarrow \overline{\mathbf{m}}_{(y,k)}\right\}$ which is the same as saying $\overline{(\mathbf{0}+\mathbf{m})}_{(y,k)}$ or just $\overline{\mathbf{m}}_{(y,k)}$.

The following shows the cases when $\overline{\mathbf{m}}_{(y,k)} = \overline{\mathbf{0}}_{(y,k)}$ and $\overline{\mathbf{0}}_{(y,k)} <_k \overline{\mathbf{m}}_{(y,k)}$. Let us assume that $\overline{\mathbf{w}}_{(x,k)} +_k \overline{\mathbf{0}}_{(y,k)} = \overline{(\mathbf{w}+\mathbf{0})}_{(y,k)}$ for all $\overline{\mathbf{w}}_{(x,k)} <_k \overline{\mathbf{n}}_{(x,k)}$, we will show that it holds for $\overline{\mathbf{n}}_{(x,k)}$. By Def. 8.2.4 we can write $\overline{\mathbf{n}}_{(x,k)}$ as follows:

$$\begin{aligned}\overline{\mathbf{n}}_{(x,k)} \equiv &\left\{ m_k(k, \bar{z}_k)\sigma \,\middle|\, \{z_i\}_{i=0}^k \in \mathcal{V}, z_i \neq x_i, i \in [0, \cdots, k]\right., \\ &\sigma = \left\{z_0 \leftarrow w_0, \cdots, z_{i-1} \leftarrow w_{i-1}, z_i \leftarrow w_i, z_{i+1} \leftarrow w_{i+1}, \cdots, z_k \leftarrow w_k\right\} \\ &\left., \left(\bigwedge_{j \in [0, \cdots i-1, i+1, \cdots k]} \varphi(j)\right), w_i \in \overline{\mathbf{n}-\mathbf{1}}_{(x,k)}\right\}\end{aligned} \tag{8.4}$$

Where $\varphi(j) = \left(\bigvee_{v \in \overline{\mathbf{0}}_{(x,k)}} w_j =_{syn} v \right) \vee \cdots \vee \left(\bigvee_{v \in \overline{\mathbf{n-1}}_{(x,k)}} w_j =_{syn} v \right)$.

Let z_i be the variable in $dom(\sigma)$ which will be replaced by a member of $\overline{\mathbf{n-1}}_{(x,k)}$ in Eq. 8.4. We can replace the substitution σ in the definition of $\overline{\mathbf{n}}_{(x,k)}$ with some new substitution σ' which is a (y,k)-substitution and is equivalent to σ except that $z_i \leftarrow (\overline{\mathbf{n-1}}_{(x,k)} +_k \overline{\mathbf{0}}_{(y,k)})$ and all x symbols in the range are replaced with y symbols. By the induction hypothesis $(\overline{\mathbf{n-1}}_{(x,k)} +_k \overline{\mathbf{0}}_{(y,k)}) = \overline{(\mathbf{n-1}) + \mathbf{0}}_{(y,k)} = \overline{(\mathbf{n-1})}_{(y,k)}$.

Now that we have shown that addition holds for arbitrary $\overline{\mathbf{n}}_{(x,k)}$ we need to show that it holds for arbitrary $\overline{\mathbf{m}}_{(y,k)}$. Let us assume that $\overline{\mathbf{n}}_{(x,k)} +_k \overline{\mathbf{w}}_{(y,k)} = \overline{(\mathbf{n+w})}_{(y,k)}$ for all $\overline{\mathbf{w}}_{(y,k)} <_k \overline{\mathbf{m}}_{(y,k)}$, we will show that it holds for $\overline{\mathbf{m}}_{(y,k)}$. As in the previous case, we need to consider the formulation of $\overline{\mathbf{m}}_{(y,k)}$ using Def. 8.2.4 as follows:

$$\begin{aligned} \overline{\mathbf{m}}_{(y,k)} &\equiv \left\{ m_k(k, \bar{z}_k)\sigma \middle| \{z_i\}_{i=0}^{k} \in \mathcal{V}, z_i \neq y_i, i \in [0, \cdots, k], \right. \\ \sigma &= \left\{ z_0 \leftarrow w_0, \cdots, z_{i-1} \leftarrow w_{i-1}, z_i \leftarrow w_i, z_{i+1} \leftarrow w_{i+1}, \cdots, z_k \leftarrow w_k \right\} \\ &\left., \left(\bigwedge_{j \in [0, \cdots i-1, i+1, \cdots k]} \varphi(j) \right), w_i \in \overline{\mathbf{m-1}}_{(y,k)} \right\} \end{aligned} \tag{8.5}$$

Where $\varphi(j) = \left(\bigvee_{v \in \overline{\mathbf{0}}_{(y,k)}} w_j =_{syn} v \right) \vee \cdots \vee \left(\bigvee_{v \in \overline{\mathbf{m-1}}_{(y,k)}} w_j =_{syn} v \right)$.

Let z_i be the variable in $dom(\sigma)$ which will be replaced by a member of $\overline{\mathbf{m-1}}_{(y,k)}$ in Eq. 8.5. We can replace the substitution σ in the definition of $\overline{\mathbf{m}}_{(y,k)}$ with some new substitution σ' which is equivalent to σ_y except that $z_i \leftarrow (\overline{\mathbf{n}}_{(x,k)} +_k \overline{\mathbf{m-1}}_{(y,k)})$ and all x symbols in the range are replaced with y symbols. We know by the induction hypothesis that $(\overline{\mathbf{n}}_{(x,k)} +_k \overline{\mathbf{m-1}}_{(y,k)}) = \overline{(\mathbf{n} + (\mathbf{m-1}))}_{(y,k)}$.

To finish this step we need to change the following part of Eq. 8.5 to adhere with the change to the substitution into z_i:

$$\varphi(j) = \left(\bigvee_{v \in \overline{\mathbf{0}}_{(y,k)}} w_j =_{syn} v \right) \vee \cdots \vee \left(\bigvee_{v \in \overline{\mathbf{m-1}}_{(y,k)}} w_j =_{syn} v \right) \tag{8.6}$$

Instead of having the disjunctions range from the equivalence class $\overline{\mathbf{0}}_{(y,k)}$ up to $\overline{\mathbf{m-1}}_{(y,k)}$ they must range up to $\overline{(\mathbf{n} + (\mathbf{m-1}))}_{(y,k)}$, essentially the following equation:

$$\varphi(j) = \left(\bigvee_{v \in \overline{\mathbf{0}}_{(y,k)}} w_j =_{syn} v \right) \vee \cdots \vee \left(\bigvee_{v \in \overline{(\mathbf{n}+(\mathbf{m-1}))}_{(y,k)}} w_j =_{syn} v \right) \tag{8.7}$$

even though it need not be the case that any of the w_j is actually chosen from these extra equivalence classes. Thus, putting the two parts together, we have constructed the set $\overline{\mathbf{n}+\mathbf{m}}_y$ as follows:

$$\begin{aligned}
&\overline{\mathbf{n}+\mathbf{m}}_{(y,k)s} \equiv \left\{ m_k(k,\bar{z}_k)\sigma \,\middle|\, \{z_i\}_{i=0}^{k} \in \mathcal{V}, z_i \neq y_i, i \in [0,\cdots,k]\right., \\
&\sigma = \left\{z_0 \leftarrow w_0, \cdots, z_{i-1} \leftarrow w_{i-1}, z_i \leftarrow w_i, z_{i+1} \leftarrow w_{i+1}, \cdots, z_k \leftarrow w_k\right\} \\
&\left., \left(\bigwedge_{j\in[0,\cdots i-1,i+1,\cdots k]} \varphi(j)\right), w_i \in \overline{\mathbf{n}-(\mathbf{m}-\mathbf{1})}_{(y,k)}\right\}
\end{aligned} \tag{8.8}$$

Where $\varphi(j) = \left(\bigvee_{v\in\overline{\mathbf{0}}_{(y,k)}} w_j =_{syn} v\right) \vee \cdots \vee \left(\bigvee_{v\in\overline{\mathbf{n}+(\mathbf{m}-\mathbf{1})}_{(y,k)}} w_j =_{syn} v\right)$.

Thus, by induction of $\mathcal{N}^k$ $\overline{\mathbf{n}+(\mathbf{m}-\mathbf{1})}_{(x,k)} +_k \overline{\mathbf{1}}_{(y,k)} = \overline{\mathbf{n}+\mathbf{m}}_{(y,k)}$ holds.

$\square$ $\square$

Theorem 8.2.2. *Let σ_x be a substitution such that $dom(\sigma_x) = \{y_0, \cdots, y_k\}$ and for some $y_i \in dom(\sigma_x)$, σ_x maps $y_i \leftarrow m_k(k,\bar{x}_k)$ and for all other cases it is a renaming. Then it is the case that given some $0'_{(y,k)} \in \overline{\mathbf{0}}_{(y,k)}$, the following holds, $0'_{(y,k)}\sigma_x \in \overline{\mathbf{1}}_{(x,k)}$ iff $0'_{(y,k)} = y_i$. However, if we are giving $n'_{(y,k)} \in \overline{\mathbf{n}}_{(y,k)}$ and $\overline{\mathbf{0}}_{(y,k)} <_k \overline{\mathbf{n}}_{(y,k)}$, then $n'_{(y,k)}\sigma_x \in \overline{\mathbf{n}+\mathbf{1}}_{(x,k)}$.*

Proof. Showing $0'_{(y,k)}\sigma_x \in \overline{\mathbf{1}}_{(x,k)}$ iff $0'_{(y,k)} = y_i$ where σ_x maps $y_i \leftarrow m_k(k,\bar{x}_k)$ is simple being that by the definition of the the set $\overline{\mathbf{0}}_{(y,k)}$, there is only one variable in each of its members. This is the major distinction between it and the other classes. Thus, it is obvious that $0'_{(y,k)} = y_i$ must hold for the substitution to result in a term from $\overline{\mathbf{1}}_{(x,k)}$.

For the case of $n'_{(y,k)} \in \overline{\mathbf{n}}_{(y,k)}$ we can prove by induction on $\overline{\mathbf{n}}_{(y,k)}$. When $\overline{\mathbf{n}}_{(y,k)} \equiv \overline{\mathbf{1}}_{(y,k)}$, we see that every $1'_{(y,k)} \in \overline{\mathbf{1}}_{(y,k)}$ contains every variable in $dom(\sigma_x)$ once. This results in $\left\{1'_{(y,k)}\sigma_x\right\} \subset \overline{\mathbf{2}}_{(x,k)}$ because by the definition of $\overline{\mathbf{2}}_{(x,k)}$ at least one position of a term in $\overline{\mathbf{2}}_{(x,k)}$ must be a member of $\overline{\mathbf{1}}_{(x,k)}$. This is guaranteed by σ_x.

Let us assume the theorem holds for $w_{(y,k)} <_k \overline{\mathbf{n}}_{(y,k)}$, we now show for $\overline{\mathbf{n}}_{(y,k)}$. We know that any member $n'_{(y,k)} \in \overline{\mathbf{n}}_{(y,k)}$ must have at least one position which is a member of $\overline{\mathbf{n}-\mathbf{1}}_{(y,k)}$. We refer to this position as $(n-1)'_{(y,k)} \in \overline{\mathbf{n}-\mathbf{1}}_{(y,k)}$. Thus, when considering $\overline{\mathbf{n}}'_{(y,k)}\sigma_x$ we also have to consider $\overline{\mathbf{n}-\mathbf{1}}'_{(y,k)}\sigma_x$ which be that induction hypothesis $\left\{\overline{\mathbf{n}-\mathbf{1}}'_{(y,k)}\sigma_x\right\} \subset \overline{\mathbf{n}}_{(x,k)}$. This means there is a sub-term in every $\overline{\mathbf{n}}'_{(y,k)}\sigma_x$ which is a member of $\overline{\mathbf{n}}_{(x,k)}$ implying that $\left\{\overline{\mathbf{n}}_{(y,k)}\sigma_x\right\} \subset \overline{\mathbf{n}+\mathbf{1}}_{(x,k)}$, Thus the theorem holds by induction.

$\square$ $\square$

Definition 8.2.8. *Given a (x,k)-substitution σ, we define a (x,k,t)-substitution σ_t as a substitution such that $\{x_i\}_{i=0}^{k} = dom(\sigma_t)$ and $\sigma_t = \{x_0 \leftarrow t, \cdots, x_k \leftarrow t\}$ for a term t built from the alphabet $\{0, s()\}$.*

Theorem 8.2.3. *There exists a function $f : \mathcal{N}^k \rightarrow \mathbb{N}$ such that $\overline{\mathbf{n}}_{(x,k)} \in \mathcal{N}^k$ iff $f(\overline{\mathbf{n}}_{(x,k)}) \in \mathbb{N}$, more specifically $f(\overline{\mathbf{n}}_{(x,k)}) = n$, i.e. the n^{th} equivalence class is mapped to the n^{th} natural number.*

Proof. We will assume that for any set $\overline{\mathbf{n}}_{(x,k)} \in \mathcal{N}^k$, the set will be preprocessed into the set $\overline{\mathbf{n}}_{(x,k)}\sigma_0$ where σ_0 is a (x, k, t)-substitution. This can be done simply by stating that $h(\overline{\mathbf{n}}_{(x,k)}) = f(\overline{\mathbf{n}}_{(x,k)}\sigma_0)$ and ignore h for the rest of the proof. We define f on $\overline{\mathbf{n}}_{(x,k)}\sigma_0$ as follows:

$$f(\overline{\mathbf{n}}_{(x,k)}\sigma_0) = \max_{n' \in \overline{\mathbf{n}}_{(x,k)}\sigma_0} \{g(n')\} \tag{8.9}$$

where $\max\{\}$ in this case is the standard arithmetic max function . By Definition 8.2.4, each of the $n' \in \overline{\mathbf{n}}_{(x,k)}\sigma_0$ will be constructed by nesting of syntactic maximum functions of the form $max(s(t), s(t'))$, $max(t, s(t'))$ or at the deepest position in the nestings 0. We can assume that $max(s(t), t')$ does not occur because we always put the nesting in the left position. Thus, g can be defined as follows:

$$g(max(t, s(t'))) = \max\left\{g(t), g(t') + 1\right\} \tag{8.10a}$$

$$g(max(s(t), s(t'))) = \max\left\{g(t) + 1, g(t') + 1\right\} \tag{8.10b}$$

$$g(0) = 0 \tag{8.10c}$$

Again we use the standard arithmetic maximum function to compute g. We will show that this function construction returns exactly the natural number n for the class $\overline{\mathbf{n}}_{(x,k)}$ by induction on $\overline{\mathbf{n}}_{(x,k)}$. Let us now show the base case for the classes $\overline{\mathbf{0}}_{(x,k)}$ and $\overline{\mathbf{1}}_{(x,k)}$. In the case of $\overline{\mathbf{0}}_{(x,k)}$, $\overline{\mathbf{0}}_{(x,k)}\sigma_0$ has only one member, 0. Thus, $f(0) = g(0) = 0$. Thus, $\overline{\mathbf{0}}_{(x,k)} \in \mathcal{N}^k$ iff $0 \in \mathbb{N}$. For the case of $\overline{\mathbf{1}}_{(x,k)}$, $\overline{\mathbf{1}}_{(x,k)}\sigma_0$ has only one member as well, $m_k(k, 0, \cdots, 0)$. Thus, $f(m_k(k, 0, \cdots, 0)) = g(m_k(k, 0, \cdots, 0))$, by the definition of $\overline{\mathbf{1}}_{(x,k)}$ we know that all the positions are going to be $s(0)$, thus, $g(m_k(k, 0, \cdots, 0)) = 1$, $\overline{\mathbf{1}}_{(x,k)} \in \mathcal{N}^k$ iff $1 \in \mathbb{N}$.

We now assume that the statement holds for all $\overline{\mathbf{j}}_{(x,k)} <_k \overline{\mathbf{n}}_{(x,k)}$, we will now show for $\overline{\mathbf{n}}_{(x,k)}$. This time $\overline{\mathbf{n}}_{(x,k)}\sigma_0$ has more than one member, but they will all be treated the same way by g so we will consider a single member of $\overline{\mathbf{n}}_{(x,k)}\sigma_0$, namely n'. For computing $g(n')$ consider the fact that the upper most level of $\overline{\mathbf{n}}'_{(x,k)}$ is equivalent to $m_k(k, t_0, \cdots, t_k)$ by def. 8.2.4. Also by this definition at least one of the t_i is a member of $\overline{\mathbf{n} - \mathbf{1}}_{(x,k)}\sigma_0$, we will refer to it as $\overline{\mathbf{n} - \mathbf{1}}'_{(x,k)}$, thus, $g(\overline{\mathbf{n}}'_{(x,k)}) = g(m_k(k, t_0, \cdots, t_{i-1}, t_i, t_{i+1}, \cdots, t_k))$. We can go even further with

$$g(\overline{\mathbf{n}}'_{(x,k)}) = \max_{i \in [0,k]} \{g(t_i) + 1\} =$$

$$\max\left\{ \max_{j \in [0,i-1] \cup [i+1,k]} \{g(t_j) + 1\}, g(\overline{\mathbf{n} - \mathbf{1}}'_{(x,k)}) + 1 \right\}$$

which is the same as, by the induction hypothesis,

$$g(\overline{\mathbf{n}}'_{(x,k)}) = \max\left\{ \max_{j \in [0,i-1] \cup [i+1,k]} \{g(t_j) + 1\}, n \right\}$$

We know be the definition of n' that none of the t_j can be larger than a member of $\overline{\mathbf{n}-\mathbf{1}}_{(x,k)}\sigma_0$, thus, $g(n') = n$. This computation holds for all member of $\overline{\mathbf{n}}_{(x,k)}\sigma_0$ and thus, $f(\overline{\mathbf{n}}'_{(x,k)}) = n$ proving that $\overline{\mathbf{n}}_{(x,k)} \in \mathcal{N}^k$ iff $f(\overline{\mathbf{n}}_{(x,k)}) \in \mathbb{N}$ or that $\overline{\mathbf{n}}_{(x,k)} \in \mathcal{N}^k$ iff $n \in \mathbb{N}$.
□ □

Definition 8.2.9. *Let $\mathcal{L}^k$ be the language constructed from the alphabet $\{0, s(), max(,)\}$ and a countably infinite set of variables $\mathcal{V}$ which are indexed by values in the interval $[0, k]$. The symbols are to be understood as, 0 is a constant, $s()$ is an unary function, and $max(,)$ is a binary function.*

Definition 8.2.10. *Using the language $\mathcal{L}^k$, We define the class:*

$$\Omega^k = \left\{ t | t \in \mathcal{L}^k \wedge (\overline{\mathbf{n}}_{(x,k)} \in \mathcal{N}^k \rightarrow t \notin \overline{\mathbf{n}}_{(x,k)}) \right\} \tag{8.11}$$

Definition 8.2.11. *We define the relation $\equiv_h$ over the language $\mathcal{L}^k$ (see def. 8.2.10) as given $t, t' \in \mathcal{L}^k$, $t \equiv_h t'$ hold iff there exists $\overline{\mathbf{n}}_{(x,k)} \in \mathcal{N}^k$ such that $t, t' \in \overline{\mathbf{n}}_{(x,k)}$ or $t, t' \in \Omega^k$.*

Lemma 8.2.4. *The relation $\equiv_h$ is an equivalence relation.*

Proof. The relation is based on membership thus, all three properties of an equivalence relation trivially hold.
□ □

8.3 Producing Quasi-Projections

Now that we have the equivalence relation and equivalence classes over the terms found in the refutation of Subsec. 8.1, we need to remove the weak quantifiers from the projections in order to get an idea of which instantiations are part of the Herbrand sequent. This is similar to the work Hetzl et al. [38] on the subject of Herbrand sequent extraction. Their algorithm collects arrays of formulae with along with the specific terms used for the weak quantifier rules. The problem with this process in a schematic proof is that the collecting algorithm would never stop. Instead we will use the equivalence relation $\equiv_h$ and a representing element from each class to mimic what the array does in [38].

Definition 8.3.1 (Quasi-Projection)**.** *A Quasi-projection is defined using the same definitions found in Subsec. 3.4, except weak quantifier rules are skipped when constructing the projections as well as contractions on formulae which have weak quantifier rules applied to them in the original proof, and no end sequent formulae are weakened into the end sequent of the projection. Again, as in previous cases, the projections discussed in this section were constructed from the NiA-schema using the first-order CERES method of projection constructed shortly discussed in Ch. 3.*

In the case of the NiA-schema, the quasi-projection, and projections in general, are extremely simple and thus, we will write down the projection without using the schematic syntax.

Quasi-Projection for (C4$_{n+1}$)

$$\dfrac{s(\alpha) \leq \beta \vdash \alpha < \beta \qquad \begin{array}{c} f(\alpha) \sim i, f(\beta) \sim i \vdash \\ f(\alpha) \sim f(\beta) \end{array}}{\begin{array}{c} f(\alpha) \sim i, f(\beta) \sim i, s(\alpha) \leq \beta \vdash \\ \alpha < \beta \wedge f(\alpha) \sim f(\beta) \end{array}} \wedge : r$$

Quasi-Projection for (C5)

$$\bigvee_{j=0}^{n} f(\beta) \sim j \vdash \bigvee_{j=0}^{n} f(\beta) \sim j$$

Careful observation of Lem. 8.1.2, 8.1.3, and 8.1.4, tells us that we can directly substitute certain terms into the quasi-projections soundly. The reason for this step is to avoid reproving the simple derivations of Subsec. 8.1. The result is the following quasi-projections:

Substituted Quasi-Projection for (C4$_{n+1}$)

$$\dfrac{s(x_i) \leq m_n(n, \bar{x}_n) \vdash m_n(n, \bar{x}_n) < m_n(n, \bar{x}_n) \qquad \begin{array}{c} f(x_i) \sim i, f(m_n(n, \bar{x}_n)) \sim i \vdash \\ f(x_i) \sim f(m_n(n, \bar{x}_n)) \end{array}}{\begin{array}{c} f(x_i) \sim i, f(m_n(n, \bar{x}_n)) \sim i, s(x_i) \leq \beta \vdash \\ x_i < m_n(n, \bar{x}_n) \wedge f(x_i) \sim f(m_n(n, \bar{x}_n)) \end{array}} \wedge : r$$

Substituted Quasi-Projection for (C5)

$$\bigvee_{j=0}^{n} f(m_n(n, \bar{x}_n)) \sim j \vdash \bigvee_{j=0}^{n} f(m_n(n, \bar{x}_n)) \sim j$$

The final step in this process is to attach the substituted quasi-projections to the clauses in the clause set, i.e. (C5) and (C4$_i$) for $0 \leq i \leq n$. The resulting set is no longer a clause set , but rather a formula set. We will work with the parts of this set of formulae which were part of the original clause set and only apply the substitutions to the other parts. The result is the set of formulae:

Definition 8.3.2. *We define the formulae set $C'(n)$ as follows:*

$$(C'1) \qquad \alpha \leq \alpha$$

$$(C'2) \qquad max(\alpha,\beta) \leq \gamma \vdash \alpha \leq \gamma$$

$$(C'3) \qquad max(\alpha,\beta) \leq \gamma \vdash \beta \leq \gamma$$

$$(C'4_0) \qquad \begin{array}{c} f(x_0) \sim 0, f(m_n(n,\bar{x}_n)) \sim 0, \\ s(x_0) \leq m_n(n,\bar{x}_n) \vdash x_0 < m_n(n,\bar{x}_n) \wedge f(x_i) \sim f(m_n(n,\bar{x}_n)) \end{array}$$

$$\vdots \qquad \vdots$$

$$(C'4_{n+1}) \qquad \begin{array}{c} f(x_{n+1}) \sim n+1, f(m_n(n,\bar{x}_n)) \sim n+1, \\ s(x_{n+1}) \leq m_n(n,\bar{x}_n) \vdash x_{n+1} < m_n(n,\bar{x}_n) \wedge f(x_{n+1}) \sim f(m_n(n,\bar{x}_n)) \end{array}$$

$$(C'5) \qquad \begin{array}{c} \bigvee_{i=0}^{n} f(m_n(n,\bar{x}_n)) \sim i \vdash \\ \bigvee_{i=0}^{n} f(m_n(n,\bar{x}_n)) \sim i \end{array}$$

8.4 Resolution Refutation of Sec. 8.1 Using Formula set $C'(n)$

In the following section we us the defined term equivalent classes and the quasi-projections to extract a Herbrand sequent of the refutation in Sec. 8.1. Technically, we are not really using the entire quasi-projection to extract a Herbrand sequent, but rather the end sequent of the quasi-projection only. The reason for only needing the end-sequent is that to extract a herbrand sequent we need to know how the resolution refutation will instantiate the variables within the end sequent parts of the formulae in $C'(n)$. The equivalence classes are used to collect like terms, in our case the like terms are terms with equivalent term depth and same term structure, i.e. using the same functions at the same term depth in the same position. Though, when using such equivalence classes some information is lost, but the validity is preserved.

The idea is as follows, as we follow the formulae from the set $C'(n)$ down the resolution refutation to the end, we apply the substitutions to the end sequent part and put the terms into the appropriate equivalence class. Remember from the previous section, unification can change the equivalence class of the terms in a predictable way. Went we reach the end of the refutation we can remove the equivalence classes by replacing them with a sequence of disjunctions on the right-hand side and conjunctions on the left-hand side. This is our Herbrand sequent. We can also use Thm. 8.2.3 and switch the equivalence classes with natural numbers, however, this would provide us with a Herbrand sequent under a specific interpretation.

For the rest of this section if a specific term t such that $t \in \overline{\mathbf{w}}_{(x,n)}$, is used then we will replace that instance of t by $\overline{\mathbf{w}}_{(x,n)}$. The idea is that by the relation $\equiv_h$ all terms in $\overline{\mathbf{w}}_{(x,n)}$ are the same. In the case of the set $\overline{\mathbf{0}}_{(x,n)}$ we would also like to know the index of the variable, thus we will write $\overline{\mathbf{0}}^{i}_{(x,n)}$, where $i \in [0,n]$ is the index, instead. Essentially, $\overline{\mathbf{0}}^{i}_{(x,n)} \equiv x_i$, but we do not want to use x_i to avoid confusion with the part of the sequent belonging to the end sequent and the part which belongs to the clause/formula set. We refrain from writing the parts of the sequent which are cut-ancestors using natural-like numbers to avoid confusion between what goes to the end-sequent and what does not.

Lemma 8.4.1. *Using the formulae/clause set $C'(n)$ one can derive the following sequent from the proof of lem. 8.1.4 where $0 \leq k \leq n$:*

$$S_b(k,n) = \begin{array}{c} \bigwedge_{i=0}^{k} f(x_{b(i)}) \sim b(i), \bigvee_{i=0}^{n} f(\overline{\mathbf{1}}_{(x,n)}) \sim i \vdash \\ \bigvee_{i=0}^{k} (\overline{\mathbf{0}}^{b(i)}_{(x,n)} < \overline{\mathbf{1}}_{(x,n)} \wedge f(\overline{\mathbf{0}}^{b(i)}_{(x,n)}) \sim f(\overline{\mathbf{1}}_{(x,n)})), \bigvee_{i=k+1}^{n} f(m_n(n,\bar{x}_n)) \sim b(i) \end{array} \tag{8.12}$$

Proof. By Lem. 8.1.3, when the clauses used are replaced by the clause of $C'(n)$ we end up with the derivation of the following sequent for $0 \leq i \leq n$:

$$S_i : \begin{array}{c} f(x_i) \sim i, f(m_n(n,\bar{x}_n)) \sim i, \\ \vdash \overline{\mathbf{0}}^{i}_{(x,n)} < \overline{\mathbf{1}}_{(x,n)} \wedge f(\overline{\mathbf{0}}^{i}_{(x,n)}) \sim f(\overline{\mathbf{1}}_{(x,n)}) \end{array} \tag{8.13}$$

Thus, all that is left to show is that the usage of the S_i sequents in Lem. 8.1.4 results in the sequents $S_b(k,n)$. We will prove this using induction over k in the definition of the sequents $S_b(k,n)$. We need not show that it also holds for the case distinction $n = 0$ and $n > 0$ because this case distinction has already been taken care of by Lem. 8.1.4. Here we are only interested if the unifiers change terms or not. When $k = 0$, the following derivation suffices:

$$\dfrac{\begin{array}{c} (C'5) \\ \bigvee_{i=0}^{n} f(\overline{\mathbf{1}}_{(x,n)}) \sim b(i) \vdash \\ \bigvee_{i=0}^{n} f(m_n(n,\bar{x}_n)) \sim b(i) \end{array} \quad \begin{array}{c} (S_{b(0)}\ Eq.8.13) \\ f(x_{b(0)}) \sim b(0), f(m_n(n,\bar{x}_n)) \sim b(0), \\ \vdash \overline{\mathbf{0}}^{b(0)}_{(x,n)} < \overline{\mathbf{1}}_{(x,n)} \wedge f(\overline{\mathbf{0}}^{b(0)}_{(x,n)}) \sim f(\overline{\mathbf{1}}_{(x,n)}) \end{array}}{S_b(0,n)} \, res(\sigma, P)$$

$$P = f(m_n(n,\bar{x}_n)) \sim b(0)$$

$$\sigma = \{\emptyset\}$$

As one can see the unifier is empty because we are resolving (cutting) like literals. Also, $f(x_{b(0)}) \sim b(0)$ and $\overline{\mathbf{0}}^{b(0)}_{(x,n)} < \overline{\mathbf{1}}_{(x,n)} \wedge f(\overline{\mathbf{0}}^{b(0)}_{(x,n)}) \sim f(\overline{\mathbf{1}}_{(x,n)})$ are in $S_b(0,n)$ because these literals are not the literals we are resolving over, nor does the unification make them equivalent to the literals we are resolving over. The same holds for $\bigvee_{i=0}^{n} f(\overline{\mathbf{1}}_{(x,n)}) \sim b(i)$. Now for the induction hypothesis we assume that the lemma holds for $m \leq k$ and show that it holds for $k + 1$. The following derivation suffices:

$$\dfrac{\begin{array}{c} (IH) \\ S_b(k,n) \end{array} \quad \begin{array}{c} (S_{b(k+1)}) \\ f(x_{b(k+1)}) \sim b(k+1), f(m_n(n,\bar{x}_n)) \sim b(k+1), \\ \vdash \overline{\mathbf{0}}^{b(k+1)}_{(x,n)} < \overline{\mathbf{1}}_{(x,n)} \wedge f(\overline{\mathbf{0}}^{b(k+1)}_{(x,n)}) \sim f(\overline{\mathbf{1}}_{(x,n)}) \end{array}}{S_b(k+1,n)} \, res(\sigma, P)$$

$$P = f(m_n(n, \bar{x}_n)) \sim b(k+1)$$

$$\sigma = \{\emptyset\}$$

Again the unifier is empty and the correct literals are added to the end sequent of this derivation. Thus, by induction the lemma holds.

□ □

Theorem 8.4.1. *Using the clause set $C'(n)$ and given $0 \leq k \leq j < n$, for all bijective functions $b : \mathbb{N}_n \to \mathbb{N}_n$ the formulae $S'_b(k, j, n)$:*

$$\bigwedge_{i=0}^{k} f(x_{b(i)}) \sim b(i), \bigwedge_{w=1}^{n-k+1} \bigvee_{i=0}^{n} f(\overline{\mathbf{w}}_{(x,n)}) \sim i \vdash$$

$$\bigvee_{i=1}^{n-k} \bigvee_{w=i+1}^{n-k+1} (\bar{\mathbf{i}}_x < \overline{\mathbf{w}}_{(x,n)} \wedge f(\bar{\mathbf{i}}_{(x,n)}) \sim f(\overline{\mathbf{w}}_{(x,n)})),$$

$$\bigvee_{i=k+1}^{j} f(m_n(n, \bar{x}_n)) \sim b(i), \bigvee_{i=0}^{k} \bigvee_{w=1}^{n-k+1} (\overline{\mathbf{0}}^{b(i)}_{(x,n)} < \overline{\mathbf{w}}_{(x,n)} \wedge f(\overline{\mathbf{0}}^{b(i)}_{(x,n)}) \sim f(\overline{\mathbf{w}}_{(x,n)}))$$

is derivable given the sequents $S_b(k, n)$ for all k, where $0 \leq k \leq n$ and all bijective function b.

Proof. Our induction uses the same ordering that was used in the proof of lem. 8.1.5. When $j = n$, it holds that $S'_b(k, n, n) = S_b(k, n)$ by lem. 8.4.1. For the base case we must assume the theorem holds for $S'_{b'}(k+1, k+1, n)$ and show that from $S'_{b'}(k+1, k+1, n)$ and $S'_b(k, n, n)$ one could derive $S'_b(k, n-1, n)$.

<u>Basecase</u>

<u>$S'_{b'}(k+1, k+1, n)$</u>

$$\bigwedge_{i=0}^{k+1} f(y_{b'(i)}) \sim b'(i), \bigwedge_{w=1}^{n-k} \bigvee_{i=0}^{n} f(\overline{\mathbf{w}}_y) \sim i \vdash$$

$$\bigvee_{i=1}^{n-k-1} \bigvee_{w=i+1}^{n-k} (\bar{\mathbf{i}}_y < \overline{\mathbf{w}}_y \wedge f(\bar{\mathbf{i}}_y) \sim f(\overline{\mathbf{w}}_y)),$$

$$\bigvee_{i=0}^{k+1} \bigvee_{w=1}^{n-k} (\overline{\mathbf{0}}^{b'(i)}_y < \overline{\mathbf{w}}_y \wedge f(\overline{\mathbf{0}}^{b'(i)}_y) \sim f(\overline{\mathbf{w}}_y))$$

<u>$S'_b(k, n, n)$</u>

$$\bigwedge_{i=0}^{k} f(x_{b(i)}) \sim b(i), \bigvee_{i=0}^{n} f(\overline{\mathbf{1}}_{(x,n)}) \sim i \vdash$$
$$\bigvee_{i=0}^{k} (\overline{\mathbf{0}}^{b(i)}_{(x,n)} < \overline{\mathbf{1}}_{(x,n)} \wedge f(\overline{\mathbf{0}}^{b(i)}_{(x,n)}) \sim f(\overline{\mathbf{1}}_{(x,n)})), \bigvee_{i=k+1}^{n} f(m_n(n,\bar{x}_n)) \sim b(i)$$

We will assume as in lem. 8.1.5 that b and b' where chosen such that $b'(k+1) = b(n)$ and b and b' are equivalent for all values $\leq k$, for every other position in the bijection we just choose either the values of b or b' as long as the final function is also a bijection. The values of the domain greater than k do not matter for members lower in the ordering. We will use the unifier:

$$\sigma_x = \left\{ y_{b'(0)} \leftarrow x_{b(0)}, \cdots, y_{b'(k+1)} \leftarrow m_n(n, x_0, \cdots, x_n), \ldots, y_{b'(n)} \leftarrow x_{b(n)} \right\}$$

Applying σ_x to $S'_{b'}(k+1,k+1,n)$ and using the results of thm. 8.2.2 we get the sequent

$$\underline{S'_{b'}(k+1,k+1,n)\sigma_x}$$

$$f(m_n(n,\bar{x}_n)) \sim b'(k+1), \bigwedge_{i=0}^{k} f(x_{b(i)}) \sim b'(i), \bigwedge_{w=2}^{n-k+1} \bigvee_{i=0}^{n} f(\overline{\mathbf{w}}_{(x,n)}) \sim i \vdash$$
$$\bigvee_{i=1}^{n-k} \bigvee_{w=i+1}^{n-k+1} (\overline{\mathbf{i}}_{(x,n)} < \overline{\mathbf{w}}_{(x,n)} \wedge f(\overline{\mathbf{i}}_{(x,n)}) \sim f(\overline{\mathbf{w}}_{(x,n)})),$$
$$\bigvee_{i=0}^{k} \bigvee_{w=2}^{n-k+1} (\overline{\mathbf{0}}^{b'(i)}_{(x,n)} < \overline{\mathbf{w}}_{(x,n)} \wedge f(\overline{\mathbf{0}}^{b'(i)}_{(x,n)}) \sim f(\overline{\mathbf{w}}_{(x,n)}))$$

After resolving, and factoring $S'_{b'}(k+1,k+1,n)\sigma_x$ and $S'_b(k,n,n)$ we get

$$\underline{S'_b(k,n-1,n)}$$

$$\bigwedge_{i=0}^{k} f(x_{b(i)}) \sim b(i), \bigwedge_{w=1}^{n-k+1} \bigvee_{i=0}^{n} f(\overline{\mathbf{w}}_{(x,n)}) \sim i \vdash \bigvee_{i=k+1}^{n-1} f(m_n(n,\bar{x}_n)) \sim b(i),$$
$$\bigvee_{i=1}^{n-k} \bigvee_{w=i+1}^{n-k+1} (\overline{\mathbf{i}}_{(x,n)} < \overline{\mathbf{w}}_{(x,n)} \wedge f(\overline{\mathbf{i}}_{(x,n)}) \sim f(\overline{\mathbf{w}}_{(x,n)})),$$
$$\bigvee_{i=0}^{k} \bigvee_{w=1}^{n-k+1} (\overline{\mathbf{0}}^{b(i)}_{(x,n)} < \overline{\mathbf{w}}_{(x,n)} \wedge f(\overline{\mathbf{0}}^{b(i)}_{(x,n)}) \sim f(\overline{\mathbf{w}}_{(x,n)}))$$

which is the correct sequent. This case had to be done distinct from the other cases because sequents $S'_b(k,j,n)$ where $j < n$, no longer have the form of lem. 8.4.1. The basecase shows that all of sequents of lem. 8.4.1 have the form stated in the statement of this theorem.

Stepcase

For the step case, we assume that the lemma holds for all sequents $S'_b(k, i+1, n)$ such that, $0 \leq k \leq j < i+1 < n$ and for all sequents $S'_b(w+1, j, n)$ such that $0 \leq k < w+1 \leq j < n$, now we want to show that the lemma holds for the sequent $S'_b(k, j, n)$. Using the sequents $S'_b(k, j+1, n)$ and $S'_{b'}(k+1, k+1, n)$ we will show that $S'_b(k, j, n)$ is derivable in a similar way as it was shown in the basecase. The two sequents are as follows:

$$\underline{S'_b(k, j+1, n)}$$

$$\bigwedge_{i=0}^{k} f(x_{b(i)}) \sim b(i), \bigwedge_{w=1}^{n-k+1} \bigvee_{i=0}^{n} f(\overline{\mathbf{w}}_{(x,n)}) \sim i \vdash \bigvee_{i=k+1}^{j+1} f(m_n(n, \bar{x}_n)) \sim b(i),$$

$$\bigvee_{i=1}^{n-k} \bigvee_{w=i+1}^{n-k+1} (\bar{\mathbf{i}}_{(x,n)} < \overline{\mathbf{w}}_{(x,n)} \wedge f(\bar{\mathbf{i}}_{(x,n)}) \sim f(\overline{\mathbf{w}}_{(x,n)})),$$

$$\bigvee_{i=0}^{k} \bigvee_{w=1}^{n-k+1} (\overline{\mathbf{0}}^{b'(i)}_{(x,n)} < \overline{\mathbf{w}}_{(x,n)} \wedge f(\overline{\mathbf{0}}^{b'(i)}_{(x,n)}) \sim f(\overline{\mathbf{w}}_{(x,n)}))$$

$$\underline{S'_{b'}(k+1, k+1, n)}$$

$$\bigwedge_{i=0}^{k+1} f(y_{b'(i)}) \sim b'(i), \bigwedge_{w=1}^{n-k} \bigvee_{i=0}^{n} f(\overline{\mathbf{w}}_{(y,n)}) \sim i \vdash$$

$$\bigvee_{i=1}^{n-k-1} \bigvee_{w=i+1}^{n-k} (\bar{\mathbf{i}}_{(y,n)} < \overline{\mathbf{w}}_{(y,n)} \wedge f(\bar{\mathbf{i}}_{(y,n)}) \sim f(\overline{\mathbf{w}}_{(y,n)})),$$

$$\bigvee_{i=0}^{k+1} \bigvee_{w=1}^{n-k} (\overline{\mathbf{0}}^{b'(i)}_{(y,n)} < \overline{\mathbf{w}}_{(y,n)} \wedge f(\overline{\mathbf{0}}^{b'(i)}_{(y,n)}) \sim f(\overline{\mathbf{w}}_{(y,n)}))$$

We will assume as in lem. 8.1.5 that b and b' where chosen such that $b'(k+1) = b(j+1)$ and b and b' are equivalent for all values $\leq k$, for every other position in the bijection we just choose either the values of b or b' as long as the final function is also a bijection. The values of the domain greater than k do not matter for members lower in the ordering. We will use the unifier:

$$\sigma_x = \left\{ y_{b'(0)} \leftarrow x_{b(0)}, \cdots, y_{b'(k+1)} \leftarrow m_n(n, x_0, \cdots, x_n), \ldots, y_{b'(n)} \leftarrow x_{b(n)} \right\}$$

Applying σ_x to $S'_{b'}(k+1, k+1, n)$, as in the basecase, and using the results of thm. 8.2.2 we get the sequent:

$$\underline{S'_{b'}(k+1, k+1, n)\sigma_x}$$

$$f(m_n(n, \bar{x}_n)) \sim b'(k+1), \bigwedge_{i=0}^{k} f(x_{b'(i)}) \sim b'(i), \bigwedge_{w=2}^{n-k+1} \bigvee_{i=0}^{n} f(\overline{\mathbf{w}}_{(x,n)}) \sim i \vdash$$

$$\bigvee_{i=1}^{n-k} \bigvee_{w=i+1}^{n-k+1} (\bar{\mathbf{i}}_{(x,n)} < \overline{\mathbf{w}}_{(x,n)} \wedge f(\bar{\mathbf{i}}_{(x,n)}) \sim f(\overline{\mathbf{w}}_{(x,n)})),$$

$$\bigvee_{i=0}^{k} \bigvee_{w=2}^{n-k+1} (\overline{\mathbf{0}}^{b'(i)}_{(x,n)} < \overline{\mathbf{w}}_{(x,n)} \wedge f(\overline{\mathbf{0}}^{b'(i)}_{(x,n)}) \sim f(\overline{\mathbf{w}}_{(x,n)}))$$

After resolving, and factoring $S'_{b'}(k+1, k+1, n)\sigma_x$ and $S'_b(k, j+1, n)$ we get

$$\underline{S'_b(k, j, n)}$$

$$\bigwedge_{i=0}^{k} f(x_{b'(i)}) \sim b(i), \bigwedge_{w=1}^{n-k+1} \bigvee_{i=0}^{n} f(\overline{\mathbf{w}}_{(x,n)}) \sim i \vdash \bigvee_{i=k+1}^{j} f(m_n(n, \bar{x}_n)) \sim b(i),$$

$$\bigvee_{i=1}^{n-k} \bigvee_{w=i+1}^{n-k+1} (\bar{\mathbf{i}}_{(x,n)} < \overline{\mathbf{w}}_{(x,n)} \wedge f(\bar{\mathbf{i}}_{(x,n)}) \sim f(\overline{\mathbf{w}}_{(x,n)})),$$

$$\bigvee_{i=0}^{k} \bigvee_{w=1}^{n-k+1} (\overline{\mathbf{0}}^{b(i)}_{(x,n)} < \overline{\mathbf{w}}_{(x,n)} \wedge f(\overline{\mathbf{0}}^{b(i)}_{(x,n)}) \sim f(\overline{\mathbf{w}}_{(x,n)}))$$

Which is the desired sequent.

□ □

Corollary 8.4.1. *Using the clause set $C'(n)$, the sequent $S'_b(0, 0, n)$ is derivable:*

$$f(x_{b(0)}) \sim b(0), \bigwedge_{w=1}^{n+1} \bigvee_{i=0}^{n} f(\overline{\mathbf{w}}_{(x,n)}) \sim i \vdash$$

$$\bigvee_{i=1}^{n} \bigvee_{w=i+1}^{n+1} (\bar{\mathbf{i}}_{(x,n)} < \overline{\mathbf{w}}_{(x,n)} \wedge f(\bar{\mathbf{i}}_{(x,n)}) \sim f(\overline{\mathbf{w}}_{(x,n)})),$$

$$\bigvee_{w=1}^{n+1} (\overline{\mathbf{0}}^{b(0)}_{(x,n)} < \overline{\mathbf{w}}_{(x,n)} \wedge f(\overline{\mathbf{0}}^{b(0)}_{(x,n)}) \sim f(\overline{\mathbf{w}}_{(x,n)}))$$

8.5 Deriving the Herbrand Sequent

Theorem 8.5.1. *By Cor. 8.4.1, the proof of Thm. 8.1.1 and using an instance of $(C'5)$, we can derive the following formula:*

$$\bigwedge_{w=0}^{n+1} \bigvee_{i=0}^{n} f(\overline{\mathbf{w}}_x\sigma_0) \sim i \vdash \bigvee_{i=0}^{n} \bigvee_{w=i+1}^{n+1} (\bar{\mathbf{i}}_x\sigma_0 < \overline{\mathbf{w}}_x\sigma_0 \wedge f(\bar{\mathbf{i}}_x\sigma_0) \sim f(\overline{\mathbf{w}}_x\sigma_0))$$

Proof. By Cor. 8.4.1 setting $b(0) = i$ for $i \in [0, \cdots, n]$ the following sequents will result:

$$\underline{S''_i(0, 0, n)}$$

$$f(x_i) \sim i, \bigwedge_{w=1}^{n+1} \bigvee_{i=0}^{n} f(\overline{\mathbf{w}}_{(x,n)}) \sim i \vdash$$

$$\bigvee_{i=1}^{n} \bigvee_{w=i+1}^{n+1} (\bar{\mathbf{i}}_{(x,n)} < \overline{\mathbf{w}}_{(x,n)} \wedge f(\bar{\mathbf{i}}_{(x,n)}) \sim f(\overline{\mathbf{w}}_{(x,n)})),$$

$$\bigvee_{w=1}^{n+1} (\overline{\mathbf{0}}^{i}_{(x,n)} < \overline{\mathbf{w}}_{(x,n)} \wedge f(\overline{\mathbf{0}}^{i}_{(x,n)}) \sim f(\overline{\mathbf{w}}_{(x,n)}))$$

Now for each of the sequents $S''_i(0,0,n)$ we will apply the following substitution

$$\sigma_0 = \{x_0 \leftarrow 0, \cdots, x_n \leftarrow 0\}$$

we will refer to these sequents as $S'_i(0,0,n)\sigma_0$. Using the following instantiation of $C'5$

$$C'5(0) : \vdash f(0) \sim 0, \cdots, f(0) \sim n$$

We can resolve, for each distinct $i \in [0, \cdots, n]$, $S''_i(0,0,n)\sigma_0$ with $C'5(0)$. After factoring the resulting sequent is as follows:

$$\bigwedge_{w=0}^{n+1} \bigvee_{i=0}^{n} f(\overline{\mathbf{w}}_{(x,n)}\sigma_0) \sim i \vdash$$

$$\bigvee_{i=0}^{n} \bigvee_{w=i+1}^{n+1} (\bar{\mathbf{i}}_{(x,n)}\sigma_0 < \overline{\mathbf{w}}_{(x,n)}\sigma_0 \wedge f(\bar{\mathbf{i}}_{(x,n)}\sigma_0) \sim f(\overline{\mathbf{w}}_{(x,n)}\sigma_0))$$

$\square$

$\square$

Corollary 8.5.1. *By thm 8.2.3 and EQ. 8.3, the sequent of thm. 8.5.1 is validity equivalent to the sequent:*

$$\bigwedge_{w=0}^{n+1} \bigvee_{i=0}^{n} f(w) \sim i \vdash \bigvee_{i=0}^{n} \bigvee_{w=i+1}^{n+1} (i < w \wedge f(i) \sim f(w))$$

Which is the pigeonhole principle for n.

Essentially, what was done in this chapter is we constructed a equivalence class structure which contained all the terms used in the resolution refutation, replaced the terms in the refutations with these equivalence classes, showed how the unifiers used in the resolution refutation convert one equivalence class into another, and in the end, found out which equivalence classes will be used at the end of the resolution refutation. As a last step we reduced all the equivalence classes to natural numbers in order to get a more elegant version of the Herbrand sequent, though, even without conversion, the resulting Herbrand sequent with equivalence classes, after expansion of the classes, would still represent a valid formula. This entire process is extremely complex and is very sensitive to changes, i.e. it will probably not work for another proof without a huge assortment of changes. However, the concept of extracting an infinite sequence of valid formulae from an infinite sequence of proofs after cut elimination, such that the sequence of formulae capture the propositional structure of the proofs, is itself a daunting task, and just being able to perform this task for a single, quite complex proof is a step towards a more general method. We hope that such a method, as outlined in this chapter can be generalized and partially automatized for a sub-class of first order proof schema, possible proof schema with a term signature containing only monadic functions . Though, one has to keep in mind, do to the necessity of skolemization for schematic CERES , that such a restriction also reduced the allowed quantifier nesting. However, we leave such considerations to future work.

CHAPTER 9

A Generalization of the NiA-Schema

In this chapter we introduce a generalization of the NiA-schema (gNiA-schema) found in Chapter 6. Though, the gNiA-schema uses two parameters in its construction, any instantiation of one of the parameters results in a proof schema as described in [30, 31]. Setting the second parameter to zero results in the NiA-schema. The successor cases increases the number of distinct values in the domain which the function must map to the same value in the range. Essentially the gNiA-schema schema states that given a total function over the natural numbers with a range of size n, there exists a subset of the domain of size m which maps to the same value in the range. In some sense, such a generalization is not difficult to imagine or prove, however, we chose this generalization because it significantly increases the combinatorial complexity of the problem. Unlike in the case of the NiA-schema where we only need to check n spaces till we reach a contradiction, in the gNiA-schema we need to check a number of spaces which is a linear combination of the parameters n and m.

We also need to define a defined predicate symbol to represent equality over m positions. The following schematic predicate accurately represents this problem:

$$eq(n+1,p) \rightarrow \exists q(p < q \wedge eq(n,q) \wedge f(q) \sim f(p))$$

$$eq(0,p) \rightarrow \exists q(p < q \wedge f(q) \sim f(p))$$

Rather than the end sequent containing the following formula as in the NiA-schema:

$$\exists p \exists q(p < q \wedge f(p) \sim f(q))$$

it contains the following formula:

$$\exists q \; eq(m+1,q)$$

where the existential quantifiers are iterated using the eq defined predicate symbol. To mark which is the active parameter we will use an underscore. The formal proof in **LKS** can be found in C.1.

Our goal in this chapter is, rather than attempting another proof analysis procedure as we did with the NiA-schema, to focus primarily on the combinatorial analysis of the derived clause set. Unexpectedly a combinatorial analysis of the set of derivable clauses from the clause set $C(n)$ was essential to the proof analysis of the NiA-schema. Essentially, the structure of the symmetric group was used to find a resolution refutation for all instances of the clause set. In the case of the gNiA-schema, the combinatorial structure is more complex and nowhere close to as much literature exist on the required structures. However, a subset structure seems to be required for a refutation of the gNiA-schema as it was for the refutation of the NiA-schema. Though, when we interpret the parameter n as the number of symbols in an alphabet and the parameter $m-1$ as the number of occurrences allowed of each symbol, we can think of it as a word problem . When n is set to three and m is left free we found that there is a bijection between these words and a specific type of permutations . We postulate that proof schema can be used to learn more about the combinatorial structure of certain logical arguments, proofs, and problems.

9.1 Two Parameter NiA-schema Characteristic Clause set

Rather than going through the entire clause set schema again, we can use the NiA-schema as a backbone and build in the changes necessary for the gNiA-schema. As one can tell, there are only a few changes required for the clause $C^g(n)$:

$$(C^g1) :\vdash \alpha \leq \alpha$$

$$(C^g2) : max(\alpha,\beta) \leq \gamma \vdash \alpha \leq \gamma$$

$$(C^g3) : max(\alpha,\beta) \leq \gamma \vdash \beta \leq \gamma$$

$$(C^g4_0) \left(\bigwedge_{i=0}^{m+1} f(\alpha_i) \sim 0\right) \wedge \left(\bigwedge_{i=0}^{m} s(\alpha_i) \leq \alpha_{i+1}\right) \vdash$$

$$\vdots$$

$$(C^g4_n) : \left(\bigwedge_{i=0}^{m+1} f(\alpha_i) \sim n\right) \wedge \left(\bigwedge_{i=0}^{m} s(\alpha_i) \leq \alpha_{i+1}\right) \vdash$$

$$(C^g5) :\vdash \bigvee_{i=0}^{n} f(\alpha) \sim i$$

Before we go ahead with the combinatorial analysis of the clause set $C^g(n)$, we want to make clear what the goal of this analysis is, essentially we would like to deduce the size of set of clauses found in the "middle" of the resolution refutation. When we say middle we are referring to Lem. 6.5.2. Given the clauses of Lem. 6.5.2 and one instance of the clause $(C5)$ a refutation can be found. So by middle we mean a set of derived clauses which makes the refutation and has some combinatorial significance.

In the case of the gNiA-schema, a theorem such as Lem. 6.5.2 has not been proven as of yet, but we do have an idea of what such clauses would look like from the structure of the clauses shown to exist by Lem. 6.5.2. Part of the difficulty of proving a theorem such as Lem. 6.5.2 for $C^g(n)$ is the construction of the terms, which we do not really care about for the combinatorial analysis, rather we really want to know the sequent structure instead.

In the case of NiA-schema the sequent structure of Lem. 6.5.2 is w literals on the left side and k on the right side where $w + k = n + 1$ and for every subset of $\{0, \cdots, n\}$ of size w we take a permutation of those w symbols and make a different sequent. In the case of gNiA-schema we also have to take into account how many times the symbols can occur which is less than the free parameter $m - 1$. If we were to continue with the proof analysis we would also need to take subsets into account as well, as was done with the NiA-schema. However, we will only work with the basic structure for now.

In the next section we will investigate the case for $n = 3$ and m is the free parameter, and show its relationship to a set of permutations.

9.2 Combinatorial Analysis of Clause set interpretation

The NiA-schema Clause set and its refutation provided interesting combinatorial results which were not expected at the start of the proof analysis. These unexpected results are based on the consequences of Lem. 6.5.2 & 6.5.3. Using these two lemmata we can construct a new clause set which has a size equivalent to the subset sum. Thus, it is the construction of this intermediary clause set which provides information concerning the combinatoric nature of the clause set. Attempting the same same type of formal mathematical argument for the more complex clause set found in this chapter proved extremely difficult, even for the one parameter case when we fix the number of symbols and increase the number of positions which must be shown equal. Thus, instead of attempting a mathematical argument we used more empirical methods to construct an analogy between a combinatoric set and a derived clause set . Though, this was still not the most trivial exercise we were able to find a few combinatoric results as well as gain a better understanding of the clause set. We will first provide the combinatoric results in this chapter and in the following chapter show the empirical process we carried out in order to reach the analogy.

We specifically work with the Two Parameter NiA clause set instantiated with 3 symbols and m equality. The case with 2 symbols and m equality resulted in some interesting combinatorial results, however we left discussion of these results to the next chapter being that they are more empirical in nature. Though for the case of 3 symbols and m equality the combinatorial results we found have a direct correspondence to the schematic CERES method , as in recursions which maybe aid in the process of defining a resolution refutation for the clause set. We were not able to construct a mathematically proven resolution refutation for this case. Though, we did assume that patterns found in the NiA-schema clause set will be found in the Two Parameter NiA-schema clause set, namely that we will need all permutations of the symbols plus all possible configurations of the occurrences of each symbol. In the NiA-schema, the end sequent is satisfied once a symbol shows up a second time, however in the generalized version, The symbol can show up $m - 1$ times before satisfying the end sequent, which adds an extra level of complexity to the permutations. These considerations are pretty much the same considerations which were made what defining and proving Lem. 6.5.2 & 6.5.3. For the rest of this section, we focus on showing that the intermediate clause set, based on the above combinatoric problem which is used in the empirical analysis, has a nicer combinatoric representation. This is shown through the construction of a bijection. Though, this "nicer" combinatoric representation does not directly aid in understanding the clause set, but rather it aids in understanding the recursion needed to refute the clause set. As well shall see, the recursion used to describe the problem requires a multiplication of two recursive functions. We, as of yet, do not have a way of interpreting this within the scope of resolution derivations, but intuitively, the best way to understand such a construction is a number of recursive calls based on the result of the other function. This is something that none of the languages used to describe the resolution refutations for proof schema are able to describe, and thus, this proof schema leave a plethora of open problems to address.

Introduction

In the work "Enumerating Permutations by their Run Structure" [33], an enumeration was provided for the problem, how many atomic permutations are there with three runs of equal length n ($RAP3(n)$) . As noted by the authors, this integer sequence was previously unknown and has been added to the OEIS (Online Encyclopedia of Integer Sequences) as A241193. In this work, we provide a deeper investigation of the set $RAP3(n)$ by providing a proof of equinumerosity with a seemingly unrelated word problem, namely, given a finite alphabet Σ of size $1 \leq m$ how many words can be constructed such that one symbol $s \in \Sigma$ occurs n times and the other symbols occur at most n times, this is the *letter repetition problem ($LRP_{\mathbf{s}}(n,m)$ where* s *is the symbol occurring n times)* . We show that equinumerosity exists between $LRP_{\mathbf{s}}(n,m)$ and $RAP3(n+1)$. While, the concept of atomic permutation perfectly captures the concept of three runs of equal length n (fixing the outer two positions fixes the run structure), this is not the case for a variety of other questions one might have concerning run structure in permutations. In those cases, it might be much easier to use sub-problems of $LRP_{\mathbf{s}}(n,m)$ and find a bijection to the permutations with a specific run structure. Also, we can apply a slight modification to $LRP_{\mathbf{s}}(n,m)$, namely $GLRP(n,m)$ (the *Generalized letter repetition problem ($GLRP(n,m)$)* to handle more complex word structures which have occurred in resolution refutations of clause sets. For $GLRP(n,m)$ we add the constraint that one symbol occurs $n+1$ times and it must be the last symbol. Though this is only a symbolic extension in that the structure is nearly identical to the problem it encapsulates. Increasing the number of symbols which occur exactly $n+1$ times in the word from 1 to k will be labelled as the *Extended $GLRP(n,m,k)$* problem, or $EGLRP(n,m,k)$. We conjecture that bijections can be drawn between some sub-problems of $EGLRP(n,m,k)$ and the number of atomic permutations with m runs each of length n ($RAP(n,m)$).

Background

We will use the following notation for integer intervals: $[a \dots b] \doteq [a,b] \cap \mathbb{N} = [a, a+1, \dots, b]$ with the special case $[n] \doteq [1 \dots n]$. Let $S \subset \mathbb{N}$ contain n elements. Being that we are only considering subsets of $\mathbb{N}$ we will use $<$ as the canonical ordering of $\mathbb{N}$. A permutation of S is a linear ordering $p_1, p_2, \dots, p_n$ of the elements of S. We denote the set of all $n!$ permutations of S by $\mathfrak{S}_S$ and by $\mathfrak{S}_n$ if $S = [n]$. It is customary to write the permutation as the word $p = p_1 p_2 \cdots p_n$ and we can identify the permutation p with the bijection $p(i) = p_i$. In some cases, when it is necessary to avoid confusion, we will denote permutations as $p_1|p_2|\cdots|p_n$, were $|$ denotes the end of one element and the beginning of another.

In [33], the set of *rising and falling atomic permutations* $\mathfrak{A}^{\pm}_S \subset \mathfrak{S}_S$, where S is an n-element set of $\mathbb{N}$ is defined. The *rising atomic permutations* $\mathfrak{A}^{+}_S$ are the atomic permutations starting with the minimum element (S_{min}) of S and end with the maximum element (S_{max}) of S. The *falling atomic permutations* $\mathfrak{A}^{-}_S$ start with S_{max} and end with S_{min}. To correct a error found in [33], the sets $\mathfrak{A}^{+}_S$ and $\mathfrak{A}^{-}_S$ have cardinality $(n-2)!$ when the cardinality of $|S| = n$, thus, the total cardinality of $\mathfrak{A}^{\pm}_S$ is $2 \cdot (n-2)!$.

An *ascending (descending) sequence* of a permutation $p \in \mathfrak{S}_S$ where $|S| = n$, is a set indices $\mathbf{A} \subseteq [1, \ldots, n]$ such that for $a, b \in \mathbf{A}$, $a < b$ ($b < a$), then $p(a) < p(b)$ ($p(a) > p(b)$). If we enforce the constraint that $\mathbf{A}$ is a sub-interval of $[1, \ldots, n]$ such that any extension of the interval is no longer strictly ascending (descending), then we have a *run* of the permutation p. The length of a run r is $l(r) = |A| - 1$. the positions within the permutation such that to the left the run is descending and to the right the run is ascending are called *valleys*. The positions within a permutation such that to the left the run is ascending and to the right the run is descending are called *peaks*. The set of peaks of a permutation p will be labelled as $\mathbf{P}_p$ and the set of valleys will be labelled as $\mathbf{V}_p$. We also define an ordering $<_x$, where $x \in \{\mathbf{p}, \mathbf{v}\}$ ($\mathbf{p}$ for peaks, $\mathbf{v}$ for valley), on $\mathbf{P}_p$ and $\mathbf{V}_p$, such that given $\mathbf{p}_1, \mathbf{p}_2 \in \mathbf{P}_p$ (or $\mathbf{v}_1, \mathbf{v}_2 \in \mathbf{V}_p$) $\mathbf{p}_1 <_p \mathbf{p}_2$ ($\mathbf{v}_1 <_v \mathbf{v}_2$) iff for $i, j \in [1, \ldots, n]$ such that $p(i) = \mathbf{p}_1$ and $p(j) = \mathbf{p}_2$ it is the case that $\min\{i-1, n-i\} < \min\{j-1, n-j\}$. For arbitrary $p \in \mathfrak{S}_S$, the ordering of peaks and valleys does not really make sense, but in the case of atomic permutations, where the highest and lowest values are at the end of the permutation, it becomes useful. We use this ordering to generalize the concept of atomic permutation.

Example 9.2.1. *Given $p \in \mathfrak{S}_{[10]}$, a rising atomic permutation with two ascending runs and one descending run would be*

$$2468975311\underline{0}$$

,i.e. $2 < 4 < 6 < 8 < 9$*,* $9 > 7 > 5 > 3 > 1$*, and* $1 < \underline{10}$.

We will now give a generalized definition of atomic permutations, that is *m-atomic permutations* $\mathfrak{A}^{\pm}_S(m)$. The 1-atomic permutations are simply the atomic permutations $\mathfrak{A}^{\pm}_S(1) \equiv \mathfrak{A}^{\pm}_S$, when $m \geq 1$ every permutation $p \in \mathfrak{A}^{\pm}_S(m)$ must have $m-1$ peaks and $m-1$ valleys (this implies that $n \geq 2m$) and for $1 < i \leq m$, the i^{th} smallest value in S (s_{min_i}) must be the $(i-1)^{th}$ valley and the i^{th} largest value in S (s_{max_i}) must be the $(i-1)^{th}$ peak. Interesting enough, $\mathfrak{A}^{\pm}_S(m)$ has cardinality $2 \cdot (n-2m)!$ ($\mathfrak{A}^{+}_S(m)$ and $\mathfrak{A}^{-}_S(m)$ both have cardinality $(n-2m)!$). This might not be completely clear from what has been provided (after the equinumerosity proof this will become apparent, and quite possibly trivial), but consider this, if one fixes the two outer most positions to be the highest and lowest values, then one also fixes the location of the first two runs.

When discussing atomic permutations from now on, we will only consider the rising atomic permutations. Because of the innate symmetry in the definition what we prove for rising atomic permutations will also hold for the falling atomic permutations.

The counting question which will be the main focus of this work is, how many rising atomic permutations are there containing 3 runs of length n. The set of these rising atomic permutations will be abbreviated as $RAP_3(n)$. This question was considered in [33] and a enumeration, as well as a generating function were provided. Again, a slight error was made in this work as the provided enumeration counts either the rising or the falling atomic permutations not both. To count both, one needs to multiply by 2. We provide an alternative enumerating function by showing equinumerosity of $RAP_3(n+1)$ and $LRP_{\mathbf{a}}(n,3)$. This was done by finding the appropriate sub-program. The set $LRP_{\mathbf{a}}(n,3)$ is the set of all words constructed from the alphabet $\{a,b,c\}$ such that $\mathbf{a}$ occurs n times and, $\mathbf{b}$ and $\mathbf{c}$ occur at most n times; they can also occur zero times. We will refer to this problem as the *Letter Repetition Problem*. There are two generalizations of this problem, one $LRP_{\mathbf{a}}(n,m)$ where the alphabet is increased from three letters to m, also can be decreased, and $GLRP(n,m)$, *Generalized Letter Repetition Problem*, which removes the constraint that $\mathbf{a}$ occurs n times and allows any one of the m letters to occur $n+1$ times (but, the said symbol most occur once at the end of the string); this is just a multiplication by m. The set $GLRP(n,m)$ was derived from a generalization of a schematic formal proof constructed in the LK-calculus. While performing cut-elimination via the schematic CERES method [14,30], patterns emerged within a set of clauses derived from the proof with terms matching the members of $GLRP(n,m)$. The hope is that providing a proof of equinumerosity and/or bijections to certain types of permutations can aid the investigations into schematic proof theory as well as the theory of permutations .

Enumerating function for GLRP(n,3)

We will start this section with a couple of simple examples of the object we are enumerating. The problem $LRP_{\mathbf{a}}(n,3)$ specifically states, how many words w are there, built from an alphabet containing three symbols $\{a,b,c\}$, such that a occurs n times in w and each other symbol occurs at most n times. For example, given $n = 3$, the following words are allowed: aaa, $babaa$, *bbbcccaaa*, but *bbbbaaa* is not allowed. What is interesting about words in $LRP_{\mathbf{a}}(n,3)$ is that we can infer the structure of the members just by knowing which symbol is fixed, i.e. we know the value of n locations in the word. Also, we already know a minimum and maximum size of the words in $LRP_{\mathbf{a}}(n,3)$ which is n and $3n$, respectively. From this knowledge we can develop an enumeration of $LRP_{\mathbf{a}}(n,3)$. There is a slight generalization of this problem (as mentioned earlier) which directly occurs in a formal proof we are analysing. The generalization is, one symbol occurs $n+1$ times and the last position must be that symbol, the other symbols occur at most n times. This generalized problem $GLRP(n,3)$ is enumerated by $3 * |LRP_{\mathbf{a}}(n,3)|$. We provide an enumerating formula for the generalized problem which can be found on the OEIS as sequence A209245.

Theorem 9.2.1. *The number of words in the set $GLRP(n,3)$ is equivalent to the following formula:*

$$3 * \sum_{i=n}^{3*n} \binom{i}{n} \cdot \left(\sum_{\substack{x+y=(i-n) \\ 0 \le x,y \le n}} \binom{(i-n)}{x,y} \right)$$

Proof. Given an arbitrary word $w \in GLRP(n,3)$ the last symbol in the word defines which symbol must occur $n+1$ times. Thus, we know that $|GLRP(n,3)| = 3 * |LRP_{\mathbf{s}}(n,3)|$ (the last position is a ternary bit), where $s \in \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$. We know that the words in $GLRP(n,3)$ vary in size from $n+1$ to $3n+1$, thus, the words in $LRP_{\mathbf{a}}(n,3)$ must vary in size from n to $3n$. Knowing that n places in the words of $LRP_{\mathbf{a}}(n,3)$ are reserved for the symbol that was in the last place, we can extend our equation as follows:

$$|GLRP(n,3)| = 3 * |LRP_{\mathbf{a}}(n,3)| = 3 \cdot \sum_{i=n}^{3*n} \binom{i}{n} \cdot LRP'_{\mathbf{a}}(i-n, n, 3)$$

where $LRP'_{\mathbf{a}}(k,n,3)$ is the number of words of length k over an alphabet with two symbols where each symbol can occur at most n times. The value of k can be between 0 and $2n$. Given that a symbol can only fill n of the positions, This can be written simply as follows:

$$LRP'_{\mathbf{a}}(k,n,3) = \sum_{\substack{x+y=k \\ 0 \le x,y \le n}} \binom{(i-n)}{x,y}$$

And thus, we end with the following equation:

$$3 \cdot \sum_{i=n}^{3*n} \binom{i}{n} \cdot \left(\sum_{\substack{x+y=(i-n) \\ 0 \le x,y \le n}} \binom{(i-n)}{x,y} \right)$$

□

Corollary 9.2.1.

$$LRP_{\mathbf{a}}(n,3) =$$

$$2^n \cdot \binom{2\cdot n}{n} + \sum_{i=0}^{n-1} \left(2^i \cdot \binom{\sigma}{n} + 2 \cdot \binom{\sigma+n+1}{n} \cdot \left(2^\sigma - \sum_{j=0}^{i} \binom{\sigma+1}{j+n+1} \right) \right)$$

where $\sigma = i + n$.

Proof. By a lot of tedious manipulation of the binomial coefficients and summations. □

Starting with $n = 0$ we get the following integer sequence for $GLRP(n,3)$:

$$3, 33, 543, 10497, 220503, 4870401, 111243135, \cdots$$

Which is the main diagonal of a 3 dimensional recurrence relation Found on OEIS A209245. When we divide the formula by 3 (i.e. $LRP_{\mathbf{a}}(n,3)$), we get another integer sequence which is exactly the same sequence (for all numbers ≤ 20, the ones we have checked so far) as the formula derived by the authors of [33] for RAP3(n+1) (A241193) when $n \ge 0$:

$$1, 11, 181, 3499, 73501, 1623467, 37081045, \cdots$$

Their enumeration formula they derived is as follows:

$$RAP3(n) = \sum_{k=1}^{n} \left(\frac{(3n-(k+1))}{(2n-k)} \cdot \frac{(3n-(k+2))!}{(n-1)!(n-1)!(n-k)!} \right)$$

Thus, now we are left with the question, is $LRP_{\mathbf{a}}(n, 3) = RAP3(n + 1)$ for $n \geq 0$. We answer this question by showing equinumerosity of $LRP_{\mathbf{a}}(n, 3)$ and $RAP3(n + 1)$.

Equinumerosity of $LRP_{\mathbf{a}}(n, 3)$ and $RAP3(n + 1)$

Basic properties of $RAP3(n + 1)$

The first thing to consider is the length of a permutation which has 3 runs of equal size n.

Lemma 9.2.1. *An atomic permutation with 3 runs of equal length* n *must be constructed from a set* $\mathbf{S}$ *such that* $|\mathbf{S}| = 3n + 1$.

Proof. Let us assume that there exists $p \in \mathfrak{A}^{\pm}_{\mathbf{S}}$ with 3 runs of length n, we label the runs as r_1, r_2, and r_3 and assume they occur in this order. We know that if $l(r_1) = n$ then letting $A_1 \subset \mathbf{S}$ be the interval corresponding to r_1, it must be the case that $|A_1| = n + 1$. Thus, we need an interval with $n + 1$ elements to construct r_1. We also can deduce that the first position of the permutation must be part of r_1 because if it is not, than there is another run before r_1, this implies that the permutation has at least four runs contradicting its definition. We can also assume that the interval for r_2 contains the last element of the interval for r_1 because if this was not the case, between r_1 and r_2 would exists a run of length one. This, implies that $|A_1 \cup A_2| = 2n + 1$, where A_2 is the interval corresponding to r_2. The same argument that was used for the position between r_1 and r_2 holds for the position between r_2 and r_3, thus the size of all three intervals together is $|A_1 \cup A_2 \cup A_2| = 3n + 1$. If we were to add an additional position to the end of the permutation we'd either get an additional run or we would increase the length of the last run. If we were to subtract a position from the permutation we'd either decrease the length of some run or concatenate two runs. Thus, permutations of length $3n + 1$ are the only atomic permutations which can have 3 runs of equal length n. The same contradiction argument works for the case when we assume $|\mathbf{S}| = 3n$. $\square$

The fact that atomic permutations with 3 runs of equal length have a fixed size is of significant importance to the equinumerosity proof. However, at the same time it is quite odd given that the words we are showing equinumerosity with have variation in word size. We conjecture that this word size is equivalent to variation in the symbol positioning within a given permutations, however, our proof of equinumerosity is non-constructive in the sense that we do not provide a bijection but rather show that bijections between the two sets exists. We hope in future works computationally beneficial bijections can be constructed. Before we continue showing more of the basic properties of $RAP3(n + 1)$, we would like to point out an observation of ours that the structure of the runs can also be fixed if one only considers rising or falling atomic permutations, not both. We will only consider rising.

Lemma 9.2.2. *Given $p \in \mathfrak{A}^+_{\mathbf{S}}$ with 3 runs of equal length n, then p must have a run structure of $(ascending)(descending)(ascending)$.*

Proof. Obviously, given that the first position is the smallest value in the set $\mathbf{S}$ the first run must be ascending. This implies that the second run cannot be ascending because for that to be the case a run of length 1 must exist between the end of the first run and the beginning of the second run. Otherwise, the first run is not a true run because it would be extendible using the second run. The third run must be ascending because the last position is the largest value in $\mathbf{S}$. Thus, no other run structure is possible. $\square$

Corollary 9.2.2. *Given $p \in \mathfrak{A}^+_{\mathbf{S}}$ with 3 runs of equal length n it will always be the case that the $n + 1^{th}$ position will be a peak and the $(2n + 1)^{th}$ position will be a valley.*

Lemma 9.2.3. *Given $p \in \mathfrak{A}^+_{[3n+1]}$ with 3 runs of equal length n, then $2n+1 \leq p(n+1) < 3n+1$ and $1 < p(2n + 1) \leq n + 1$.*

Proof. Let us assume to the contrary that $n+1 < p(n+1) < 2n+1$, and $1 < p(2n+1) \leq n+1$. We know the interval $[n + 1, 2n + 1]$ is descending, thus, allowing the value in the position $p(2n+1) < 2n+1$ forces the second run $|r_2| \leq n-1$ and the last run $r_3 \geq n+1$ which contradicts what we would like to be the case. If we instead allow, to the contrary, $2n+1 > p(2n+1) > n+1$ and $2n + 1 \leq p(n + 1) < 3n + 1$. We force the run $|r_2| \leq n - 1$ again, but this time the run $|r_1| \geq n + 1$. When we allow both of the positions be contrary to the rule, we allow further shrinking of the run r_2 and/or further expansion of the runs r_1 and r_3. $\square$

Theorem 9.2.2. *Given $p \in \mathfrak{A}^+_{[3n+1]}$ with 3 runs of equal length n, then there is only one permutation such that $p(n + 1) = 2n + 1$ and $p(2n + 1) = n + 1$.*

Proof. As in Lem.9.2.3, we will consider the individual intervals over which the runs are defined. Let us consider the first interval $[1, \cdots , n + 1]$, we know that $p(1) = 1$ and $p(n + 1) = 2n + 1$. Also, we know that $p(n + 1)$ is a peak and that the first run, r_1, is be ascending. Given that $p(2n + 1)$ is a valley and assigned the value $n + 1$, there are exactly $n - 1$ value which cannot be assigned to the runs r_2 and r_3 which must be assigned to r_1. Thus the first run must be $1, 2, \cdots , n - 1, n, 2n + 1$. How let us consider run r_3, there are exactly $n - 1$ values which are greater than $2n + 1$ and thus cannot be part of the run r_2, these values are $2n + 2, \cdots , 3n$. Thus, we now know the only possible run for r_3 which is $n + 1, 2n + 2, \cdots , 3n, 3n + 1$. By inertia, we now know the only possible run for r_2. $\square$

Now that we have outlined the basic properties of $p \in \mathfrak{A}^+_{[3n+1]}$ that are needed for the proof of equinumerosity, we can prove the necessary parts of the construction, as well as, the sub-problems which are at the heart of the proof of equinumerosity.

Definition 9.2.1. *Let $RAP3(n)(i, j)$ for $0 \leq i, j < n$ be the same as $RAP3(n)$ except we fix the positions $p(n + 1) = 2n + 1 + i$ and $p(2n + 1) = n + 1 - j$.*

Corollary 9.2.3. $RAP3(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} RAP3(n)(i, j)$

Theorem 9.2.3. *For $0 \leq i \leq n$, $|RAP3(n + 1)(i, 0)| = |RAP3(n + 1)(0, i)| = \binom{n+i}{i}$.*

Proof. We prove for i and leave j as an exercise, the problems are symmetric. When $i = 0$, we know by Lem. 9.2.2 that there is only one $p \in \mathfrak{A}^{+}_{[3n+4]}$ which fits the constraints thus, $|RAP3(n + 2)(0, 0)| = \binom{n+0}{0} = 1$. Let us assume the theorem holds for all $i \leq m$ let us now show for $m + 1$. Let us now investigate what happens when we take a permutation $p \in RAP3(n+1)(m, 0)$ and turn it into a permutation $p' \in RAP3(n+2)(m+1, 0)$. We know that $p(n + 2) = 2 * n + 3 + m$ and that there is only one location where $2 * n + 4 + m$ can be located, namely, $p(2 * n + 4 + m)$. The reason $p(2 * n + 4 + m) = 2 * n + 4 + m$ is that the value is greater than both the peak and valley so it must be in r_3 and if it where in a position lower than $2 * n + 4 + m$ there would have to be a value smaller than $2 * n + 4 + m$ in position $2 * n + 4 + m$ which would break the run structure.

Now let us consider switching the value at $p(n+2)$ with the value at $p(2*n+4+m)$, for the resulting permutation p' the following holds, $p(n + 2) = 2 * n + 4 + m$ and $p(2 * n + 4 + m) = 2*n+3+m$. Now we can ask, is $2*n+4+m$ the only position where we can put $2*n+3+m$? In turns out there is one other location where $2 * n + 3 + m$ can be placed Which is $n + 3$. We will call the permutation where the position $n + 3$ is $2 * n + 3 + m$, p''. We can consider the number of permutations which have the constraints we placed on p' to be the same as the number of permutations which have the constraints of p because we only switched two values. In the case of p'', we had to switch 3 values to get the configuration defined above, but in the end the number of position which are not fixed is the same, $n + m$. For the p' configuration, we use the induction hypothesis that $|RAP3(n + 2)(m, 0)| = \binom{n+m}{m}$, which implies that there are $n + m$ places to choose from. For the p'' configuration, we know that run r_2 has n places which are not fixed and run r_3 has $m + 1$ places which are not fixed. Which gives us $\binom{n+m}{m+1}$ possible permutations. If we add the two statements together we get the following:

$$\binom{n+m}{m} + \binom{n+m}{m+1} = \binom{n+m+1}{m+1}$$

Which is the size of $RAP3(n + 1)(m + 1, 0)$ by the induction hypothesis. Using this same argument one can prove $RAP3(n + 1)(0, m + 1) = \binom{n+m+1}{m+1}$ as well. □

Theorem 9.2.4. *For* $0 \leq i, j \leq n$, $|RAP3(n + 1)(i, j)| = \binom{n+i+j}{j} \cdot \binom{n+i}{i}$.

Proof. We will prove this theorem by induction using Thm. 9.2.3 as a basecase for $j = 0$. We now assume the theorem holds for $k \leq j$ and show the theorem holds for $j + 1$. Using a similar argument as the one used in Thm. 9.2.3, we know that $p(2n + 3) = n + 1 - j$ and that $p(n + 1 - j) = n + 2 - j$ is a possible configuration of a permutation in the RAP3(n+1)(i,j+1) case. However the problem , unlike in the previous case, allows for two other possibilities for the placement of $n + 2 - j$, i.e. $p(2n + 4)$ and $p(2n + 2)$. When we place $n + 2 - j$ in the position $p(n + 1 - j)$ we have the induction hypothesis $\binom{n+i+j}{j} \cdot \binom{n+i}{i}$. We can also derive this using the following steps:

$$\binom{n+i+j+1}{j+1} \cdot \binom{n+i}{i} = \binom{n+i+j}{j+1} \cdot \binom{n+i}{i} + \binom{n+i+j}{j} \cdot \binom{n+i}{i} =$$

$$\binom{n+i+j}{j+1}\cdot\binom{n+i-1}{i}+\binom{n+i+j}{j+1}\cdot\binom{n+i-1}{i-1}+\binom{n+i+j}{j}\cdot\binom{n+i}{i}$$

What we need now is a combinatorial interpretation of the first two statements in the equation. We can interpret $\binom{n+i+j}{j+1}\cdot\binom{n+i-1}{i}$ as setting $p(2n+2) = n+2-j$ Because we need to still fix $j+1$ positions in the first run out of the $n+i+j$ elements available and out of the remaining $n+i-1$ elements we need to fix i of them. We can interpret $\binom{n+i+j}{j+1}\cdot\binom{n+i-1}{i-1}$ as $p(2n+4) = n+2-j$ because again we need to choose $j+1$ positions in the first run out of the $n+i+j$ but this time there are only $i-1$ positions that need to be chosen in the third run. Thus, we have completed the induction step and the theorem is proven. □

Corollary 9.2.4. $|RAP3(n+1)| = \sum_{i=0}^{n}\sum_{j=0}^{n}\binom{n+i+j}{j}\cdot\binom{n+i}{i}$

Basic properties of $LRP_{\mathbf{a}}(n,3)$

In this section we list some properties of $LRP_{\mathbf{a}}(n,3)$ which will help with constructing the function used to prove equinumerosity.

Definition 9.2.2. *Given a finite alphabet Σ, let $W(\Sigma)$ be the words constructable from Σ. If $w \in W(\Sigma)$ and $\mathrm{s} \in \Sigma$, then the* concatenation *of* s *and* **w** *is also in* $\mathrm{s}w \in W(\Sigma)$. *It also holds that if* $\mathrm{s}w \in W(\Sigma)$, *then* $w \in W(\Sigma)$.

Definition 9.2.3. *Let $LRP_{\mathbf{a}}(n,3)(i,j)$ for $0 \le i,j \le n$ be the same as $LRP_{\mathbf{a}}(n,3)$ except we fix the number of times that* b *and* c *occur to i and j, respectively.*

Corollary 9.2.5. $LRP_{\mathbf{a}}(n,3) = \sum_{i=0}^{n}\sum_{j=0}^{n} LRP_{\mathbf{a}}(n,3)(i,j)$

Lemma 9.2.4. $\mathbf{a}w \in LRP_{\mathbf{a}}(n+1,3)(i,j)$ *where $i,j \le n$ and* **a** *occurs exactly $n+1$ times iff $w \in LRP_{\mathbf{a}}(n,3)(i,j)$.*

Proof. We know that $\mathbf{a}w$ has $n+1$ occurrences of **a**, thus, w has only n occurrences of **a**. Then the removal of the leading **a** from $\mathbf{a}w$ leaves w with the same structure as $\mathbf{a}w$, which is in $LRP_{\mathbf{a}}(n+1,3)(i,j)$. Thus, we see that $w \in LRP_{\mathbf{a}}(n,3)(i,j)$. □

Lemma 9.2.5. $\mathbf{b}w \in LRP_{\mathbf{a}}(n,3)(i+1,j)$ *where* **b** *occurs $i+1$ times iff $w \in LRP_{\mathbf{a}}(n,3)(i,j)$.*

Proof. We know that $\mathbf{b}w$ has $i+1$ occurrences of **b**, thus, w has only i occurrences of **b**. Then the removal of the leading **b** from $\mathbf{b}w$ leaves w with the same structure as $\mathbf{b}w$, which is in $LRP_{\mathbf{a}}(n+1,3)(i,j)$. Thus, we see that $w \in LRP_{\mathbf{a}}(n,3)(i,j)$. □

Corollary 9.2.6. $\mathbf{c}w \in LRP_{\mathbf{a}}(n,3)(i,j+1)$ *where* **c** *occurs $j+1$ times iff $w \in LRP_{\mathbf{a}}(n,3)(i,j)$.*

Lemma 9.2.6. $|LRP_{\mathbf{a}}(n,3)(n,0)| = 2 * |LRP_{\mathbf{a}}(n,3)(n-1,0)|$.

Proof. any word $w \in LRP_{\mathbf{a}}(n,3)(n,0)$ contains n occurrences of the symbols $\mathbf{a}$ and $\mathbf{b}$. Let us consider only the words in $LRP_{\mathbf{a}}(n,3)(n,0)$ which start with $\mathbf{b}$ which we will refer to as $W_b(n,n)$. By Lem. 9.2.5, $\mathbf{b}w \in W_b(n,n)$ iff $w \in LRP_{\mathbf{a}}(n,3)(n-1,0)$. We now consider the words in $LRP_{\mathbf{a}}(n,3)(n,0)$ which start with $\mathbf{a}$, which we will refer to as $W_a(n,n)$. If we treat this case exactly the same as the $\mathbf{b}$ case, we would again get $\mathbf{a}w \in W_a(n,n)$ iff $w \in LRP_{\mathbf{b}}(n,3)(n-1,0)$. For this to be the case we can use the following function f on words $w \in W_a(n,n)$:

$$f(\mathbf{a}w) \to \mathbf{b}f(w)$$
$$f(\mathbf{b}w) \to \mathbf{a}f(w)$$
$$f(\_) \to \_$$

Where _ is the empty word. Thus we have $\mathbf{a}w \in W_a(n,n)$ iff $f(w) \in LRP_{\mathbf{b}}(n,3)(n-1,0)$. putting the two parts together we get that

$$|LRP_{\mathbf{a}}(n,3)(n,0)| = |W_a(n,n)| + |W_b(n,n)| = |LRP_{\mathbf{a}}(n,3)(n-1,0)| + |LRP_{\mathbf{b}}(n,3)(n-1,0)|$$

The two sets $LRP_{\mathbf{a}}(n,3)(n-1,0)$ and $LRP_{\mathbf{b}}(n,3)(n-1,0)$ are symmetric and thus, have the same size. Just we can conclude that

$$|LRP_{\mathbf{a}}(n,3)(n-1,0)| + |LRP_{\mathbf{b}}(n,3)(n-1,0)| = 2 * |LRP_{\mathbf{a}}(n,3)(n-1,0)|$$

□

Corollary 9.2.7. $|LRP_{\mathbf{a}}(n,3)(0,n)| = 2 * |LRP_{\mathbf{a}}(n,3)(0,n-1)|$

Now that we have shown many of the important properties of the interlacing structure of $LRP_{\mathbf{a}}(n,3)(i,j)$, we can use this structure to create a recursive enumeration function . This recursive enumeration function is at the heart of the proof of equinumerosity.

Theorem 9.2.5. *For $n \geq m \geq 0$ and $k \geq 1$, $|LRP_{\mathbf{a}}(n,3)(m,0)| = |LRP_{\mathbf{a}}(n,3)(0,m)| = c(n,m)$ where $c(\cdot,\cdot)$ is defined as follows:*

$c(n, n+k) \longrightarrow 0$
$c(n, 0) \longrightarrow 1$
$c(m+1, m+1) \longrightarrow 2 \cdot c(m+1, m)$
$c(n+1, m+1) \longrightarrow c(n+1, m) + c(n, m+1)$

Proof. We begin with the statement that $c(n,0) \longrightarrow 1$, Consider this as a string only containing one symbol. For any value of n there is only one word containing a single symbol in $LRP_{\mathbf{a}}(n,3)(0,0)$. When we have $c(n,n+k)$, this violates the constraint that $n \geq m$, thus we derive that every set $LRP_{\mathbf{a}}(n,3)(n+k,0) \equiv \emptyset$. The case $c(m+1,m+1) \longrightarrow 2 \cdot c(m+1,m)$ is handled by Lem. 9.2.6. And finally, the case $c(n+1,m+1) \longrightarrow c(n+1,m) + c(n,m+1)$ is handle by Lem. 9.2.4 & 9.2.5. □

Theorem 9.2.6. *for* $0 \leq i, j \leq n$*, it holds that* $|LRP_{\mathbf{a}}(n,3)(i,j)| = c(n,i) * c(n+i,j)$

Proof. When $j = 0$ we have by Thm. 9.2.5, that

$$|LRP_{\mathbf{a}}(n,3)(i,0)| = c(n,i) * c(n+i,0) = c(n,i)$$

which proves one basecase. When $i = 0$ we have by Thm. 9.2.5, that

$$|LRP_{\mathbf{a}}(n,3)(0,j)| = c(n,0) * c(n,j) = c(n,j)$$

which proves the second basecase . We now construct two induction hypothesises, the first being for all i and for all $k < j$ the theorem holds and the second being for all j and for all $w < i$ holds. We now show that it holds for j and i, respectively.

Before we finish proving the theorem we need to define a few things. Let us assign letters to each of the variables n, i and j, namely $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$, respectively. We know that all words found in $LRP_{\mathbf{a}}(n,3)(i,j)$ must start with one of the three letters. Thus we can rewrite $LRP_{\mathbf{a}}(n,3)(i,j)$ as $W_{\mathbf{a}}(n,i,j) \cup W_{\mathbf{b}}(n,i,j) \cup W_{\mathbf{c}}(n,i,j)$ where $W_{\mathbf{a}}$ are the words which start with $\mathbf{a}$ and so on. We are now working with the equation:

$$|W_{\mathbf{a}}(n,i,j) \cup W_{\mathbf{b}}(n,i,j) \cup W_{\mathbf{c}}(n,i,j)| = c(n,i) * c(n+i,j) \tag{9.1}$$

First we will unroll the $c(\cdot,\cdot)$s and derive the following:

$$c(n,i) * c(n+i,j) = c(n,i) * c(n+(i-1),j) + c(n,i) * c(n+i,j-1) \tag{9.2}$$

which implies by Eq. 9.1,

$$\begin{gathered}|W_{\mathbf{a}}(n,i,j) \cup W_{\mathbf{b}}(n,i,j) \cup W_{\mathbf{c}}(n,i,j)| = \\ c(n,i) * c(n+(i-1),j) + c(n,i) * c(n+i,j-1)\end{gathered} \tag{9.3}$$

In Eq. 9.3 the term $c(n,i) * c(n+i,j-1)$ holds by the first induction hypothesis and is equivalent to $W_{\mathbf{c}}(n,i,j)$. Thus, we derive the following:

$$|W_{\mathbf{a}}(n,i,j) \cup W_{\mathbf{b}}(n,i,j)| = c(n,i) * c(n+(i-1),j) \tag{9.4}$$

If we unroll the $c(n,i)$ term this time we get the following:

$$\begin{gathered}|W_{\mathbf{a}}(n,i,j) \cup W_{\mathbf{b}}(n,i,j)| = \\ c(n-1,i) * c(n-1+i,j) + c(n,i-1) * c(n+(i-1),j)\end{gathered} \tag{9.5}$$

As we see the right part of Eq. 9.5 holds by the second induction hypothesis and is equivalent to $W_{\mathbf{b}}(n,i,j)$ resulting in the following:

$$\begin{gathered}|W_{\mathbf{a}}(n,i,j)| = \\ c(n-1,i) * c(n-1+i,j)\end{gathered} \tag{9.6}$$

In Eq. 9.6, it is obvious that $W_{\mathbf{a}}(n,i,j)$ is equivalent to $c(n-1,i) * c(n-1+i,j)$ because we have reduced the value of n. Thus, the theorem holds. □

Proof of Equinumerosity

Theorem 9.2.7. *for $n > m \geq 0$, $c(n,m) = \binom{n+m}{n}$*

Proof. For the basecase we see that $c(n,0) = 1 = \binom{n+0}{n} = 1$. Thus, the theorem holds for the basecase. We assume as the induction hypothesis that the theorem holds for all $n \geq 0$ and $k < m$ and we show that it holds for m. First let us unroll the $c(n,m)$ term into the needed form:

$$c(n,m) = c(n,m-1) + c(n-1,m)$$

Now we unroll the statement $\binom{n+m}{n}$:

$$\binom{n+m}{n} = \binom{n+m-1}{n} + \binom{n+m-1}{n-1}$$

equating the two derivations we get the following:

$$\binom{n+m-1}{n} + \binom{n+m-1}{n-1} = c(n,m-1) + c(n-1,m)$$

By the induction hypothesis we get:

$$\binom{n+m-1}{n} + \binom{n+m-1}{n-1} = \binom{n+m-1}{n} + c(n-1,m)$$

and thus,

$$\binom{(n-1)+m}{n-1} = c(n-1,m) \tag{9.7}$$

To finish this proof we need to do a second induction on the difference between n and m in $c(n,m)$. For the basecase let us assume the difference is zero then we would have the following using Eq. 9.7:

$$\binom{2 \cdot m}{m} = c(m,m) \tag{9.8}$$

By the definition of $c(\cdot,\cdot)$ Eq. 9.8 implies that the following holds:

$$\binom{2 \cdot m}{m} = c(m,m-1) + c(m,m-1) \tag{9.9}$$

By the initial induction hypothesis we get that the following holds:

$$\binom{2 \cdot m}{m} = \binom{2 \cdot m - 1}{m} + \binom{2 \cdot m - 1}{m} = \binom{2 \cdot m - 1}{m} + \binom{2 \cdot m - 1}{m-1} = \binom{2m}{m} \tag{9.10}$$

Thus Eq. 9.10 proofs the basecase. For the induction hypothesis we assume that Eq. 9.7 holds for all gaps k, s.t. $0 \leq k < g < n$, we show for a gap of size g. This implies the following equation.

$$\binom{2\cdot m+g}{m+g} = c(m+g,m) \tag{9.11}$$

By the rules of $c(\cdot,\cdot)$ Eq. 9.11 implies the following:

$$\binom{2\cdot m+g}{m+g} = c(m+g-1,m) + c(m+g,m-1) \tag{9.12}$$

By the initial induction hypothesis we get:

$$\binom{2\cdot m+g}{m+g} = c(m+g-1,m) + \binom{m+g+m-1}{m+g} \tag{9.13}$$

And by the second induction hypothesis we get:

$$\binom{2\cdot m+g}{m+g} = \binom{m+g+m-1}{m+g-1} + \binom{m+g+m-1}{m+g} = \binom{2\cdot m+g}{m+g} \tag{9.14}$$

Thus, by Eq. 9.14 we have proven the second induction holds and in turn have proven the first induction as well. Thus, the theorem holds. □

Corollary 9.2.8. *for $n \geq 0$, $c(n,n) = \binom{2\cdot n}{n}$*

Proof. For the basecase we see that $c(0,0) = 1 = \binom{0}{0} = 1$. Thus, the theorem holds for the basecase. We assume as the induction hypothesis that the theorem holds for $m < n$ and we show that it holds for n. First let us unroll the $c(n,n)$ term into the needed form:

$$c(n,n) = c(n,n-1) + c(n,n-1) = 2 * c(n-1,n-1) + 2 * c(n,n-2)$$

We can also unroll the $\binom{2\cdot n}{n}$ term as follows:

$$\binom{2\cdot n}{n} = \binom{2\cdot n-1}{n} + \binom{2\cdot n-1}{n-1} =$$

$$\binom{2\cdot n-2}{n-2} + \binom{2\cdot n-2}{n-1} + \binom{2\cdot n-2}{n-1} + \binom{2\cdot n-2}{n-2} =$$

$$2 * \binom{2\cdot n-2}{n-1} + 2 * \binom{2\cdot n-2}{n-2} = 2 * c(n-1,n-1) + 2 * c(n,n-2)$$

Which by the induction hypothesis implies the following:

$$2 * \binom{2\cdot n-2}{n-1} + 2 * \binom{2\cdot n-2}{n-2} = 2 * \binom{2\cdot n-2}{n-1} + 2 * c(n,n-2)$$

$$2 * \binom{2\cdot n-2}{n-2} = 2 * c(n,n-2)$$

By Thm. 9.2.7 we know the value of $c(n,n-2)$ and thus we can derive the following:

$$2 * \binom{2\cdot n-2}{n-2} = 2 * \binom{2\cdot n-2}{n}$$

Thus proving the theorem. $\square$

Theorem 9.2.8. *for* $n \geq 0$ *and* $k \geq 0$, $c(n+k,n) = \binom{2\cdot n+k}{n} = \binom{2\cdot n+k}{n+k}$

Proof. The induction basecase is provided by Cor. 9.2.8. We assume that the theorem holds for $m < k$, we show for k.

$$\binom{2n+k}{n} = \binom{2n+(k-1)}{n} + \binom{2(n-1)+k}{n-1} =$$

$$\binom{2n+(k-1)}{n} + \frac{n}{(n-1)+k}\cdot\binom{2n+(k-1)}{n}$$

$$\left(1+\frac{n}{(n-1)+k}\right)\cdot\binom{2n+(k-1)}{n} = \left(\frac{2n+(k-1)}{n+(k-1)}\right)\cdot\binom{2n+(k-1)}{n} =$$

$$\left(\frac{2n+(k-1)}{n+(k-1)}\right)\cdot c(n+k-1,n) \rightarrow$$

$$\binom{2n+k}{n} = \left(\frac{2n+k-1}{n+k-1}\right)\cdot c(n+k-1,n) \rightarrow$$

$$\left(\frac{n+k-1}{2n+k-1}\right)\cdot\binom{2n+k}{n} = c(n+k-1,n) \rightarrow$$

$$\binom{2n+(k-1)}{n} = c(n+k-1,n)$$

Thus it follows:

$$\binom{2n+k}{n} = c(n+k,n)$$

proving the theorem.

$\square$

What might not be directly apparent from the above result of Thm. 9.2.8 is that this recursion allows us to build Pascal's triangle columnwise rather then row or diagonalwise . We see from Cor. 9.2.8 that the central binomial coefficients (A000984) are built using $c(n,n)$. Using $c(n+1,n)$ we produce the sequence (A001700) which are the two columns adjacent to the central binomial coefficients. For every value of $k \geq 0$ we get the two symmetric columns distance k from the central binomial coefficients . $k=0$ is the special case when we get only one column, the central binomial coefficients themselves. We can define a new function called *Pascal Column Counter* $PCC(n,k) = c(n+k,n)$ which returns the n^{th} value of the k^{th} column of pascal's triangle.

Theorem 9.2.9 (Equinumerosity of $RAP3(n+1)(i,0)$ & $LRP_{\mathbf{a}}(n,3)(i,0)$)**.** *For* $0 \leq i \leq n$, $|RAP3(n+1)(i,0)| = |LRP(n,3)(i,0)|$

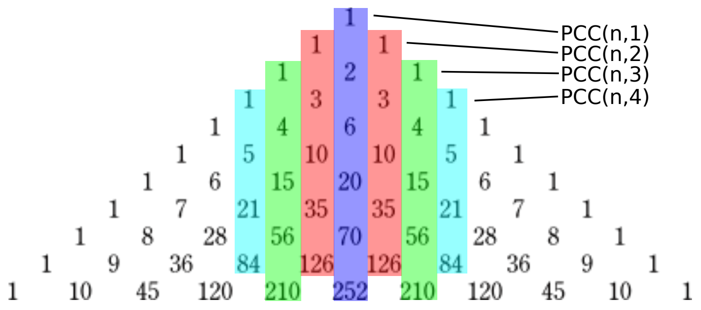


Figure 9.1: Example of the first four instances of the Pascal Column Counter.

Proof. By Thm. 9.2.7, Cor. 9.2.8, & Thm. 9.2.3, we know that the two sets have the same closed form formula, thus, they must be equinumerous. □

Lemma 9.2.7. *For* $0 \leq i, j \leq n$,

$$c(n,i) \cdot c(n+i,j) = \binom{n+i}{i} \cdot \binom{n+i+j}{j}$$

Proof. By Thm. 9.2.7, Cor. 9.2.8. □

Theorem 9.2.10 (Equinumerosity Theorem)**.** *For* $0 \leq i, j \leq n$, $|RAP3(n+1)(i,j)| = |LRP_{\mathbf{a}}(n,3)(i,j)|$

Proof. By Thm. 9.2.4 & Lem 9.2.7. □

Corollary 9.2.9. *For* $0 \leq n$, *there exist functions* f_n *and* g_n *such that for* $p \in RAP3(n+1)$ *then* $f_n(p) \in LRP_{\mathbf{a}}(n,3)$ *and for* $w \in LRP_{\mathbf{a}}(n,3)$ *then* $g_n(p) \in RAP3(n+1)$

As shown by 9.2.9, we do not have a method for construction a bijection between the two set, we only know that they exist. Thus, an open problem is to find and construct such a bijection.

Additional Results

We can now derive another way of computing both sets.

Corollary 9.2.10. $|LRP_{\mathbf{a}}(n,3)| = |RAP3(n+1)| = \sum_{i=0}^{n}\sum_{j=0}^{n}\binom{n+i+j}{j}\cdot\binom{n+i}{i} = \sum_{i=0}^{n}\binom{n+i}{i}\cdot\sum_{j=0}^{n}\binom{n+i+j}{j} = \sum_{i=0}^{n}\binom{n+i}{i}\cdot\binom{2n+i+1}{n}$

A very nice result of this work is that, not only did we show that the enumeration of the set rising atomic permutations with 3 runs of length n is equivalent to the enumeration of our word problem, we also defined the equivalence of an assortment of sub-problems based on the value of the peak and valley. We now construct the *peak and valley* matrix using $RAP3(n+1)(i,j)$ as defined earlier. The following is a peak and valley matrix for $RAP3(1)(i,j)$ through $RAP3(4)(i,j)$:

$$RAP3(1)(i,j) = \begin{pmatrix} 1 \end{pmatrix}$$

$$RAP3(2)(i,j) = \begin{pmatrix} 1 & 2 \\ 2 & 6 \end{pmatrix}$$

$$RAP3(3)(i,j) = \begin{pmatrix} 1 & 3 & 6 \\ 3 & 12 & 30 \\ 6 & 30 & 90 \end{pmatrix}$$

$$RAP3(4)(i,j) = \begin{pmatrix} 1 & 4 & 10 & 20 \\ 4 & 20 & 60 & 140 \\ 10 & 60 & 210 & 560 \\ 20 & 140 & 560 & 1680 \end{pmatrix}$$

Notice the matrix is symmetric along the diagonal. What we can do is define a new problem, but before we do so, let us define the *primitive peak* and *primitive valley* . The primitive peak and valley (p_p and p_v) are the values, which when assigned to the peak and valley position allow for only one rising atomic permutation with 3 runs of length n. Essentially, p_p and p_v are the peak and valley of $RAP3(4)(0,0)$. The new enumeration problem is the number of rising atomic permutations p with 3 runs of length n with peak ρ and valley ν such that the following holds $|\rho - p_p| = |p_v - \nu|$, $ERAP3(n)$. For example, $1|3|6|4|2|5|7$ is an example of such a permutation (i.e. $|6-5| = |3-2|$), but $1|2|6|4|3|5|7$ is not ($|6-5| \neq |3-3|$). Such permutations would be enumerated by the formula:

$$ERAP3(n+1) = \sum_{i=0}^{n} \binom{n+2i}{i} \cdot \binom{n+i}{i} = \sum_{i=0}^{n} \binom{n+2i}{i,n,i}$$

and results in the following sequence:

$$1, 7, 103, 1911, 39301, 856885, 19401159, 451019191, \cdots$$

which is not found on the OEIS . This is not the only sequence which can be derived from the peak and valley matrix. A more interesting sequence found in the peak and valley matrix is the sequence derived from $RAP3(n+1)(n,n)$, namely the 2-atomic permutations with 3 runs of equal length n, which happens to be:

$$\binom{3n}{n,n,n}$$

Sequence A006480 in the OEIS. This sequence eludes to one more interesting property of the peak and valley matrix, that is, it is a planar cut of pascal's pyramid starting at one of the 1 positions in the $n-1^{th}$ row and ending at the central position in the row $2n$. The following diagram provides the first Six rows of pascal's pyramid with the cut across the pyramid made by $n = 3$ Marked in red:

$\begin{array}{l} 1 \end{array}$	$\begin{array}{ll} 1 & \\ 1 & 1 \end{array}$	$\begin{array}{lll} 1 & & \\ 2 & 2 & \\ 1 & 2 & 1 \end{array}$
$\begin{array}{llll} 1 & & & \\ 3 & 3 & & \\ 3 & 6 & 3 & \\ 1 & 3 & 3 & 1 \end{array}$	$\begin{array}{lllll} 1 & & & & \\ 4 & 4 & & & \\ 6 & 12 & 6 & & \\ 4 & 12 & 12 & 4 & \\ 1 & 4 & 6 & 4 & 1 \end{array}$	$\begin{array}{llllll} 1 & & & & & \\ 5 & 5 & & & & \\ 10 & 20 & 10 & & & \\ 10 & 30 & 30 & 10 & & \\ 5 & 20 & 30 & 20 & 5 & \\ 1 & 5 & 10 & 10 & 5 & 1 \end{array}$
	$\begin{array}{lllllll} 1 & & & & & & \\ 6 & 6 & & & & & \\ 15 & 30 & 15 & & & & \\ 20 & 60 & 60 & 20 & & & \\ 15 & 60 & 90 & 60 & 15 & & \\ 6 & 30 & 60 & 60 & 30 & 6 & \\ 1 & 6 & 15 & 20 & 15 & 6 & 1 \end{array}$	

The other cuts made by the peak and valley matrix are easy to find. Also, an interesting fact is given that Pascal's Pyramid is a graphic illustration of the coefficients of the formula $(a+b+c)^n$ we can consider which combinations of a, b and c are in the peak and valley matrix. It turns out that if one chooses to start the plane from the corner assigned the polynomial a^{n-1} every coefficient in the peak and valley matrix will have a polynomial part containing a^{n-1}.

There is one last question we have which we would like to answer. In the case of $GLRP(n, 3)$, we get a nice natural interpretation of the 3 symmetries found in the pascal pyramid when considering the peaks and valley matrix's planar cut. However, in the case of atomic permutations, we only have two symmetries. Is there are third symmetry for RAP3(n+1) to account for the three symmetries found in the pyramid? Such a discovery can aid the construction of a bijection. Also, how can one generalize these results to other subsets of the atomic permutations and permutations in general.

9.3 Conclusion

The majority of the work outlined in the Sec. 9.2 is at a tangent to the original goal of this section, that is namely to perform an analysis of the clause set for the gNiA-schema . As mentioned in the introduction we would like to find lemmata for the gNiA-schema similar to the Lemmata found in Ch. 6 for the NiA-schema. The problem is that finding such lemmata is extremely difficult for this generalization. However, the gNiA-schema was derived from the NiA-schema, and thus, we can assume (of course not formally) that an analogous series of lemmata exist for the$g$NiA-schema, more specifically, that a derived clause set exists with a similar combinatorial structure as that of the NiA-schema. Empirical evidence which can be found in the following chapter suggest this to be true. Following this assumption we proceeded to construct a combinatorial interpretation based on the NiA-schema results and attempt to count the expected number of clauses needed in the derived clauses which will lead to a refutation. This is exactly what we do in Sec. 9.2 with the addition step of showing a second combinatorial interpretation for the derived clause set.

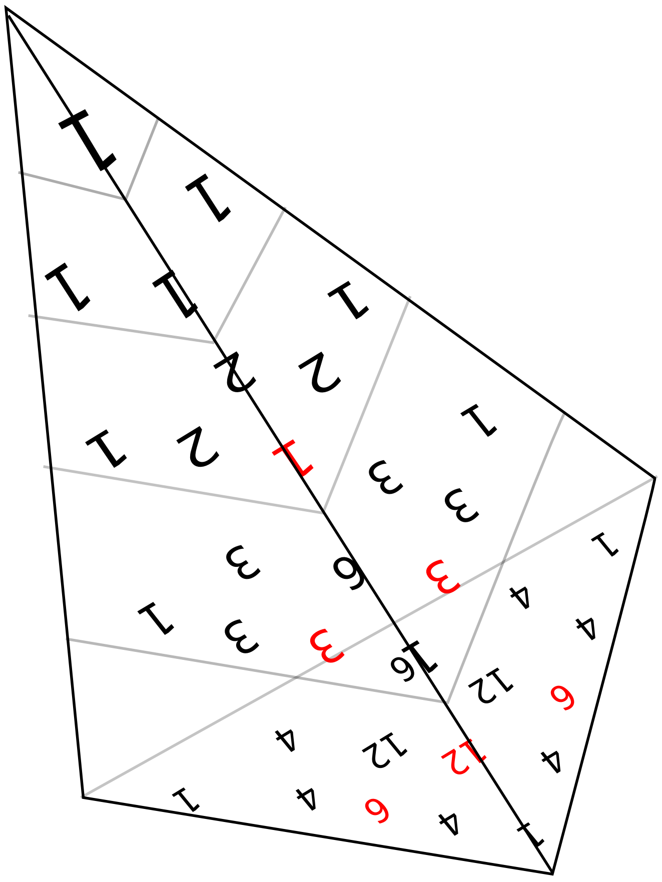


Figure 9.2: A 3-D representation of Pascal's Pyramid up to layer 4. The planar cut made by the peaks and valleys matrix is marked in red. The slight angle of the planar cut towards the central elements can be seen in this image.

What we where able to construct was not only a enumeration of the expected combinatorial structure of the derived clause set, but also a recursion for constructing this structure, an essential part of schematic CERES. This recursion was constructed in Thm. 9.2.5. Not only was this recursion constructed but an interesting and unexpected result was discovered, namely that the recursion necessary for the full combinatorial interpretation of the derived clause set required a multiplication of two recursive functions. This multiplication mimics the continuous splitting we see in the SPASS proof which will be discussed in the following chapter. The hope is that given the recursion constructing this interpretation can be backwards engineered to construct a refutation of the original clause set for the gNiA-schema. It is also beneficial that we know approximately the size of the derived clause set. More of these issues will be discussed in the next chapter.

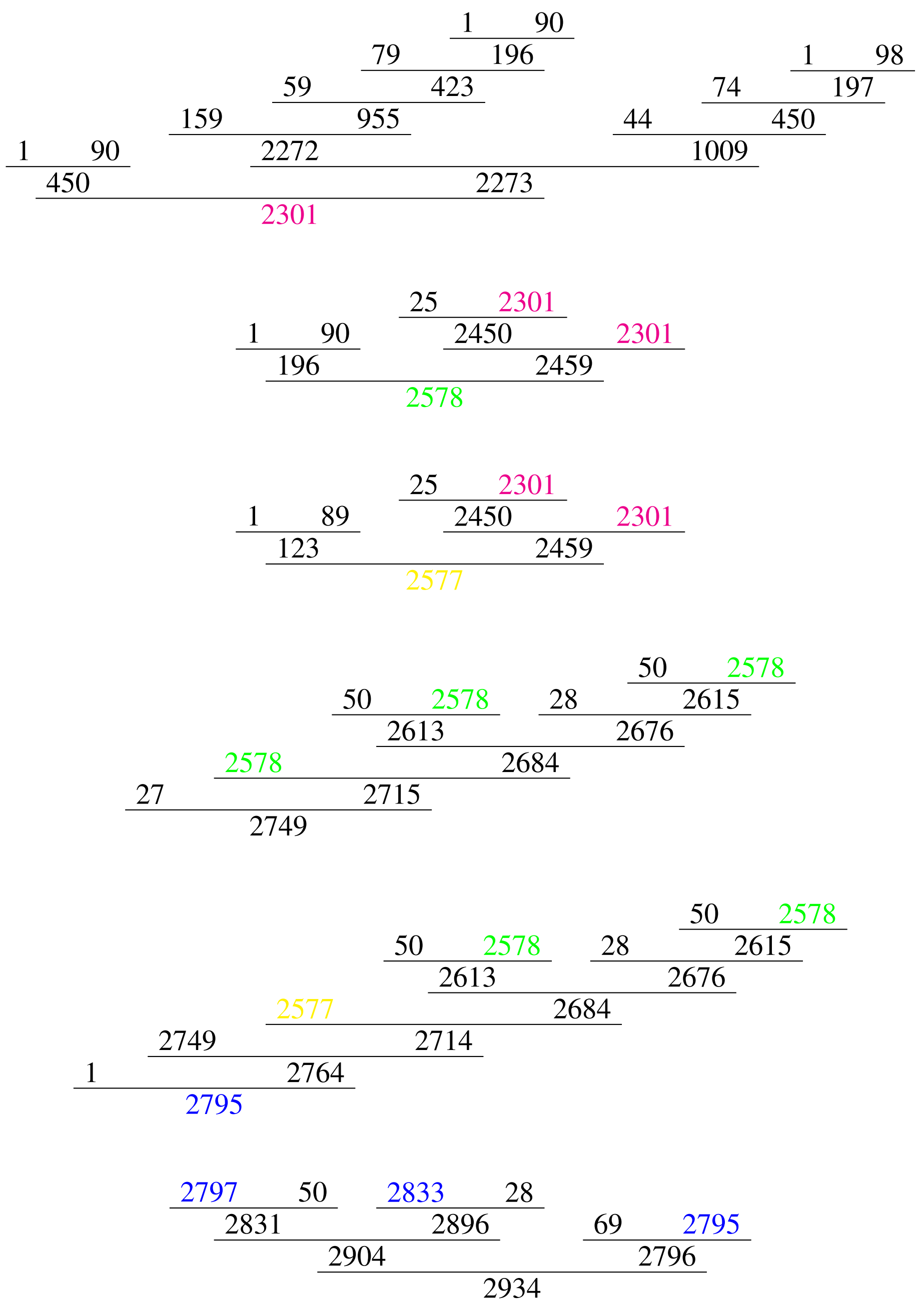
CHAPTER 10

Developing Schematic Refutations Using Theorem Provers

10.1 Theorem Prover Limits and Human Intervention

In Ch. 5, the chapter where we perform proof analysis on the ECS-schema using the schematic CERES method of Ch. 4, we provided relevant parts of the refutation constructed by SPASS of instances of the schematic clause set extracted from ECS-schema. The parts of the refutation we included outlined the induction invariant needed to write the schematic resolution refutation of the clause set of the ECS-schema . For this schema specifically, it was very easy to extract the schematic resolution refutation from the individual instances. After analysing roughly four instances of the clause set's refutation found by SPASS, we where able to construct the schematic resolution refutation. Essentially, the refutations provided by SPASS helped with the discovery of the term invariants used by the global unifiers and the structure of the derived clauses which shows up after every step of the resolution refutation invariant.

However, in the case of the NiA-schema, a much more complex schematic proof, the output provided by SPASS required more analysis and it was not as clear, from the comprehension point of view, as the output for the ECS-schema. Partially this has to do with the compression of the proof tree, but also, SPASS did not find the most efficient method of solving the problem, to be more specific SPASS's solution was quite bad. As a quick note, we tested only the Automatic mode of the named theorem provers. Probably, using specific search strategies would have helped, but this would have been no different from the methods of analysis used in this chapter anyway. To give an idea of how bad SPASS's solution is, let us consider the number of times a clause of schematic length is used at a leaf in the refutation tree as a measure of complexity. The idea behind using this measure of complexity was outlined in Ch. 6.4 when we show a lower bound for the size of the resolution refutation of the clause set. The resolution refutation that SPASS gave as output used a clause of schematic length at a leaf in the refutation tree 1806 times for the instance 4 of the NiA-schema . The resolution refutation we provide in Ch. 6 used a clause of schematic length at a leaf in the refutation tree only 65 times for the instance 4. This pattern can be found in refutations for higher and lower instances as well. The refutation found by SPASS can be found in Appendix D. The following tree represents the resolution refutation found in D.2 expanded so one can see the tree structure of the refutation. Like colors represent places where the branches are to be connected and the numbers are taken directly from the proof in D.2. The numbers mark derived clauses. The number 1 represents the number of the schematic clause which can be found at the leaves of the refutation and thus, it is the clause we are counting.



$$\dfrac{2795 \qquad \dfrac{\dfrac{27 \qquad \dfrac{\dfrac{\dfrac{2797 \quad 50}{2831} \quad \dfrac{2833 \quad 28}{2896}}{2904} \qquad \dfrac{70 \quad 2795}{2797}}{2935}}{2969} \qquad 2934}{2984}}{3015}$$

$$\dfrac{3015 \qquad \dfrac{\dfrac{42 \quad 3015}{3016} \qquad \dfrac{27 \quad \dfrac{43 \quad 3015}{3017}}{3050}}{3065}}{3096}$$

$$\dfrac{\dfrac{23 \quad 3096}{23} \qquad 3096}{3098}$$

What one can see from the outline of this resolution refutation is that sub-trees deeper in the refutation are called on many different branches, essentially, we are multiplying the number of occurrences of those sub-trees. Though, it is hard to see, this results in super-factorial growth in the size of the refutation as we increase the instantiation of the free parameter. Thus, unlike with the ECS-schema, were we where able to, almost, directly construct the schematic resolution refutation from the instances found by SPASS, in the case of the NiA-schema, a lot of post-processing was necessary to find the schematic resolution refutation.

10.2 From SPASS output to Schematic Refutation

The title of this section is a little misleading being that we will not guide the reader through every step of the process, but rather highlight the points when SPASS helped us find crucial steps needed for the resolution refutation construction (for an example in mathematics see [46]). We do not find the output provided by SPASS, nor of any theorem prover, in general necessary when constructing schematic resolution refutations, but the output is extremely helpful with finding patterns which otherwise can take ages to find, or are constructed in such a way, by the prover, that a human would not think about them or find them.

One of the first observation we made, given the output of SPASS, concerning the Nia-schema was that, within the set of derived clauses used by SPASS to refute the clause set, there are schemata of clauses which start from a clause within the clause set and end at the root of the refutation. The schema of clauses which we find particularly important starts at the clause with schematic length in the original clause set and at every step in the derived schema of clauses the length of the derived clause is reduced by one. Here is the schema of clauses found with the set of clauses derived by SPASS as found in D.2:

$1[0 : Inp] \,\|\|\, \Rightarrow \; eq(f(U), 3) \; , \; eq(f(U), 2) \; , \; eq(f(U), 1) \; , \; eq(f(U), 0)*$

$2795[0:MRR:1.3,2764.0]\ |||\ \Rightarrow\ eq(f(U),3)\ ,\ eq(f(U),2)\ ,\ eq(f(U),1)$

$3015[0:MRR:2795.2,2984.0]\ |||\ \Rightarrow\ eq(f(U),3)\ ,\ eq(f(U),2)$

$3096[0:MRR:3015.1,3065.0]\ |||\ \Rightarrow\ eq(f(U),3)$

Essentially, if we where to interpret the initial clause as defining a function (a function whose domain is the natural numbers and whose codomain is the set $[0,n]$) we see that at first we assume the function has a codomain of size four, and than we derive that it cannot have a codomain of size four, but rather of size three, and so on, until we derive that its codomain is empty contradicting the original assumption, that is that the codomain is non-empty. This pattern can be found in other instances of the NiA-schema. We include a refutation for the instance three, for comparison, which can be found in D.1.

Given these four clauses for the fourth instance of the NiA-schema's clause set (Three clauses for the third instance), it seems like we have discovered, with the help of SPASS, a important part of the recursion mechanism needed to refute the clause set. However, looking at the numbers assigned to the clauses, we can see that there is a large gap between the initial clause and the first reduction of the initial clause, i.e. clause 1 and clause 2795. Even though theorem provers do not give consecutive numbers to the clauses used in the final proofs one can easily see in D.2 that a lot of work is done between clause 1 and clause 2795 to get the resulting refutation. This is not as apparent in D.1, which is why we decided to work with the fourth instance in this chapter.

At this point, we started to realise the complexity of this refutation, not just the one provided by SPASS, but in general, i.e. a simple one loop program (recursive function without a nested primitive recursion) would not suffice. To make this more apparent, using the SPASS output, consider how one can make a transition from clause 1 to clause 2795 in one step. It is quite obvious that one will need to derive the clause:

$2764[0:MRR:2714.1,2749.1]\ |||\ eq(f(U),0)*\ \Rightarrow$

Now the question is how hard is it to derive this clause? After one studies the refutation in D.2 for a few minutes it becomes immediately apparent that deriving clause 2764 requires an iteration through every value in the codomain, in other words, it requires an inner loop whose size is dependent on the value of the free parameter .

From this point the importance of one of the derived clauses become apparent, namely the completely negative clause:

$2272[0:Res:955.3,159.1]\ |||\ eq(f(U),0)*\ ,\ eq(f(V),1)*\ ,\ eq(f(W),2)*\ ,\ eq(f(X),3)*\ \Rightarrow$

This clause is necessary for the derivation of not only clause 2764, but also the derivation of all clauses of the form $eq(f(U),k)\ \Rightarrow$ for $0 \leq k < n$. Realization of the necessity of this clause and clauses which are derivable from this clause is what lead to the discover of Lem. 6.5.2.

The last result that was derived from empirical analysis using SPASS was the correct term instantiations for the unification steps in the proof. The terms found directly in SPASS output are similar to, but still along way from, the recursive max term construction Def. 6.5.3. However, the terms used in the SPASS refutations told us that each symbol needed to be associated with a variable in the nested max function. This is the reason why the term used in the unification steps of the SPASS refutation grows with each instance. For example, in the refutation of the third instance the following clause is derived:

$20[0 : Res : 15.0, 4.0] \,||||\, \Rightarrow \; le(U, max(max(max(V, U), W), X))$

in the fourth instance a similar clause is derived:

$54[0 : Res : 19.0, 4.0] \,||||\, \Rightarrow \; le(U, max(max(max(max(V, U), W), X), Y))$

which highlights the growth of this nested term. However, we have not been able to prove the necessity of these max function constructions, nor find a refutation without them. For simplicity of the proofs in Ch. 6, we concatenated several unification steps and adjusted the way the resolution derivation was constructed in the SPASS refutation. This is why Def. 6.5.3 directly has the successor functions in the term and the nesting is right associative.

The result of all these observations as we said was Lem. 6.5.2, which leads to the construction of the following clause set for the instance three:

$1 : eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 2)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 1)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 0)$

$2 : \neg eq(f(B), 2) \vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 1)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 0)$

$3 : \neg eq(f(C), 1) \vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 2)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 0)$

$4 : \neg eq(f(D), 0) \vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 2)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 1)$

$5 : \neg eq(f(B), 2) \vee \neg eq(f(C), 1)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 0)$

$6 : \neg eq(f(B), 2) \vee \neg eq(f(D), 0)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 1)$

$7 : \neg eq(f(C), 1) \vee \neg eq(f(D), 0) \vee$

$eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 2)$

$8 : \neg eq(f(C), 1) \vee \neg eq(f(D), 0) \vee \neg eq(f(B), 2)$

Feeding this derived clause set to SPASS, the result was the following refutation which only shows us the structure of the end of the proof, i.e. how the above clauses should be resolved to construct a refutation. of course this is leaving out the most difficult parts of the resolution refutation, but it also provides us with an idea of the shape of the outer most resolution refutation invariant.

$1[0 : Inp]|| \Rightarrow eq(f(max(max(max(s(U), s(V)), s(W)), s(X))), 2)*$, $eq(f(max(max(max(s(U), s(V)), s(W)), s(X))), 1)$, $eq(f(max(max(max(s(U), s(V)), s(W)), s(X))), 0$

$2[0 : Inp]||eq(f(U), 2) \Rightarrow eq(f(max(max(max(s(V), s(W)), s(U)), s(X))), 1)*$, $eq(f(max(max(max(s(V), s(W)), s(U)), s(X))), 0$

$3[0 : Inp]||eq(f(U), 1) \Rightarrow eq(f(max(max(max(s(V), s(U)), s(W)), s(X))), 2)*$, $eq(f(max(max(max(s(V), s(U)), s(W)), s(X))), 0$

$4[0 : Inp]||eq(f(U), 0) \Rightarrow eq(f(max(max(max(s(U), s(V)), s(W)), s(X))), 2)*$, $eq(f(max(max(max(s(U), s(V)), s(W)), s(X))), 1$

$5[0 : Inp]||eq(f(U), 2)$, $eq(f(V), 1) \Rightarrow eq(f(max(max(max(s(W), s(V)), s(U)), s(X))), 0)$

$6[0 : Inp]||eq(f(U), 2)$, $eq(f(V), 0) \Rightarrow eq(f(max(max(max(s(V), s(W)), s(U)), s(X))), 1)$

$7[0 : Inp]||eq(f(U), 1)$, $eq(f(V), 0) \Rightarrow eq(f(max(max(max(s(V), s(U)), s(W)), s(X))), 2)$

$8[0 : Inp]||eq(f(U), 1)*$, $eq(f(V), 0)*$, $eq(f(W), 2)* \Rightarrow$

$9[0 : MRR : 5.2, 8.1]||eq(f(U), 2)*$, $eq(f(V), 1)* \Rightarrow$

$10[0 : MRR : 6.2, 9.1]||eq(f(U), 2)*$, $eq(f(V), 0)* \Rightarrow$

$11[0 : MRR : 7.2, 10.0]||eq(f(U), 1)*$, $eq(f(V), 0)* \Rightarrow$

$12[0 : MRR : 2.1, 2.2, 9.1, 10.1]||eq(f(U), 2)* \Rightarrow$

$13[0 : MRR : 3.1, 3.2, 12.0, 11.1]||eq(f(U), 1)* \Rightarrow$

$14[0 : MRR : 4.1, 4.2, 12.0, 13.0]||eq(f(U), 0)* \Rightarrow$

$15[0 : MRR : 1.0, 1.1, 1.2, 12.0, 13.0, 14.0]|| \Rightarrow$

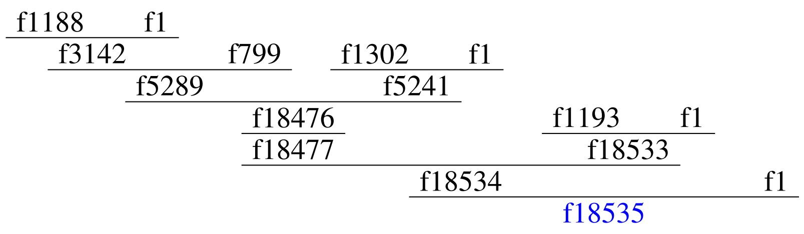
For the derived clause set and refutation of fourth instance see Appendix D.3.

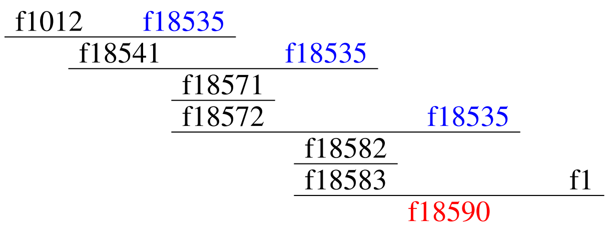
10.3 Applying the above methods to the *g*NiA-schema

The empirical analysis outlined in the previous section was very successful and lead to the construction of a schematic resolution refutation. As mentioned earlier, there is a good chance that, without the use of a theorem prover, the process would have taken much more time, or even, be out of the reach of a human. In the case of *g*NiA-schema, empirical analysis aided by theorem provers is even more essential given that there are two parameters and more than one schematic length clause. Luckily, the schematic length clauses are schematic in only one of the free parameters. However, there is a set of clauses where each clause is schematic in length concerning one of the parameters and the set itself is schematic in size concerning the other parameter.

However, due to the complexity of the *g*NiA-schema, even the simplest extension, the case when we have two symbols and one of the symbols must occur at least three times, cannot be refutated by SPASS. However, in many of the instances when SPASS failed VAMPIRE [40] was able to find a refutation, thus in this section many of the results will use refutations found by VAMPIRE rather than refutations found by SPASS. However, due to readability of the output, when SPASS was able to find a refutation we will refer to these results rather than the refutation found by VAMPIRE.

Getting back to the case of two symbols and one of the symbols must occur at least three times, An interesting structure occurs within the refutation:





$$\dfrac{\dfrac{\dfrac{\dfrac{\text{f45} \quad \text{f2}}{\text{f7263}} \quad \text{f18590}}{\text{f18648}} \quad \text{f18590}}{\text{f18649}} \quad \text{f18590}}{\text{f18650}}$$

The structure we are referring to is the repetition of $f18590$ and $f18535$ three times.

$fof(f18535, plain, ((![X0, X1] : (eq(f(max(s(X0), X1)), 0)| \; eq(f(X0), 1)))),$
$inference(resolution, [], [f18534, f1])).$

$fof(f18590, plain, ((![X0] : (eq(f(X0), 0)))),$
$inference(subsumption_resolution, [], [f1, f18583])).$

In the refutation referred to in the previous section, the refutations of Appendix D, similar clauses where only used twice. This extra occurrence is directly a result of increasing the number of times each symbol can occur. This extra occurrence manifests itself in the clauses of Sec. 9.1, specifically C^g4_0 to C^g4_n. Considering the results of Sec. 9.2 where the combinatorics is related to Pascal's Pyramid rather than Pascal's Triangle , as was the case in Ch. 6, This change in the number of calls to a specific derived clauses eludes to the change in dimensionality we see in the combinatoric interpretations . However, this is still conjecture in that we have not been able to analyse enough instances for various cases to find a definitive pattern and prove results about these patterns. The full refutation as found by VAMPIRE can be found in Appendix D.4.

Other than this extra dimensional structure, it was very difficult to find patterns given how few instances are refutable using the theorem provers at hand, pretty much the most state of the art theorem provers. Thus, we considered using the results from the empirical analysis for the NiA-schema for the gNiA-schema in the sense that we assumed similar patterns will hold. This is exactly the same idea which was used in Ch. 9. Again, as mentioned in Ch. 9, we assumed that there exists a lemma similar to Lem. 6.5.2 for the gNiA-schema, and that the nested max functions will be used in a similar way. Remember that the part that was difficult for the theorem prover concerning the NiA-schema was precisely the work it had to do prior to deriving the clause set defined by Lem. 6.5.2. Thus, our idea was that, by all these extra assumptions, we can cut out the difficult part and get the theorem provers to provide refutations for more instances than when we feed the provers the standard clause set as extracted from the schematic formal proof. This is especially useful when we know how to derive the clauses, and the theorem prover has an extremely hard time figuring it out.

Thinking about the extra occurrences as defined by the second parameter as adding extra dimensional structure aided our construction of the conjectured gNiA-schema Lem. 6.5.2, specifically for the term structure. Essentially, the NiA-schema used a term structure which is a one dimensional array, what if the term structure for three occurrences was a two dimensional array? How can we guarantee that every possible position can be represented in the clause set as was done by Lem. 6.5.2?

It is quite hard to see this term structure in the case studied in Ch. 9. Thus, instead of looking at the case of three symbols m occurrences, we will instead study, the two symbol m occurrences case. Our goal is to find an appropriate term schema which works similarly to the max function

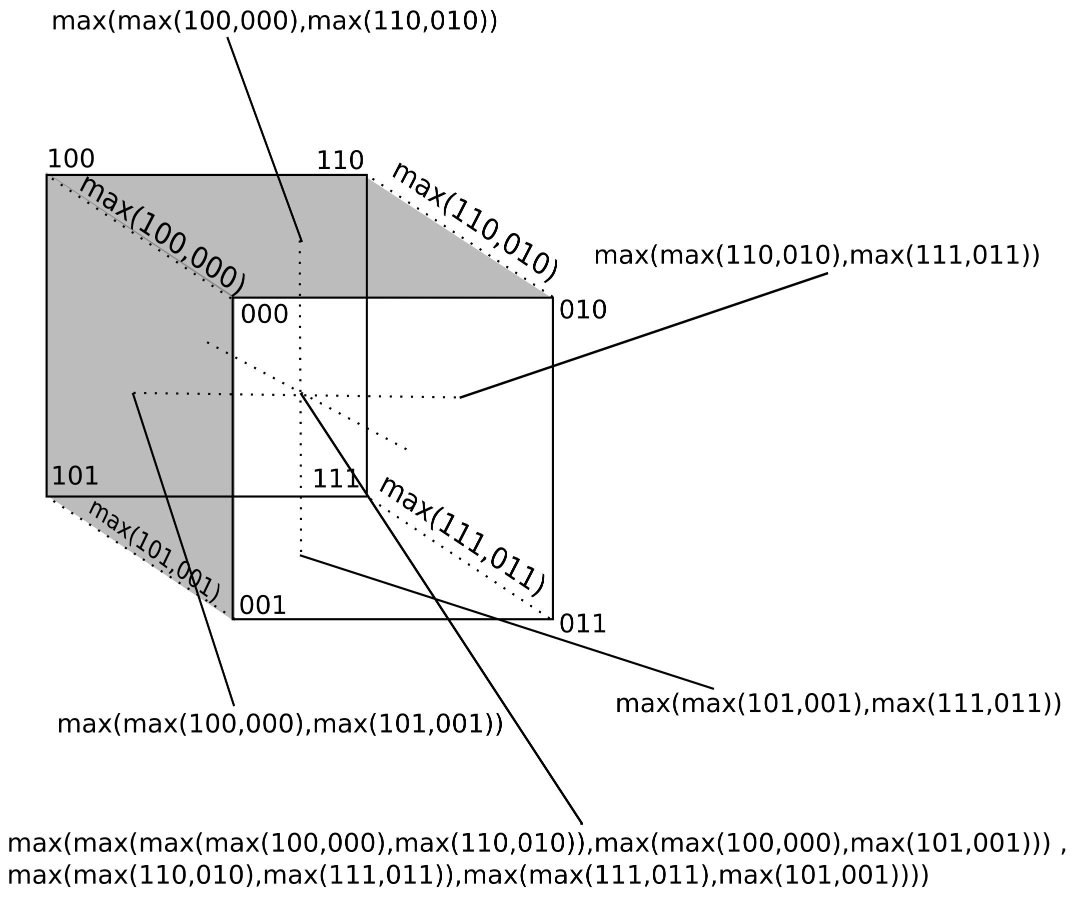

max(max(100,000),max(110,010))
100
110
max(100,000)
max(110,010)
max(max(110,010),max(111,011))
000
010
101
111
max(111,011)
max(101,001)
001
011
max(max(101,001),max(111,011))
max(max(100,000),max(101,001))
max(max(max(max(100,000),max(110,010)),max(max(100,000),max(101,001))) ,
max(max(110,010),max(111,011)),max(max(111,011),max(101,001))))

Figure 10.1: This diagram shows the construction of the derived clauses term schema for the gNiA-schema when the number of symbols is two and the number of occurrences is 4. This diagram represents the term schema for the clause set found in Appendix D.6.

array found in Lem. 6.5.2. As we shall show, only empirically, the structure of this term schema is precisely a generalization of the max function array of Lem. 6.5.2. Also, there is a beautiful combinatorial interpretation of the derived clauses we construct as labellings of the points, lines, and faces of an m-*cube* , i.e. the occurrences of symbols (either zero or one) mark two of the $m - faces$ and the lines between the two chosen n-$faces$ can be labelled by the maximum of the two points the line is connecting (one of which is a zero and one of which is a one). The m-$faces$ which are not of the chosen two can be labelled by the maximum of the lines used to construct it which do not sit on the chosen two m-$faces$. And finally, the cube itself is labelled by the maximum of all the labels of the m-$faces$. Fig. 10.1 provides a diagrammatic representation of the above description.

Unlike Lem. 6.5.2 which constructs all the possible configurations of the function f such that the function is bijective for the first n values in the domain, our construction here only tells us which term language to use in order to get a refutation. The next step would be to use the clause set of D.6 to construct a lemma similar to Lem. 6.5.2 for the *g*NiA-schema. The outcome of such a construction will in almost all likelihood resemble a growth pattern the combinatorial enumeration produced in Ch. 9. Though in this chapter we will refrain from constructing a more general Lem. 6.5.2 for the *g*NiA-schema being that this is beyond the empirical study of the schematic characteristic clause sets and delves deeper into the mathematical structure of the problem.

Also to note, in D.5 & D.6 all derived clauses found in the clause set provided are derivable from instances of the schematic clause set of Ch. 9. Showing that it is possible to derive these clauses is quite trivial, however, it is very messy and does not highlight the focus of this work. What is most interesting to point out about this empirical work is that, working interactively with the theorem prover, one can find the patterns necessary for constructing a resolution refutation schema for a given schematic clause set. The clause set, as derived from the *g*NiA-schema was only refutable for a few instances using SPASS and a few more using VAMPIRE. Using these few derivations and our knowledge of the resolution refutation of the less general NiA-schema we where able to derive the terms needed for the refutation and construct a new schematic clause set subsumed by the original clause set. This new clause set was easier for theorem provers to deal with and provides us with more instances to analyse. Hopefully, in future work we will be able to use these new instances to construct a resolution refutation schema of the *g*NiA-schema's clause set.

CHAPTER 11

Conclusion and Open Problems

The most proper way to start a conclusion to this work would be to take all the puzzle pieces we have been gathering chapter after chapter and putting them together to get a final picture. Up till this point, the disconnect between each of the chapters has been more apparent than their unifying qualities. Though, it is clear that the central topic is analyse of proof schema, it has not been all that apparent why, certain directions where chosen and what the goal was, is currently, and will be for future work. To sum up this hole we have left open, the initial problem, from the very beginning of this dissertation, is the lack of literature, up to the present date, on proof analysis of proof schemata. Other than the work on the Fürstenburg's proof and [31], there has not been much literature concerning this topic. Thus, what we have done for the majority of this dissertation is exploratory research into the topic. At many points not even knowing if the idea, languages, and concepts will actually work. At some points, magically everything works, at other points, we struggle to put the different aspects together to get the desired result.

As an example, The NiA-schema is the result of over a year's time of trying to formalize the tape proof (proof of the infinitary pigeonhole principle) within the schematic LK-calculus . At one point we stopped putting time into the formalization because it seemed that it was not possible. However, in the end we found out that a subtle problem with the cut formulae schema we where using meant that the cut formulae schema was relying on a calculus that allowed a second bounded parameter. Though, as one can see from reading through Ch. 6, we essentially pushed this problem to the construction of the resolution refutation of the schematic characteristic clause set, it still stands out as an example of the difficultly of the subject and of how little is known about proof schemata.

To continue describing this problem, we essentially, with each chapter, explored different ways of dealing with proof schema. From Ch. 5, where everything worked perfectly, to Ch. 10, where we were lost on a raging river heading towards a waterfall without a paddle. We essentially pushed certain chosen concepts to their limits to see what results we got. In some chapters the results where great, see Ch. 8, where we actually got the desired result after much work, to other chapters like Ch. 7, where we got some result, but no one would conclude that it is spectacular.

One of the biggest disconnects found in this work, which we have failed to mention until now, is the disconnect between Ch. 2 and the rest of the thesis. Truthfully, it seems to be tangent to majority of this work; for one, it deals with propositional schematic formulae, and two, it is not about proof analysis. However, an initial goal of the original dissertation plan was to deal with cut-elimination when the proof schemata have two free parameters . We barely scratched the surface of this problem in Ch. 9 & 10, and one may ask why this problem was not tackled to a deeper extent if initially it was thought to be essential. The work of Ch. 2 eludes to the problem of multiple parameters, essentially, if there is a dependency between the two parameters, numerically, than the algorithm of Ch. 2 fails and the problem of proof structure becomes undecidable; this is in the propositional case. When we move to first order and are dealing with proof schemata and cut-elimination where proof structure is extremely important, creating a framework becomes an extremely daunting task. The propositional work was done in order to act as a foundation for the first-order case. However, given the difficultly of accurately describing a minor subset of the propositional problem of multiple parameter schema, we instead focused on the problem of how can one even analyse a proof schema with multiple parameter. It seemed much more essential to see what happens when one adds a second parameter to an already studied proof schema than to dive in head first into the two parameter case. Planned future work we deal with this difficult problem and will be aided by the case study of Ch. 9 & 10 .

We will continue this conclusion with a section for each of the chapters where we will give a description of chapter, the motivation behind it, the future work, and the open problems.

Ch. 2 Propositional Schemata

As mentioned above in this chapter, our work concerning propositional schemata makes up only a small portion of this dissertation and is almost a complete tangent to the rest of the chapters. However, studying the exact conditions when we can have a decidable procedure for propositional schemata with multiple parameter and when we cannot is important for moving to the first-order case. Though, in slightly earlier work Aravantinos et al. [6] a procedure deciding the same class we provide a decision procedure for in Ch. 2 was provided. However, not only did it require a specific language to express the schematic formulae, it also was a much more complex procedure when compared to the procedure of [4]. We showed that a modified version of [4] can decide the class defined in Ch. 2, namely *pure overlap schemata*. Pure overlap schemata are constructed from the concept of relatively pure literals, which are essentially literals of opposite polarity which may be indexed by two different free parameter and do not share the same set of numbers, i.e. there would be no distinction between giving each free parameter a set of natural numbers and given each parameter the same set.

This work provided a precise definition of when problem of constructing a proof of a schematic propositional formulae with multiple parameter is complex enough to be undecidable and not simple to be decidable. We did not use this distinction directly in the chapter dealing with multiple parameter first order proof schemata (the problem of constructing a proof is already undecidable to begin with), but in future work, when trying to build a framework in which proof analysis of multiple parameter first order proof schemata can be dealt with, this distinction can be of importance.

Other than the application to multiple parameter first order proof schemata, there are also a few extensions to this work which can be considered, as well as finding equivalences to temporal logics. Some of this was mentioned in the concluding remarks of Ch. 2. For example, we made mention of the connection between Regular schemata and linear temporal logic [5], It is quite likely that such a connection can be drawn to this new class as well. Also, integrating nested iterations into the theory is another possible research direction. However, one open problem we would still like to answer is is this class defined in Ch. 2 the most expressive decidable multi-parameter class of propositional schemata?

We will skip over both Ch. 3 & 4 given that these chapters only provide preliminaries needed to understand the rest of the dissertation.

Ch. 5 Proof Analysis by Ceres$_s$: the ECS-Schema

As mentioned earlier, the ECS-Schema was designed specifically because the NiA-Schema's resolution refutation could not be expressed in the framework found in Ch. 4. We introduced a more complex framework for the resolution refutations of the proof schema in Ch. 7 and needed another proof schema which could not be expressed in the framework Ch. 4 in order to better define the framework of Ch. 7. However, the ECS-schema that we developed for this task turned out to fit the initial framework. This unexpected result lead to us questioning what is the difference between the two schemata which lead to one being expressible and the other not being expressible.

The way we derived the ECS-schema was by weakening the quantifier complexity of the cuts by switching the $\forall$ and $\exists$ quantifiers found in the cuts of the NiA-schema. Of course, such a change had major consequences for the structure of the proof. As a result the end-sequent had to be skolemized. However, another distinct change to the proof was that the term signature has two monadic function symbols and one constant. The NiA-schema had a binary function symbol in its signature. It is well known that the complete predicate logic can be expressed once a binary function symbol is present. Thus, we conjecture that this change from having a binary function symbol to not having one could have been what allowed the resolution refutation to be expressed in the framework of Ch. 4. It is important to note that such a restriction to the term signature also restricts the quantifier prefix (assuming prenex form) as well being that proof-schema need to be skolemized in other for schematic CERES to work. A complex quantifier prefix can force the introduction of a function with a large arity. We leave as an open problem for future work the discovery of what allowed the ECS-schema to be expressible and what made the NiA-schema not expressible in the framework of Ch. 4. We conjecture that the term signature is the reason for this gap.

One other interesting result for the ECS-schema was the extraction of a schematic Herbrand sequent. We found the schematic Herbrand sequent before the construction of the concept of Herbrand systems mentioned in [31], thus, we stuck to the syntax found in Ch. 2. In turns out that the Herbrand sequent for the ECS-schema is very similar to the Herbrand sequent of the NiA-schema, except an ordering constraint is present. We leave to future work the formalization of this sequent as a Herbrand system and the discovery of an algorithm for the extraction of a Herbrand system from the resolution refutation of a given proof schema formalized in the framework of Ch. 4. I expect that using this example of a proof schema with a Herbrand sequent already extracted can aid future work focused on the designing of an algorithm for extraction.

Ch. 6 Beyond Ceres$_s$: the NiA-Schema

In the introduction, we made mention of why we designed a schema based on the infinitary pigeonhole principle, and essentially it came down to, it is a very important concept in the study of proof complexity in various formal systems, as well as being a basic concept with some mathematical weight behind it. Though, the culmination of all these works did not prepare us for the difficulties this problem would cause when we formalized it as a proof schema. The NiA-schema, has Π_2 cuts (a schematic number of them) unlike the Σ_2 cuts of the ECS-schema. These Π_2 cuts seem to have had a major impact on the complexity of the extracted schematic clause set.

Not only was the refutation of the clause set much more than the calculus found in Ch. 4 could handle it pushed many of the concepts found in Ch. 4 to the limit and we had to rethink some of the definitions. For example, *defined term symbols* turned out to be too weak of a concept because defined terms with arbitrary arity were needed. This same problem was witnessed with defined predicate symbols. Also, the growth rate of the refutation with respect to the free parameter in terms of the number of schematic clauses was extremely fast, factorially growing. However, it is not the growth rate which specifically caused issues it was the fact that it was growing wider at a faster speed then it was growing depthwise. Specifically, it was growing widthwise with respect to the free parameter within the refutation itself. This means that when calling a resolution schema lower in the order, knowledge was needed from the resolution schema which where called from higher in the order. Essentially the deeper one went into the refutation, the more information one would need to compute the next step correctly. The resolution refutation calculus essentially needed memory.

We made mention in the previous section of the conclusion that the necessity (at least it seems to be necessary) of the resolution refutation to have terms with a schematic number of variables is the reason behind the difficulties we came across. This problem is partially due to the issue of the term signature having a binary function symbol. However, one could imagine using a binary function symbol and not having an arbitrary number of variables in the resolution refutation of a schematic clause set. This problem is open for investigation and hopefully will be evaluated in future work.

In the end, rather that a resolution refutation schema within a nice language, like the language from Ch. 4, we ended up with a resolution refutation only by proof. This issue lead us to attempt the design of a new language for resolution refutations of schematic clause sets which is powerful enough to express the refutation of the clause set of the NiA-schema. Before we start on the conclusion for the next chapter, we would like to mention the combinatorial side of this proof of refutability. In the process of constructing a refutation, we found certain patterns in the number of derived clauses used with respect to the instantiation of the free parameter. These patterns, or more correctly enumerations, of specific sets of derived clauses provided a better understanding of what amount of expressive power is needed in order to construct a language powerful enough to express the refutation. This was when we first realized how important the combinatorics of the refutations and of the proof schema in general will be for constructing more powerful frameworks and finding refutations.

Ch. 7 Extension of Schematic Resolution Calculus

As we made mention of at the end of the last section, evaluating the growth rate of certain sets of clauses derived from the initial clause set for the refutation, we where able to deduce a rough idea of what a language powerful enough to express the refutation of the NiA-schema's clause set needed to look like. We used these combinatoric results to define a well ordering which would replace natural numbers as the iterator of the recursively defined resolution refutation. The problem with natural numbers is that they form a total ordering and thus, branching can only be defined (in a primitive recursive way) for a finite number of branches at a time, we needed to define the branching for a schematic number of branches. This ordering, namely the permutation vectors, was then used to define a new type of recursion where the components are guarded by a fragment of array theory. The recursion allows more then two components, each of which does not have a distinct, by definition, set of accepted values. We use a fragment of array theory as a guarding language leaving the definition of the components guards to the user of the language. Many examples are provided in Ch. 7. We also provide a large portion of the resolution refutation from Ch. 5 formalized in this new language. Though there is still a major problem with this language, it is mathematically correct and works for this specific proof schema, but it is not general enough yet to be easily adopted for another similar proof schema. We have not found a good way to formalize which terms can be used for example. From the logical point of view this formalization is quite weak. Part of the expected future work will be to expand on this language and better formalize it. We consider this chapter as scaffolding for future work in the subject.

Ch. 8 Extraction of Schematic Herbrand Sequents: NiA Schema

In our attempt to find a more generalized language, we fell short of our goal. The language, while having the expressive power to define the resolution refutation of the NiA-schema, was not formal enough to really be considered a logical definition of a language. There are still a few loose ends concerning our handling of the allowed terms for the defined term symbols and other similar matters. Thus, in a way this exploratory direction failed, and thus in light of this failure we attempted another possible direction in proof analysis of proof schema, namely extracting a Herbrand sequent from a cut free proof [38].

Though up till this point the construction of the ACNF and cut-free proofs of the NiA-schema have not been of importance, when attempting to extract a Herbrand sequent they are sort of essential (in that there is a way around it). The problem with constructing an ACNF for a proof schema is that the schematic projections must remain separate from the resolution refutation until the parameter is instantiated [30], and secondly, we have not defined schematic projections in this dissertation. The reason we have avoid the schematic projections is because they were published in [30] but later removed from the more recent publication [31] for being too complex to define. Also, in [31] the final output of the procedure was switched to a schematic Herbrand system rather than a schematic ACNF. Another interesting point is the usefulness of a schematic ACNF, in Ch. 6, we have shown that the size of the resolution refutation grows factorially with respect to the free parameter. A Herbrand sequent, or by the language of [31] a Herbrand system, is a much more concise way to respresent essential parts of the proof schema after cut-elimination.

Though, for the Herbrand sequent, instead of full projections we need to use the end sequent of the full projections without weak quantifier rules applied, this might also imply not using contraction on formulae which would have otherwise been contractable in the presence of weak quantification. Also, we will not use a ground projection of the resolution refutation, instead we will apply the unifiers of the resolution refutation to the end-sequent formulae as well as the clause set formulae. The idea here is to collect the changes to the end-sequent of the proof as the formulae pass down the resolution refutation, very similar to the method used in [38], only without a ground projection. One problem we have still, is that we do not have a resolution refutation written down in a specific language. We would like to avoid the language we designed in the last chapter and construct the Herbrand sequent directly from the proof that the clause set is refutable because we wanted the work of this chapter to be parallel to the previous chapter's results. Also, because the results of the previous chapter, as we stated are partially complete. Using the proof that the clause set is refutable, we consider how each step of the proof will change the end sequent formulae. However, there is still a problem with this idea. what if the number of formulae grows schematically, then we run into the problem of trying to describe a very hard to describe sequent.

As we have shown, we dealt with this problem by introducing a class of structures with a total linear ordering and which use the unifiers found in the proof of refutability as a successor function, i.e. moving to the next element in the total ordering. This class of structures was called the natural like numbers. The way we structured the ordering of the natural like numbers was such that individual variables are the bottom of the ordering, and the maximum depth of the schematic max functions define the terms equivalence classes in the ordering. The only problem introduced by the usage of these equivalence classes is that each of the equivalence classes contained all the possible terms with a maximum schematic max term depth n. Many of these terms are not used in the proof what so ever, so the final Herbrand sequent ended up being full of junk. Though, the true Herbrand sequent is guaranteed to be in this junk infested Herbrand sequent. We got around this junk collection problem by proving that there is a bijection between the equivalence classes and the natural numbers. Using this bijection we took the end sequent with all the junk and replaced each of the equivalence classes with the natural numbers they map to. This final Herbrand sequent isn't the true Herbrand sequent, but it is much better for a human to read than both the true Herbrand sequent and the Herbrand sequent with the extra junk.

The problem with the method used in this chapter is that it is again specific to this particular proof schema. Though, a problem with this subject in general is that it is hard to get general results concerning proof schema. However this time, we where able to get the results we where looking for, and to top it off, the resulting schematic Herbrand sequent is every instance of the finite pigeonhole principle. This is exactly the Herbrand sequent we where hoping for. For future work, we are hoping this method can be generalized into something similar to the algorithm of [38]. We believe that with the construction of a more general language, a better version of the work of Ch. 7, a algorithm for Herbrand sequent extraction can be constructed. If an algorithm cannot be constructed for general proof schema, we conjecture that for some reasonable restrictions on the terms used in the proof schema, an algorithm can be constructed. This conjecture is related to the conjecture of Ch.5 concerning monadic function symbol term signatures.

Ch. 9 A Generalization of the NiA-Schema

This is the chapter where we address the problem of multiple parameter first order proof schema for the first time. The way we constructed the *g*NiA-schema was by taking the NiA-schema and adding a second parameter which iterated through the number of times a symbol can occur. For example the standard NiA-schema can be derived from the *g*NiA-schema by setting the second parameter to zero. The influence of this addition parameter on the clause set is that the clauses labelled $C4_i$ become schematic in length based on the value of the second parameter. Even though the *g*NiA-schema is a two parameter schema, we don't consider both parameters at the same time when performing proof analysis, instead we fix the free parameter which adjust the number of symbols in the codomain of the function $f()$ and worked with a one parameter schema over the second free parameter, the number of occurrences of a symbol. This provides us with a completely new type of one parameter proof schema. In Appendix D.7, one can see a very good reason for avoiding two parameter proof schemata without a better study of the single parameter case, namely, the two parameter schema show wildly different behaviour for different instance pairs. The results for the *g*NiA-schema when we have three symbols occurring at most 3 times has derived clauses which we could not match to derived clauses found in lower instantiations.

Rather than performing proof analysis, as we did with the NiA-schema, we instead focused on combinatorial analysis of different sets of derived clauses. Being that the *g*NiA-schema was derived from the NiA-schema, we assumed that certain sets of derived clauses which existed for the NiA-schema will also exist for the *g*NiA-schema. Of course these sets of derived clauses will be more complex than they were for the NiA-schema, and thus, we constructed a new combinatorial word problem accordingly. The idea of this section is to see if combinatorial analysis of the clause set can help in the discovery of the recursion necessary for the defining the resolution refutation of the clause set. As we saw with the NiA-schema, analysing the combinatorial structure of the clause set helped in finding the recursion which lead to the resolution refutation.

In the process of finding this recursion, we found out that the set of derived clauses we were analysing had a bijection to a set of permutations. We spend a lot of the chapter analysing this bijection and proving its existence. Though in the end we did find a recursion which may help with the definition of the resolution refutation, but the best results of this chapter dealt with the bijection.

The goal, in terms of future work, is to find a more methodological way of doing this combinatorial analysis. At the moment we just took a guess as to which set to evaluate based on a related proof. As we can see in the next chapter, this chosen set is very hard to find in the theorem prover output, and it is even possible that the theorem provers avoid using it, or it only exist as a meta-level construct. The only derived clause sets we where able to find empirically defined the entire term structure which is needed for the refutation. Nonetheless, the role of combinatorial analysis of proof schema is important in proof analysis of the said object.

Ch. 10 Empirical Analysis of Schematic Clause Sets Using Theorem Provers

In this final chapter of this dissertation we provide insight into the empirical methods which lead to the mathematical results from earlier chapters. The importance of this empirical work can be seen in how difficult it is to study, finding refutations for, and proving things about proof schema. This is especially true about finding refutations of the clause sets extracted from proof schema being that inductive clause sets require finding invariants over resolution derivations, which is not a simple task. Though, this subject (finding such invariants) has not been well studied, and probably does not have any good solutions, at least for the near future, the only thing we can rely on is having a huge number of instances of these recursive clause sets, refute them, and see what patterns we can find by hand.

A key problem with the conclusion of the last paragraph is the "huge" number of instances. It turns out that for pretty small instantiations of the free parameter the clause sets cannot be refuted by even the most advanced theorem provers. The simplest schema in this thesis was only refutable up to instance 9 by SPASS. Thus, we are left with the question, what to do when the instances refutable by the theorem provers are not enough for us to find the invariants? The step we outline in this chapter were mainly based on looking for patterns in the terms used by the theorem provers for the instances they can refute and then make a new clause set with these terms already in the clause set. It is essentially doing unification for the theorem prover. This was used to refute very difficult cases of the gNiA-schema.

Our hope for future work in field of empirical analysis of the characteristic clause sets of proof schema is that the methods used in this work can be made more streamlined. That is, that a specific procedure can be developed. In a sense, the methods outlined in this chapter can be turned into a specific procedure, but we have only tested them on three roughly similar proofs. Once there are more proof schema available/constructed for proof analysis we can refine the methods introduced here and put them in a more general form. Thus, formalizing many more simple logical theorems will aid the empirical analysis of proof schema.

Closing Remarks

The subject of recursive proof theory is at the forefront of our understanding of mathematical structure, though, it is also a subject which has not entirely been given the spotlight, quite the opposite. This has been mainly the fault of its complexity as well as the difficultly of describing results. However, it has been shown in the work of [31] that it is not a useless subject, a subject worth our time. In this paper it was shown that cut-elimination in the presence of induction can be performed using a recursive calculus, something which cannot be performed using the standard calculus.

In this work we have shown how the topic also connects with other field and can actually lead to results in this fields, fields such as enumeration of combinatorial sets, complexity, and combinatorics itself. Our end goal was to introduce into the literature more results and analysis of proof schema and hopefully, provide an invitation to others to start studying the field.

APPENDIX A

Eventually Constant Statement: Proofs and Refutations

A.1 LKS Proof of ECS

Cut ancestors have a * and cut-configuration ancestors have a **. The proof has already been skolemized. There is only one skolem term which is $g(\cdot)$.

ψ stepcase

$$\dfrac{\bigvee_{i=0}^{n} i = f(\alpha) \vdash f(\alpha) < n+1^*}{\begin{array}{c}\bigvee_{i=0}^{n} i \sim f(\alpha), \\ f(\alpha) \leq f(\alpha), 0 \leq \alpha^* \vdash \\ f(\alpha) < n+1^*\end{array}}\; w:l$$

$$\dfrac{\dfrac{\begin{array}{c}(2)\\ \bigvee_{i=0}^{n} i = f(\alpha), \\ f(\alpha) \leq f(\alpha), 0 \leq \alpha^* \vdash \\ f(\alpha) < n+1^*\end{array} \qquad \begin{array}{c} n+1 = f(\alpha) \vdash \\ n+1 = f(\alpha)^*\end{array}}{\begin{array}{c}\bigvee_{i=0}^{n+1} i = f(\alpha), \\ f(\alpha) \leq f(\alpha), 0 \leq \alpha^* \vdash \\ n+1 = f(\alpha)^*, f(\alpha) < n+1^*\end{array}}\; \vee:l \qquad \begin{array}{c}\vdash \\ \alpha \leq \alpha\end{array}}{\begin{array}{c}\bigvee_{i=0}^{n+1} i = f(\alpha), \\ \alpha \leq \alpha \rightarrow f(\alpha) \leq f(\alpha), 0 \leq \alpha^* \vdash \\ n+1 = f(\alpha)^*, f(\alpha) < n+1^* \\ (1)\end{array}}\; \rightarrow: l$$

$$\begin{gathered}(1)\\ \bigvee_{i=0}^{n+1} i = f(\alpha),\\ \alpha \leq \alpha \to f(\alpha) \leq f(\alpha), 0 \leq \alpha^* \vdash\\ n+1 = f(\alpha)^*, f(\alpha) < n+1^*\end{gathered}$$

$$\overline{\hspace{12em}}\ \forall : l$$

$$\begin{gathered}\bigvee_{i=0}^{n+1} i = f(\alpha),\\ \forall y\Big(\alpha \leq y \to f(y) \leq f(\alpha)\Big), 0 \leq \alpha^* \vdash\\ n+1 = f(\alpha)^*, f(\alpha) < n+1^*\end{gathered}$$

$$\overline{\hspace{12em}}\ \forall : l$$

$$\begin{gathered}\bigvee_{i=0}^{n+1} i = f(\alpha),\\ \forall x\forall y\Big(x \leq y \to f(y) \leq f(x)\Big), 0 \leq \alpha^* \vdash\\ n+1 = f(\alpha)^*, f(\alpha) < n+1^*\end{gathered}$$

$$\overline{\hspace{12em}}\ \forall : l$$

$$\begin{gathered}\forall x(\bigvee_{i=0}^{n+1} i = f(x)),\\ \forall x\forall y\Big(x \leq y \to f(y) \leq f(x)\Big), 0 \leq \alpha^* \vdash\\ n+1 = f(\alpha)^*, f(\alpha) < n+1^*\end{gathered}$$

$$\overline{\hspace{12em}}\ \to : r$$

$$\begin{gathered}\forall x(\bigvee_{i=0}^{n+1} i = f(x)),\\ \forall x\forall y\Big(x \leq y \to f(y) \leq f(x)\Big) \vdash\\ ((0 \leq \alpha) \to n+1 = f(\alpha))^*\\ , f(\alpha) < n+1^*\end{gathered}$$

$$\overline{\hspace{12em}}\ \vee : r$$

$$\begin{gathered}\forall x(\bigvee_{i=0}^{n+1} i = f(x)),\\ \forall x\forall y\Big(x \leq y \to f(y) \leq f(x)\Big) \vdash\\ ((0 \leq \alpha) \to n+1 = f(\alpha))\\ \vee f(\alpha) < n+1^*\end{gathered}$$

$$\overline{\hspace{12em}}\ \forall : r$$

$$\begin{gathered}\forall x(\bigvee_{i=0}^{n+1} i = f(x)),\\ \forall x\forall y\Big(x \leq y \to f(y) \leq f(x)\Big) \vdash\\ \forall y\,(((0 \leq y) \to n+1 = f(y))\\ \vee f(y) < n+1)^*\end{gathered}$$

$$\overline{\hspace{12em}}\ \exists : r$$

$$\begin{gathered}\forall x(\bigvee_{i=0}^{n+1} i = f(x)),\\ \forall x\forall y\Big(x \leq y \to f(y) \leq f(x)\Big) \vdash\\ \exists x\forall y\,(((x \leq y) \to n+1 = f(y))\\ \vee f(y) < n+1)^*\\ (3)\end{gathered}$$

$$\dfrac{\begin{gathered}(3)\\ \forall x(\bigvee_{i=0}^{n+1} i = f(x)),\\ \forall x\forall y\Big(x \leq y \to f(y) \leq f(x)\Big) \vdash\\ \exists x\forall y\,(((x \leq y) \to n+1 = f(y))\\ \vee f(y) < n+1)^*\end{gathered} \qquad \begin{gathered}\varphi(n+1)\\ \cdots\cdots\cdots\cdots\cdots\cdots\\ \exists x\forall y\,(((x \leq y) \to n+1 = f(y))\\ \vee f(y) < n+1)^* \vdash\\ \exists x(x \leq g(x) \to f(x) = f(g(x)))\end{gathered}}{\begin{gathered}\forall x(\bigvee_{i=0}^{n+1} i = f(x)),\\ \forall x\forall y\Big(x \leq y \to f(y) \leq f(x)\Big) \vdash\\ \exists x(x \leq g(x) \to f(x) = f(g(x)))\end{gathered}}\ cut$$

ψ **basecase**

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}0 = f(\alpha) \vdash \\ 0 = f(\alpha)^*\end{array}}{\begin{array}{c}0 = f(\alpha), \\ f(\alpha) \leq f(\alpha), 0 \leq \alpha^* \vdash \\ 0 = f(\alpha)^*\end{array}}\ w:l}{\begin{array}{c}0 = f(\alpha), \\ f(\alpha) \leq f(\alpha), 0 \leq \alpha^* \vdash \\ 0 = f(\alpha)^*, f(\alpha) < 0^*\end{array}}\ w:r \qquad \begin{array}{c}\vdash \\ \alpha \leq \alpha\end{array}}{\begin{array}{c}0 = f(\alpha), \\ \alpha \leq \beta \rightarrow f(\beta) \leq f(\alpha), 0 \leq \alpha^* \vdash \\ 0 = f(\alpha)^*, f(\alpha) < 0^* \\ (1)\end{array}}\ \rightarrow: l$$

$$\begin{array}{cl}
(1) & \\
0 = f(\alpha), & \\
\alpha \leq \alpha \to f(\alpha) \leq f(\alpha), 0 \leq \alpha^* \vdash & \\
0 = f(\alpha)^*, f(\alpha) < 0^* & \\
\hline
0 = f(\alpha), & \forall : l \\
\forall y\Big(\alpha \leq y \to f(y) \leq f(\alpha)\Big), 0 \leq \alpha^* \vdash & \\
0 = f(\alpha)^*, f(\alpha) < 0^* & \\
\hline
0 = f(\alpha), & \forall : l \\
\forall x \forall y\Big(x \leq y \to f(y) \leq f(x)\Big), 0 \leq \alpha^* \vdash & \\
0 = f(\alpha)^*, f(\alpha) < 0^* & \\
\hline
\forall x(0 = f(x)), & \forall : l \\
\forall x \forall y\Big(x \leq y \to f(y) \leq f(x)\Big), 0 \leq \alpha^* \vdash & \\
0 = f(\alpha)^*, f(\alpha) < 0^* & \\
\hline
\forall x(0 = f(x)), & \to : r \\
\forall x \forall y\Big(x \leq y \to f(y) \leq f(x)\Big) \vdash & \\
((0 \leq \alpha) \to 0 = f(\alpha))^* & \\
, f(\alpha) < 0^* & \\
\hline
\forall x(0 = f(x)), & \vee : r \\
\forall x \forall y\Big(x \leq y \to f(y) \leq f(x)\Big) \vdash & \\
((0 \leq \alpha) \to 0 = f(\alpha)) & \\
\vee f(\alpha) < 0^* & \\
\hline
\forall x(0 = f(x)), & \forall : r \\
\forall x \forall y\Big(x \leq y \to f(y) \leq f(x)\Big) \vdash & \\
\forall y\,(((0 \leq y) \to 0 = f(y)) & \\
\vee f(y) < 0)^* & \\
\hline
\forall x(0 = f(x)), & \exists : r \\
\forall x \forall y\Big(x \leq y \to f(y) \leq f(x)\Big) \vdash & \\
\exists x \forall y\,(((x \leq y) \to 0 = f(y)) & \\
\vee f(y) < 0)^* & \\
(3) &
\end{array}$$

$$\dfrac{\begin{array}{c} (3) \\ \forall x(0 = f(x)), \\ \forall x \forall y\Big(x \leq y \to f(y) \leq f(x)\Big) \vdash \\ \exists x \forall y\,(((x \leq y) \to 0 = f(y)) \\ \vee f(y) < 0)^* \end{array} \qquad \begin{array}{c} \varphi(0) \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ \exists x \forall y\,(((x \leq y) \to 0 = f(y)) \\ \vee f(y) < 0)^* \vdash \\ \exists x(x \leq g(x) \to f(x) = f(g(x))) \end{array}}{\begin{array}{c} \forall x(0 = f(x)), \\ \forall x \forall y\Big(x \leq y \to f(y) \leq f(x)\Big) \vdash \\ \exists x(x \leq g(x) \to f(x) = f(g(x))) \end{array}}\ cut$$

φ stepcase

$$\dfrac{n+1=f(\beta)^{**} \vdash n+1=f(\beta)^{*} \qquad \alpha \leq \beta^{*} \vdash \alpha \leq \beta^{**}}{((\alpha \leq \beta) \to n+1=f(\beta))^{**},\ \alpha \leq \beta^{*} \vdash n+1=f(\beta)^{*}} \ \to: l$$

$$\dfrac{((\alpha \leq \beta) \to n+1=f(\beta))^{**},\ \alpha \leq \beta^{*} \vdash n+1=f(\beta)^{*} \qquad f(\beta) < n+1^{**} \vdash f(\beta) < n+1^{*}}{((\alpha \leq \beta) \to n+1=f(\beta)) \vee f(\beta) < n+1^{**}, \alpha \leq \beta^{*} \vdash f(\beta) < n+1^{*},\ n+1=f(\beta)^{*}} \ \vee : l$$

$$\dfrac{}{((\alpha \leq \beta) \to n+1=f(\beta)) \vee f(\beta) < n+1^{**} \vdash f(\beta) < n+1^{*},\ \alpha \leq \beta \to n+1=f(\beta)^{*}} \ \to: r$$

$$\dfrac{}{((\alpha \leq \beta) \to n+1=f(\beta)) \vee f(\beta) < n+1^{**} \vdash \exists x\,(f(x) < n+1)^{*},\ \alpha \leq \beta \to n+1=f(\beta)^{*}} \ \exists : r$$

$$\dfrac{}{\forall y\,(((\alpha \leq y) \to n+1=f(y)) \vee f(y) < n+1)^{**} \vdash \exists x\,(f(x) < n+1)^{*},\ \alpha \leq \beta \to n+1=f(\beta)^{*}} \ \forall : l$$

$$\dfrac{}{\forall y\,(((\alpha \leq y) \to n+1=f(y)) \vee f(y) < n+1)^{**} \vdash \exists x\,(f(x) < n+1)^{*},\ \forall y(\alpha \leq y \to n+1=f(y))^{*}} \ \exists : r$$

$$\dfrac{}{\forall y\,(((\alpha \leq y) \to n+1=f(y)) \vee f(y) < n+1)^{**} \vdash \exists x\,(f(x) < n+1)^{*},\ \exists x \forall y(x \leq y \to n+1=f(y))^{*}} \ \exists : r$$

$$\dfrac{}{\exists x \forall y\,(((x \leq y) \to n+1=f(y)) \vee f(y) < n+1)^{**} \vdash \exists x\,(f(x) < n+1)^{*},\ \exists x \forall y(x \leq y \to n+1=f(y))^{*}} \ \exists : l$$

(3)

$$
\dfrac{\begin{array}{c}(3)\\ \exists x \forall y\,(((x \leq y) \to n+1 = f(y)) \\ \vee f(y) < n+1)^{**} \vdash \\ \exists x\,(f(x) < n+1)^*, \\ \exists x \forall y (x \leq y \to n+1 = f(y))^*\end{array}
\qquad
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c} f(\alpha) < n+1^*, \alpha \leq \beta^* \vdash \\ n = f(\beta)^*, f(\beta) < n^* \end{array}}{\begin{array}{c} f(\alpha) < n+1^* \vdash \\ (\alpha \leq \beta) \to n = f(\beta)^*, f(\beta) < n^* \end{array}}\to: r}{\begin{array}{c} f(\alpha) < n+1^* \vdash \\ ((\alpha \leq \beta) \to n = f(\beta)) \vee f(\beta) < n^* \end{array}}\vee : r}{\begin{array}{c} f(\alpha) < n+1^* \vdash \\ \forall y\,(((\alpha \leq y) \to n = f(y)) \\ \vee f(y) < n)^* \end{array}}\forall : r}{\begin{array}{c} f(\alpha) < n+1^* \vdash \\ \exists x \forall y\,(((x \leq y) \to n = f(y)) \\ \vee f(y) < n)^* \end{array}}\exists : r}{\begin{array}{c} \exists x\,(f(x) < n+1)^* \vdash \\ \exists x \forall y\,(((x \leq y) \to n = f(y)) \\ \vee f(y) < n)^* \end{array}}\exists : l}{\begin{array}{c} \exists x \forall y\,(((x \leq y) \to n+1 = f(y)) \\ \vee f(y) < n+1)^{**} \vdash \\ \exists x \forall y\,(((x \leq y) \to n = f(y)) \\ \vee f(y) < n)^*, \\ \exists x \forall y (x \leq y \to n+1 = f(y))^* \\ (1)\end{array}}\mathit{cut}
$$

$$
\dfrac{\dfrac{\begin{array}{c} n+1 = f(\alpha)^*, \\ n+1 = f(g(\alpha))^* \vdash \\ f(\alpha) = f(g(\alpha)) \end{array} \qquad \vdash \alpha \leq \alpha^*}{\begin{array}{c} \alpha \leq \alpha \to n+1 = f(\alpha)^*, \\ n+1 = f(g(\alpha))^* \vdash \\ f(\alpha) = f(g(\alpha)) \end{array}}\to: l \qquad \begin{array}{c} \alpha \leq g(\alpha) \vdash \\ \alpha \leq g(\alpha)^* \end{array}}{\begin{array}{c} \alpha \leq \alpha \to n+1 = f(\alpha)^*, \\ \alpha \leq g(\alpha) \to n+1 = f(g(\alpha))^*, \\ \alpha \leq g(\alpha) \vdash \\ f(\alpha) = f(g(\alpha)) \\ (2) \end{array}}\to: l
$$

$$\dfrac{\begin{array}{c}(1)\\ \exists x\forall y\,(((x\leq y)\to n+1=f(y))\\ \vee f(y)<n+1)^{**}\vdash\\ \exists x\forall y\,(((x\leq y)\to n=f(y))\\ \vee f(y)<n)^{*},\\ \exists x\forall y(x\leq y\to n+1=f(y))^{*}\end{array}\qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c}(2)\\ \alpha\leq\alpha\to n+1=f(\alpha)^{*},\\ \alpha\leq g(\alpha)\to n+1=f(g(\alpha))^{*},\\ \alpha\leq g(\alpha)\vdash\\ f(\alpha)=f(g(\alpha))\end{array}}{\begin{array}{c}\alpha\leq\alpha\to n+1=f(\alpha)^{*},\\ \alpha\leq g(\alpha)\to n+1=f(g(\alpha))^{*}\vdash\\ \alpha\leq g(\alpha)\to f(\alpha)=f(g(\alpha))\end{array}}\to:r}{\begin{array}{c}\alpha\leq\alpha\to n+1=f(\alpha)^{*},\\ \alpha\leq g(\alpha)\to n+1=f(g(\alpha))^{*}\vdash\\ \exists x(x\leq g(x)\to f(x)=f(g(x)))\end{array}}\exists:r}{\begin{array}{c}\alpha\leq\alpha\to n+1=f(\alpha)^{*},\\ \forall y(\alpha\leq y\to n+1=f(y))^{*}\vdash\\ \exists x(x\leq g(x)\to f(x)=f(g(x)))\end{array}}\forall:l}{\begin{array}{c}\forall y(\alpha\leq y\to n+1=f(y))^{*},\\ \forall y(\alpha\leq y\to n+1=f(y))^{*}\vdash\\ \exists x(x\leq g(x)\to f(x)=f(g(x)))\end{array}}\forall:l}{\begin{array}{c}\forall y(\alpha\leq y\to n+1=f(y))^{*}\vdash\\ \exists x(x\leq g(x)\to f(x)=f(g(x)))\end{array}}c:l}{\begin{array}{c}\exists x\forall y(x\leq y\to n+1=f(y))^{*}\vdash\\ \exists x(x\leq g(x)\to f(x)=f(g(x)))\end{array}}\exists:l}{\begin{array}{c}\exists x\forall y\,(((x\leq y)\to n+1=f(y))\\ \vee f(y)<n+1)^{**}\vdash\\ \exists x\forall y\,(((x\leq y)\to n=f(y))\\ \vee f(y)<n)^{*},\\ \exists x(x\leq g(x)\to f(x)=f(g(x)))\\ (4)\end{array}}cut$$

$$\dfrac{\dfrac{\begin{array}{c}(4)\\ \exists x\forall y\,(((x\leq y)\to n+1=f(y))\\ \vee f(y)<n+1)^{**}\vdash\\ \exists x\forall y\,(((x\leq y)\to n=f(y))\\ \vee f(y)<n)^{*},\\ \exists x(x\leq g(x)\to f(x)=f(g(x)))\end{array}\qquad \begin{array}{c}\varphi(n)\\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\ \exists x\forall y\,(((x\leq y)\to n=f(y))\\ \vee f(y)<n)^{*}\vdash\\ \exists x(x\leq g(x)\to f(x)=f(g(x)))\end{array}}{\begin{array}{c}\exists x\forall y\,(((x\leq y)\to n+1=f(y))\\ \vee f(y)<n+1)^{**}\vdash\\ \exists x(x\leq g(x)\to f(x)=f(g(x))),\\ \exists x(x\leq g(x)\to f(x)=f(g(x)))\end{array}}cut}{\begin{array}{c}\exists x\forall y\,(((x\leq y)\to n+1=f(y))\\ \vee f(y)<n+1)^{**}\vdash\\ \exists x(x\leq g(x)\to f(x)=f(g(x)))\end{array}}c:l$$

φ **basecase**

$$\dfrac{\dfrac{\begin{array}{c}0=f(\alpha)^{**},\\ 0=f(g(\alpha))^{**}\vdash\\ f(\alpha)=f(g(\alpha))\end{array}\qquad \alpha\leq g(\alpha)\vdash\alpha\leq g(\alpha)^{**}}{\begin{array}{c}0=f(\alpha)^{**},\\ \left((\alpha\leq g(\alpha))\rightarrow 0=f(g(\alpha))\right)^{**},\\ \alpha\leq g(\alpha)\vdash\\ f(\alpha)=f(g(\alpha))\end{array}}\rightarrow: l\qquad \vdash\alpha\leq\alpha^{**}}{\begin{array}{c}(\alpha\leq\alpha)\rightarrow 0=f(\alpha)^{**},\\ \left((\alpha\leq g(\alpha))\rightarrow 0=f(g(\alpha))\right)^{**},\\ \alpha\leq g(\alpha)\vdash\\ f(\alpha)=f(g(\alpha))\\ (1)\end{array}}\rightarrow: l$$

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c}(1)\\ (\alpha \leq \alpha) \to 0 = f(\alpha)^{**},\\ ((\alpha \leq g(\alpha)) \to 0 = f(g(\alpha)))^{**},\\ \alpha \leq g(\alpha) \vdash\\ f(\alpha) = f(g(\alpha))\end{array} \qquad f(g(\alpha)) < 0^{**} \vdash}{\begin{array}{c}(\alpha \leq \alpha) \to 0 = f(\alpha)^{**},\\ ((\alpha \leq g(\alpha)) \to 0 = f(g(\alpha)))\\ \vee f(g(\alpha)) < 0^{**},\\ \alpha \leq g(\alpha) \vdash\\ f(\alpha) = f(g(\alpha))\end{array}}\ \vee : l \qquad f(\alpha) < 0^{**} \vdash}{\begin{array}{c}((\alpha \leq \alpha) \to 0 = f(\alpha))\\ \vee f(\alpha) < 0^{**},\\ ((\alpha \leq g(\alpha)) \to 0 = f(g(\alpha)))\\ \vee f(g(\alpha)) < 0^{**},\\ \alpha \leq g(\alpha) \vdash\\ f(\alpha) = f(g(\alpha))\end{array}}\ \vee : l}{\begin{array}{c}((\alpha \leq \alpha) \to 0 = f(\alpha))\\ \vee f(\alpha) < 0^{**},\\ ((\alpha \leq g(\alpha)) \to 0 = f(g(\alpha)))\\ \vee f(g(\alpha)) < 0^{**} \vdash\\ \alpha \leq g(\alpha) \to f(\alpha) = f(g(\alpha))\end{array}}\ \to: r}{\begin{array}{c}((\alpha \leq \alpha) \to 0 = f(\alpha))\\ \vee f(\alpha) < 0^{**},\\ ((\alpha \leq g(\alpha)) \to 0 = f(g(\alpha)))\\ \vee f(g(\alpha)) < 0^{**} \vdash\\ \exists x(x \leq g(x) \to f(x) = f(g(x)))\end{array}}\ \exists : r}{\begin{array}{c}((\alpha \leq \alpha) \to 0 = f(\alpha))\\ \vee f(\alpha) < 0^{**},\\ \forall y\,(((\alpha \leq y) \to 0 = f(y))\\ \vee f(y) < 0)^{**} \vdash\\ \exists x(x \leq g(x) \to f(x) = f(g(x)))\end{array}}\ \forall : l}{\begin{array}{c}\forall y\,(((\alpha \leq y) \to 0 = f(y))\\ \vee f(y) < 0)^{**},\\ \forall y\,(((\alpha \leq y) \to 0 = f(y))\\ \vee f(y) < 0)^{**} \vdash\\ \exists x(x \leq g(x) \to f(x) = f(g(x)))\end{array}}\ \forall : l}{\begin{array}{c}\forall y\,(((\alpha \leq y) \to 0 = f(y))\\ \vee f(y) < 0)^{**} \vdash\\ \exists x(x \leq g(x) \to f(x) = f(g(x)))\end{array}}\ c : l}{\begin{array}{c}\exists x \forall y\,(((x \leq y) \to 0 = f(y))\\ \vee f(y) < 0)^{**} \vdash\\ \exists x(x \leq g(x) \to f(x) = f(g(x)))\end{array}}\ \exists : l
$$

A.2 Refutation of $C^{ECS}(n)$ for $n = 3$

In this section we provide an unrolled version of the schematic resolution refutation.

$$\dfrac{\dfrac{\begin{array}{c} x \leq y, \\ f(x) < 1 \vdash \\ 0 = f(y), \\ f(y) < 0 \end{array} \qquad \vdash g(g(g(x))) \leq g(g(g(g(x))))}{\begin{array}{c} f(g(g(g(x)))) < 1 \vdash \\ 0 = f(g(g(g(g(x))))), \\ f(g(g(g(g(x))))) < 0 \end{array}} \qquad f(g(g(g(g(x))))) < 0 \vdash}{\begin{array}{c} f(g(g(g(x)))) < 1 \vdash \\ 0 = f(g(g(g(g(x))))) \\ (30) \end{array}}$$

$$\dfrac{\begin{array}{c} (30) \\ f(g(g(g(x)))) < 1 \vdash \\ 0 = f(g(g(g(g(x))))) \end{array} \qquad \begin{array}{c} 0 = f(g(g(g(g(x))))), \\ 0 = f(g(g(g(x)))) \vdash \end{array}}{\begin{array}{c} f(g(g(g(x)))) < 1, \\ 0 = f(g(g(g(x)))) \vdash \\ (29) \end{array}}$$

$$\dfrac{\dfrac{\begin{array}{c} (29) \\ f(g(g(g(x)))) < 1, \\ 0 = f(g(g(g(x)))) \vdash \end{array} \qquad \dfrac{\vdash g(g(g(x))) \leq g(g(g(x))) \qquad \begin{array}{c} x \leq y, \\ f(x) < 1 \vdash \\ 0 = f(y), \\ f(y) < 0 \end{array}}{\begin{array}{c} f(g(g(g(x)))) < 1 \vdash \\ 0 = f(g(g(g(x)))), \\ f(g(g(g(x)))) < 0 \end{array}}}{\begin{array}{c} f(g(g(g(x)))) < 1 \\ \vdash f(g(g(g(x)))) < 0 \end{array}} \qquad f(g(g(g(x)))) < 0 \vdash}{\begin{array}{c} f(g(g(g(x)))) < 1 \vdash \\ (26) \end{array}}$$

$$\dfrac{\dfrac{\dfrac{\begin{array}{c} x \leq y, \\ f(x) < 1 \vdash \\ 0 = f(y), \\ f(y) \prec 0 \end{array} \qquad \vdash g(g(x)) \leq g(g(g(x)))}{\begin{array}{c} f(g(g(x))) < 1 \vdash \\ 0 = f(g(g(g(x)))), \\ f(g(g(g(x)))) < 0 \end{array}} \qquad f(g(g(g(x)))) < 0 \vdash}{\begin{array}{c} f(g(g(x))) < 1 \vdash \\ 0 = f(g(g(g(x)))) \end{array}} \qquad \begin{array}{c} 0 = f(g(g(g(x)))), \\ 0 = f(g(g(x))) \vdash \end{array}}{\begin{array}{c} f(g(g(x))) < 1, \\ 0 = f(g(g(x))) \vdash \\ (28) \end{array}}$$

$$
\dfrac{
\dfrac{
\begin{array}{c}(28)\\ f(g(g(x)))<1,\\ 0=f(g(g(x)))\vdash\end{array}
\qquad
\dfrac{
\vdash g(g(x))\leq g(g(x))
\qquad
\begin{array}{c}x\leq y,\\ f(x)<1\vdash\\ 0=f(y),\\ f(y)<0\end{array}
}{
\begin{array}{c}f(g(g(x)))<1\vdash\\ 0=f(g(g(x))),\\ f(g(g(x)))<0\end{array}
}
}{
\begin{array}{c}f(g(g(x)))<1\\ \vdash f(g(g(x)))<0\end{array}
}
\qquad
f(g(g(x)))<0\vdash
}{
\begin{array}{c}f(g(g(x)))<1\vdash\\ (24)\end{array}
}
$$

$$
\dfrac{
\dfrac{
\dfrac{
\begin{array}{c}x\leq y,\\ f(x)<1\vdash\\ 0=f(y),\\ f(y)<0\end{array}
\qquad
\vdash g(x)\leq g(g(x))
}{
\begin{array}{c}f(g(x))<1\vdash\\ 0=f(g(g(x))),\\ f(g(g(x)))<0\end{array}
}
\qquad
f(g(g(x)))<0\vdash
}{
\begin{array}{c}f(g(x))<1\vdash\\ 0=f(g(g(x)))\end{array}
}
\qquad
\begin{array}{c}0=f(g(g(x))),\\ 0=f(g(x))\vdash\end{array}
}{
\begin{array}{c}f(g(x))<1,\\ 0=f(g(x))\vdash\\ (27)\end{array}
}
$$

$$
\dfrac{
\dfrac{
\begin{array}{c}(27)\\ f(g(x))<1,\\ 0=f(g(x))\vdash\end{array}
\qquad
\dfrac{
\begin{array}{c}x\leq y,\\ f(x)<1\vdash\\ 0=f(y),\\ f(y)<0\end{array}
\qquad
\vdash g(x)\leq g(x)
}{
\begin{array}{c}f(g(x))<1\vdash\\ 0=f(g(x)),\\ f(g(x))<0\end{array}
}
}{
\begin{array}{c}f(g(x))<1\\ \vdash f(g(x))<0\end{array}
}
\qquad
f(g(x))<0\vdash
}{
\begin{array}{c}f(g(x))<1\vdash\\ (22)\end{array}
}
$$

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}x \leq y,\\ f(x) < 2 \vdash\\ 1 = f(y),\\ f(y) < 1\end{array} \qquad \vdash g(g(x)) \leq g(g(g(x)))}{\begin{array}{c}f(g(g(x))) < 2 \vdash\\ 1 = f(g(g(g(x)))),\\ f(g(g(g(x)))) < 1\end{array}} \qquad \begin{array}{c}(26)\\ f(g(g(g(x)))) < 1 \vdash\end{array}}{\begin{array}{c}f(g(g(x))) < 2 \vdash\\ 1 = f(g(g(g(x))))\end{array}} \qquad \begin{array}{c}1 = f(g(g(g(x)))),\\ 1 = f(g(g(x))) \vdash\end{array}}{\begin{array}{c}f(g(g(x))) < 2,\\ 1 = f(g(g(x))) \vdash\\ (25)\end{array}}$$

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}x \leq y,\\ (x) < 2 \vdash\\ 1 = f(y),\\ f(y) < 1\end{array} \qquad \vdash g(g(x)) \leq g(g(x))}{\begin{array}{c}f(g(g(x))) < 2 \vdash\\ 1 = f(g(g(x))),\\ f(g(g(x))) < 1\end{array}} \qquad \begin{array}{c}(25)\\ f(g(g(x))) < 2,\\ 1 = f(g(g(x))) \vdash\end{array}}{\begin{array}{c}f(g(g(x))) < 2\\ \vdash f(g(g(x))) < 1\end{array}} \qquad \begin{array}{c}(24)\\ f(g(g(x))) < 1 \vdash\end{array}}{\begin{array}{c}f(g(g(x))) < 2 \vdash\\ (20)\end{array}}$$

$$\dfrac{\dfrac{\begin{array}{c}x \leq y,\\ f(x) < 2 \vdash\\ 1 = f(y),\\ f(y) < 1\end{array} \qquad \vdash g(x) \leq g(g(x))}{\begin{array}{c}f(g(x)) < 2 \vdash\\ 1 = f(g(g(x))),\\ f(g(g(x))) < 1\end{array}} \qquad \begin{array}{c}(24)\\ f(g(g(x))) < 1 \vdash\end{array}}{\begin{array}{c}f(g(x)) < 2 \vdash\\ 1 = f(g(g(x)))\\ (23)\end{array}}$$

$$\dfrac{\dfrac{\begin{array}{c}x \leq y,\\ f(x) < 2 \vdash\\ 1 = f(y),\\ f(y) < 1\end{array} \qquad \vdash g(x) \leq g(x)}{\begin{array}{c}f(g(x)) < 2 \vdash\\ 1 = f(g(x)),\\ f(g(x)) < 1\end{array}} \qquad \dfrac{\begin{array}{c}(23)\\ f(g(x)) < 2 \vdash\\ 1 = f(g(g(x)))\end{array} \qquad \begin{array}{c}1 = f(g(g(x))),\\ 1 = f(g(x)) \vdash\end{array}}{\begin{array}{c}f(g(x)) < 2,\\ 1 = f(g(x)) \vdash\end{array}}}{\begin{array}{c}f(g(x)) < 2\\ \vdash f(g(x)) < 1\\ (21)\end{array}}$$

$$\dfrac{\begin{array}{c}(22)\\ f(g(x)) < 1 \vdash\end{array} \qquad \begin{array}{c}(21)\\ f(g(x)) < 2\\ \vdash f(g(x)) < 1\end{array}}{\begin{array}{c}f(g(x)) < 2 \vdash\\ (18)\end{array}}$$

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}x \leq y,\\ f(x) < 3 \vdash\\ 2 = f(y),\\ f(y) < 2\end{array} \qquad \vdash g(x) \leq g(g(x))}{\begin{array}{c}f(g(x)) < 3 \vdash\\ 2 = f(g(g(x))),\\ f(g(g(x))) < 2\end{array}} \qquad \begin{array}{c}(20)\\ f(g(g(x))) < 2 \vdash\end{array}}{\begin{array}{c}f(g(x)) < 3 \vdash\\ 2 = f(g(g(x)))\end{array}} \qquad \begin{array}{c}2 = f(g(g(x))),\\ 2 = f(g(x)) \vdash\end{array}}{\begin{array}{c}f(g(x)) < 3,\\ 2 = f(g(x)) \vdash\\ (19)\end{array}}$$

$$\dfrac{\begin{array}{c}(19)\\ f(g(x)) < 3,\\ 2 = f(g(x)) \vdash\end{array} \qquad \dfrac{\begin{array}{c}x \leq y,\\ f(x) < 3 \vdash\\ 2 = f(y),\\ f(y) < 2\end{array} \qquad \vdash g(x) \leq g(x)}{\begin{array}{c}f(g(x)) < 3 \vdash\\ 2 = f(g(x)),\\ f(g(x)) < 2\end{array}}}{\begin{array}{c}f(g(x)) < 3 \vdash\\ f(g(x)) < 2\\ (17)\end{array}}$$

$$\dfrac{\begin{array}{c}(17)\\ f(g(x)) < 3 \vdash\\ f(g(x)) < 2\end{array} \qquad \begin{array}{c}(18)\\ f(g(x)) < 2 \vdash\end{array}}{\begin{array}{c}f(g(x)) < 3 \vdash\\ (16)\end{array}}$$

$$\dfrac{\dfrac{\dfrac{\begin{array}{c} x\le y,\\ f(x)<1\vdash\\ 0=f(y),\\ f(y)<0\end{array} \qquad \vdash g(g(x))\le g(g(g(x)))}{\begin{array}{c} f(g(g(x)))<1\vdash\\ 0=f(g(g(g(x)))),\\ f(g(g(g(x))))<0\end{array}} \qquad f(g(g(g(x))))<0\vdash}{\begin{array}{c} f(g(g(x)))<1\vdash\\ 0=f(g(g(g(x))))\end{array}} \qquad \begin{array}{c}0=f(g(g(g(x)))),\\ 0=f(g(g(x)))\vdash\end{array}}{\begin{array}{c} f(g(g(x)))<1,\\ 0=f(g(g(x)))\vdash\\ (1)\end{array}}$$

$$\dfrac{\dfrac{\begin{array}{c}(1)\\ f(g(g(x)))<1,\\ 0=f(g(g(x)))\vdash\end{array} \qquad \dfrac{\vdash g(g(x))\le g(g(x)) \qquad \begin{array}{c} x\le y,\\ f(x)<1\vdash\\ 0=f(y),\\ f(y)<0\end{array}}{\begin{array}{c} f(g(g(x)))<1\vdash\\ 0=f(g(g(x))),\\ f(g(g(x)))<0\end{array}}}{\begin{array}{c} f(g(g(x)))<1\\ \vdash f(g(g(x)))<0\end{array}} \qquad f(g(g(x)))<0\vdash}{\begin{array}{c} f(g(g(x)))<1\vdash\\ (2)\end{array}}$$

$$\dfrac{\dfrac{\dfrac{\begin{array}{c} x\le y,\\ f(x)<1\vdash\\ 0=f(y),\\ f(y)<0\end{array} \qquad \vdash g(x)\le g(g(x))}{\begin{array}{c} f(g(x))<1\vdash\\ 0=f(g(g(x))),\\ f(g(g(x)))<0\end{array}} \qquad f(g(g(x)))<0\vdash}{\begin{array}{c} f(g(x))<1\vdash\\ 0=f(g(g(x)))\end{array}} \qquad \begin{array}{c}0=f(g(g(x))),\\ 0=f(g(x))\vdash\end{array}}{\begin{array}{c} f(g(x))<1,\\ 0=f(g(x))\vdash\\ (3)\end{array}}$$

$$\dfrac{\dfrac{\begin{array}{c}(3)\\ f(g(x))<1,\\ 0=f(g(x))\vdash\end{array} \qquad \dfrac{\vdash g(x)\le g(x) \qquad \begin{array}{c}x\le y,\\ f(x)<1\vdash\\ 0=f(y),\\ f(y)<0\end{array}}{\begin{array}{c}f(g(x))<1\vdash\\ 0=f(g(x)),\\ f(g(x))<0\end{array}}}{\begin{array}{c}f(g(x))<1\\ \vdash f(g(x))<0\end{array}} \qquad f(g(x))<0\vdash}{\begin{array}{c}f(g(x))<1\vdash\\ (4)\end{array}}$$

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}x\le y,\\ f(x)<1\vdash\\ 0=f(y),\\ f(y)<0\end{array} \qquad \vdash x\le g(x)}{\begin{array}{c}f(x)<1\vdash\\ 0=f(g(x)),\\ f(g(x))<0\end{array}} \qquad f(g(x))<0\vdash}{\begin{array}{c}f(x)<1\vdash\\ 0=f(g(x))\end{array}} \qquad \begin{array}{c}0=f(g(x)),\\ 0=f(x)\vdash\end{array}}{\begin{array}{c}f(x)<1,\\ 0=f(x)\vdash\\ (13)\end{array}}$$

$$\dfrac{\dfrac{\begin{array}{c}(13)\\ f(x)<1,\\ 0=f(x)\vdash\end{array} \qquad \dfrac{\begin{array}{c}x\le y,\\ f(x)<1\vdash\\ 0=f(y),\\ f(y)<0\end{array} \qquad \vdash x\le x}{\begin{array}{c}f(x)<1\vdash\\ 0=f(x),\\ f(x)<0\end{array}}}{\begin{array}{c}f(x)<1\\ \vdash f(x)<0\end{array}} \qquad f(x)<0\vdash}{\begin{array}{c}f(x)<1\vdash\\ (5)\end{array}}$$

$$
\dfrac{
 \dfrac{
  \dfrac{\begin{array}{c}x \le y,\\ f(x) < 2 \vdash\\ 1 = f(y),\\ f(y) < 1\end{array} \qquad \vdash g(x) \le g(g(x))}
  {\begin{array}{c}f(g(x)) < 2 \vdash\\ 1 = f(g(g(x))),\\ f(g(g(x))) < 1\end{array}}
  \qquad \begin{array}{c}(2)\\ f(g(g(x))) < 1 \vdash\end{array}}
 {\begin{array}{c}f(g(x)) < 2 \vdash\\ 1 = f(g(g(x)))\end{array}}
 \qquad \begin{array}{c}1 = f(g(g(x))),\\ 1 = f(g(x)) \vdash\end{array}}
{\begin{array}{c}f(g(x)) < 2,\\ 1 = f(g(x)) \vdash\\ (12)\end{array}}
$$

$$
\dfrac{
 \dfrac{
  \dfrac{\begin{array}{c}x \le y,\\ (x) < 2 \vdash\\ 1 = f(y),\\ f(y) < 1\end{array} \qquad \vdash g(x) \le g(x)}
  {\begin{array}{c}f(g(x)) < 2 \vdash\\ 1 = f(g(x)),\\ f(g(x)) < 1\end{array}}
  \qquad \begin{array}{c}(12)\\ f(g(x)) < 2,\\ 1 = f(g(x)) \vdash\end{array}}
 {\begin{array}{c}f(g(x)) < 2\\ \vdash f(g(x)) < 1\end{array}}
 \qquad \begin{array}{c}(4)\\ f(g(x)) < 1 \vdash\end{array}}
{\begin{array}{c}f(g(x)) < 2 \vdash\\ (6)\end{array}}
$$

$$
\dfrac{
 \dfrac{\begin{array}{c}x \le y,\\ f(x) < 2 \vdash\\ 1 = f(y),\\ f(y) < 1\end{array} \qquad \vdash x \le g(x)}
 {\begin{array}{c}f(x) < 2 \vdash\\ 1 = f(g(x)),\\ f(g(x)) < 1\end{array}}
 \qquad \begin{array}{c}(4)\\ f(g(x)) < 1 \vdash\end{array}}
{\begin{array}{c}f(x) < 2 \vdash\\ 1 = f(g(x))\\ (9)\end{array}}
$$

$$
\dfrac{
 \dfrac{\begin{array}{c}x \le y,\\ f(x) < 2 \vdash\\ 1 = f(y),\\ f(y) < 1\end{array} \qquad \vdash x \le x}
 {\begin{array}{c}f(x) < 2 \vdash\\ 1 = f(x),\\ f(x) < 1\end{array}}
 \qquad
 \dfrac{\begin{array}{c}(9)\\ f(x) < 2 \vdash\\ 1 = f(g(x))\end{array} \qquad \begin{array}{c}1 = f(g(x)),\\ 1 = f(x) \vdash\end{array}}
 {\begin{array}{c}f(x) < 2,\\ 1 = f(x) \vdash\end{array}}}
{\begin{array}{c}f(x) < 2\\ \vdash f(x) < 1\\ (10)\end{array}}
$$

$$\dfrac{\begin{array}{c}(5)\\ f(x)<1\vdash\end{array} \qquad \begin{array}{c}(10)\\ f(x)<2\\ \vdash f(x)<1\end{array}}{\begin{array}{c}f(x)<2\vdash\\ (7)\end{array}}$$

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}x\leq y,\\ f(x)<3\vdash\\ 2=f(y),\\ f(y)<2\end{array} \qquad \vdash x\leq g(x)}{\begin{array}{c}f(x)<3\vdash\\ 2=f(g(x)),\\ f(g(x))<2\end{array}} \qquad \begin{array}{c}(6)\\ f(g(x))<2\vdash\end{array}}{\begin{array}{c}f(x)<3\vdash\\ 2=f(g(x))\end{array}} \qquad \begin{array}{c}2=f(g(x)),\\ 2=f(x)\vdash\end{array}}{\begin{array}{c}f(x)<3,\\ 2=f(x)\vdash\\ (11)\end{array}}$$

$$\dfrac{\begin{array}{c}(11)\\ f(x)<3,\\ 2=f(x)\vdash\end{array} \qquad \dfrac{\begin{array}{c}x\leq y,\\ f(x)<3\vdash\\ 2=f(y),\\ f(y)<2\end{array} \qquad \vdash x\leq x}{\begin{array}{c}f(x)<3\vdash\\ 2=f(x),\\ f(x)<2\end{array}}}{\begin{array}{c}f(x)<3\vdash\\ f(x)<2\\ (8)\end{array}}$$

$$\dfrac{\begin{array}{c}(8)\\ f(x)<3\vdash\\ f(x)<2\end{array} \qquad \begin{array}{c}(7)\\ f(x)<2\vdash\end{array}}{\begin{array}{c}f(x)<3\vdash\\ (15)\end{array}}$$

$$\dfrac{\dfrac{\begin{array}{c}3=f(x),\\ 3=f(g(x))\vdash\end{array} \qquad \dfrac{\begin{array}{c}(16)\\ f(g(x))<3\vdash\end{array} \qquad \begin{array}{c}\vdash 3=f(g(x)),\\ f(g(x))<3\end{array}}{\vdash 3=f(g(x))}}{3=f(x)\vdash} \qquad \begin{array}{c}\vdash f(x)<3,\\ 3=f(x)\end{array}}{\begin{array}{c}\vdash f(x)<3\\ (14)\end{array}}$$

$$\dfrac{\begin{array}{c}(8)\\ f(x) < 3 \vdash\\ f(x) < 2\end{array} \qquad \begin{array}{c}(7)\\ f(x) < 2 \vdash\end{array}}{\begin{array}{c}f(x) < 3 \vdash\\ (15)\end{array}}$$

$$\dfrac{\begin{array}{c}(15)\\ f(x) < 3 \vdash\end{array} \qquad \begin{array}{c}(14)\\ \vdash f(x) < 3\end{array}}{\vdash}$$

APPENDIX B

Non-injectivity Assertion: Proofs and Refutations

B.1 LKS Proof of NiA

In this section we provide the formal proof of the NIA statement using the infinity lemmata. We will demarcate the cut ancestors with * and cut-configuration ancestors with **. The colors denote which cut the formulae come from. Also, instead of using $=$ to represent equality in the formal proof we will use $\sim$ being that all equality is occurring over the ω sort. f should be understood as a function mapping the members of the individual sort to the ω sort.

φ

Stepcase of φ

$$\dfrac{\bigvee_{i=0}^{n} f(\gamma) \sim i^{**} \vdash \bigvee_{i=0}^{n} f(\gamma) \sim i^{*} \qquad f(\gamma) \sim n+1^{**} \vdash f(\gamma) \sim n+1^{*}}{\bigvee_{i=0}^{n+1} f(\gamma) \sim i^{**} \vdash \bigvee_{i=0}^{n} f(\gamma) \sim i^{*} , f(\gamma) \sim n+1^{*}}\ \vee : l$$

$$\dfrac{\bigvee_{i=0}^{n+1} f(\gamma) \sim i^{**} \vdash \bigvee_{i=0}^{n} f(\gamma) \sim i^{*} , f(\gamma) \sim n+1^{*}}{\bigvee_{i=0}^{n+1} f(\gamma) \sim i^{**}, f(\gamma) \not\sim n+1^{*} \vdash \bigvee_{i=0}^{n} f(\gamma) \sim i^{*}}\ \neg : l$$

$$\dfrac{max(\alpha,\beta) \leq \gamma^{**} \vdash \alpha \leq \gamma^{*} \qquad \bigvee_{i=0}^{n+1} f(\gamma) \sim i^{**}, f(\gamma) \not\sim n+1^{*} \vdash \bigvee_{i=0}^{n} f(\gamma) \sim i^{*}}{max(\alpha,\beta) \leq \gamma^{**}, \bigvee_{i=0}^{n+1} f(\gamma) \sim i^{**}, f(\gamma) \not\sim n+1^{*} \vdash \alpha \leq \gamma \wedge \bigvee_{i=0}^{n} f(\gamma) \sim i^{*}}\ \wedge : r$$

$$\dfrac{max(\alpha,\beta) \leq \gamma^{**}, \bigvee_{i=0}^{n+1} f(\gamma) \sim i^{**}, f(\gamma) \not\sim n+1^{*} \vdash \alpha \leq \gamma \wedge \bigvee_{i=0}^{n} f(\gamma) \sim i^{*} \qquad max(\alpha,\beta) \leq \gamma^{**} \vdash \beta \leq \gamma^{*}}{max(\alpha,\beta) \leq \gamma^{**}, max(\alpha,\beta) \leq \gamma^{**}, \bigvee_{i=0}^{n+1} f(\gamma) \sim i^{**}, \beta \leq \gamma \rightarrow f(\gamma) \not\sim n+1^{*} \vdash \alpha \leq \gamma \wedge \bigvee_{i=0}^{n} f(\gamma) \sim i^{*}}\ \rightarrow : l$$

$$\dfrac{max(\alpha,\beta) \leq \gamma^{**}, max(\alpha,\beta) \leq \gamma^{**}, \bigvee_{i=0}^{n+1} f(\gamma) \sim i^{**}, \beta \leq \gamma \rightarrow f(\gamma) \not\sim n+1^{*} \vdash \alpha \leq \gamma \wedge \bigvee_{i=0}^{n} f(\gamma) \sim i^{*}}{max(\alpha,\beta) \leq \gamma^{**}, \bigvee_{i=0}^{n+1} f(\gamma) \sim i^{**}, \beta \leq \gamma \rightarrow f(\gamma) \not\sim n+1^{*} \vdash \alpha \leq \gamma \wedge \bigvee_{i=0}^{n} f(\gamma) \sim i^{*}}\ c : l$$

$$\dfrac{max(\alpha,\beta) \leq \gamma^{**}, \bigvee_{i=0}^{n+1} f(\gamma) \sim i^{**}, \beta \leq \gamma \rightarrow f(\gamma) \not\sim n+1^{*} \vdash \alpha \leq \gamma \wedge \bigvee_{i=0}^{n} f(\gamma) \sim i^{*}}{max(\alpha,\beta) \leq \gamma \wedge \bigvee_{i=0}^{n+1} f(\gamma) \sim i^{**}, \beta \leq \gamma \rightarrow f(\gamma) \not\sim n+1^{*} \vdash \alpha \leq \gamma \wedge \bigvee_{i=0}^{n} f(\gamma) \sim i^{*}}\ \wedge : l$$

$$\dfrac{max(\alpha,\beta) \leq \gamma \wedge \bigvee_{i=0}^{n+1} f(\gamma) \sim i^{**}, \beta \leq \gamma \rightarrow f(\gamma) \not\sim n+1^{*} \vdash \alpha \leq \gamma \wedge \bigvee_{i=0}^{n} f(\gamma) \sim i^{*}}{max(\alpha,\beta) \leq \gamma \wedge \bigvee_{i=0}^{n+1} f(\gamma) \sim i^{**}, \forall y(\beta \leq y \rightarrow f(y) \not\sim n+1)^{*} \vdash \alpha \leq \gamma \wedge \bigvee_{i=0}^{n} f(\gamma) \sim i^{*}}\ \forall : l$$

$$\dfrac{max(\alpha,\beta) \leq \gamma \wedge \bigvee_{i=0}^{n+1} f(\gamma) \sim i^{**}, \forall y(\beta \leq y \rightarrow f(y) \not\sim n+1)^{*} \vdash \alpha \leq \gamma \wedge \bigvee_{i=0}^{n} f(\gamma) \sim i^{*}}{max(\alpha,\beta) \leq \gamma \wedge \bigvee_{i=0}^{n+1} f(\gamma) \sim i^{**}, \forall y(\beta \leq y \rightarrow f(y) \not\sim n+1)^{*} \vdash \exists y(\alpha \leq y \wedge \bigvee_{i=0}^{n} f(y) \sim i)^{*}}\ \exists : r$$

$$\dfrac{max(\alpha,\beta) \leq \gamma \wedge \bigvee_{i=0}^{n+1} f(\gamma) \sim i^{**}, \forall y(\beta \leq y \rightarrow f(y) \not\sim n+1)^{*} \vdash \exists y(\alpha \leq y \wedge \bigvee_{i=0}^{n} f(y) \sim i)^{*}}{\exists y(max(\alpha,\beta) \leq y \wedge \bigvee_{i=0}^{n+1} f(y) \sim i)^{**}, \forall y(\beta \leq y \rightarrow f(y) \not\sim n+1)^{*} \vdash \exists y(\alpha \leq y \wedge \bigvee_{i=0}^{n} f(y) \sim i)^{*}}\ \exists : l$$

$$\dfrac{\exists y(max(\alpha,\beta) \leq y \wedge \bigvee_{i=0}^{n+1} f(y) \sim i)^{**}, \forall y(\beta \leq y \rightarrow f(y) \not\sim n+1)^{*} \vdash \exists y(\alpha \leq y \wedge \bigvee_{i=0}^{n} f(y) \sim i)^{*}}{\forall x \exists y(x \leq y \wedge \bigvee_{i=0}^{n+1} f(y) \sim i)^{**}, \forall y(\beta \leq y \rightarrow f(y) \not\sim n+1)^{*} \vdash \exists y(\alpha \leq y \wedge \bigvee_{i=0}^{n} f(y) \sim i)^{*}}\ \forall : l$$

$$\dfrac{\forall x \exists y(x \leq y \wedge \bigvee_{i=0}^{n+1} f(y) \sim i)^{**}, \forall y(\beta \leq y \rightarrow f(y) \not\sim n+1)^{*} \vdash \exists y(\alpha \leq y \wedge \bigvee_{i=0}^{n} f(y) \sim i)^{*}}{\forall x \exists y(x \leq y \wedge \bigvee_{i=0}^{n+1} f(y) \sim i)^{**}, \exists x \forall y(x \leq y \rightarrow f(y) \not\sim n+1)^{*} \vdash \exists y(\alpha \leq y \wedge \bigvee_{i=0}^{n} f(y) \sim i)^{*}}\ \exists : l$$

$$\dfrac{\forall x \exists y(x \leq y \wedge \bigvee_{i=0}^{n+1} f(y) \sim i)^{**}, \exists x \forall y(x \leq y \rightarrow f(y) \not\sim n+1)^{*} \vdash \exists y(\alpha \leq y \wedge \bigvee_{i=0}^{n} f(y) \sim i)^{*}}{\forall x \exists y(x \leq y \wedge \bigvee_{i=0}^{n+1} f(y) \sim i)^{**}, \exists x \forall y(x \leq y \rightarrow f(y) \not\sim n+1) \vdash \forall x \exists y(x \leq y \wedge \bigvee_{i=0}^{n} f(y) \sim i)}\ \forall : r$$

$$\dfrac{\forall x \exists y(x \leq y \wedge \bigvee_{i=0}^{n+1} f(y) \sim i)^{**}, \exists x \forall y(x \leq y \rightarrow f(y) \not\sim n+1) \vdash \forall x \exists y(x \leq y \wedge \bigvee_{i=0}^{n} f(y) \sim i)}{\forall x \exists y(x \leq y \wedge \bigvee_{i=0}^{n+1} f(y) \sim i)^{**}, \exists x \forall y(x \leq y \rightarrow f(y) \not\sim n+1)^{*} \vdash \forall x \exists y(x \leq y \wedge \bigvee_{i=0}^{n} f(y) \sim i)^{*}}$$

$$\vdots$$

$$\cfrac{\begin{array}{c}\vdots\\ \forall x\exists y(x\leq y\land\bigvee_{i=0}^{n+1}f(y)\sim i)^{**},\\ \exists x\forall y(x\leq y\to f(y)\not\sim n+1)^{*}\vdash\\ \forall x\exists y(x\leq y\land\bigvee_{i=0}^{n}f(y)\sim i)^{*}\end{array}\qquad\begin{array}{c}\exists x\forall y(x\leq y\to f(y)\neq n+1)^{*}\vdash\\ \forall x\exists y(x\leq y\land f(y)=n+1)^{*}\end{array}}{\begin{array}{c}\forall x\exists y(x\leq y\land\bigvee_{i=0}^{n+1}f(y)\sim i)^{**}\vdash\\ \forall x\exists y(x\leq y\land\bigvee_{i=0}^{n}f(y)\sim i)^{*},\\ \forall x\exists y(x\leq y\land f(y)=n+1)^{*}\\ \vdots\end{array}}\;cut$$

$$\cfrac{\begin{array}{c}\vdots\\ \forall x\exists y(x\leq y\land\bigvee_{i=0}^{n+1}f(y)\sim i)^{**}\vdash\\ \forall x\exists y(x\leq y\land\bigvee_{i=0}^{n}f(y)\sim i)^{*},\\ \forall x\exists y(x\leq y\land f(y)=n+1)^{*}\end{array}\qquad\begin{array}{c}\varphi(n)\\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\ \forall x\exists y(x\leq y\land\bigvee_{i=0}^{n}f(y)\sim i)^{*}\vdash\\ \exists p\exists q(p<q\land f(p)\sim f(q))\end{array}}{\begin{array}{c}\forall x\exists y(x\leq y\land\bigvee_{i=0}^{n+1}f(y)\sim i)^{**}\vdash\\ \exists p\exists q(p<q\land f(p)\sim f(q)),\\ \forall x\exists y(x\leq y\land f(y)=n+1)^{*}\\ \vdots\end{array}}\;cut$$

$$\cfrac{\cfrac{\begin{array}{c}\vdots\\ \forall x\exists y(x\leq y\land\bigvee_{i=0}^{n+1}f(y)\sim i)^{**}\vdash\\ \exists p\exists q(p<q\land f(p)\sim f(q)),\\ \forall x\exists y(x\leq y\land f(y)=n+1)^{*}\end{array}\qquad\begin{array}{c}\mu(n+1)\\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\ \forall x\exists y(x\leq y\land f(y)=n+1)^{*}\vdash\\ \exists p\exists q(p<q\land f(p)\sim f(q))\end{array}}{\begin{array}{c}\forall x\exists y(x\leq y\land\bigvee_{i=0}^{n+1}f(y)\sim i)^{**}\vdash\\ \exists p\exists q(p<q\land f(p)\sim f(q)),\\ \exists p\exists q(p<q\land f(p)\sim f(q))\end{array}}\;cut}{\begin{array}{c}\forall x\exists y(x\leq y\land\bigvee_{i=0}^{n+1}f(y)\sim i)^{**}\vdash\\ \exists p\exists q(p<q\land f(p)\sim f(q))\end{array}}\;c:r$$

Basecase of φ

$$\begin{array}{c}\mu(0)\\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\ \forall x\exists y(x\leq y\land f(y)\sim 0)\vdash\\ \exists p\exists q(p<q\land f(p)\sim f(q))\end{array}$$

ω

Stepcase of ω

$$
\begin{prooftree}
\AxiomC{$\varphi(n+1)$}
\dottedLine
\UnaryInfC{$\forall x \exists y (x \leq y \wedge \bigvee_{i=0}^{n+1} f(y) \sim i)^{*} \vdash \exists p \exists q (p < q \wedge f(p) \sim f(q))$}
\AxiomC{$\vdash \alpha \leq \alpha^{*}$}
\AxiomC{$\bigvee_{i=0}^{n+1} f(\alpha) \sim i \vdash \bigvee_{i=0}^{n+1} f(\alpha) \sim i^{*}$}
\RightLabel{$\wedge : r$}
\BinaryInfC{$\bigvee_{i=0}^{n+1} f(\alpha) \sim i \vdash \alpha \leq \alpha \wedge \bigvee_{i=0}^{n+1} f(\alpha) \sim i^{*}$}
\RightLabel{$\exists : r$}
\UnaryInfC{$\bigvee_{i=0}^{n+1} f(\alpha) \sim i \vdash \exists y (\alpha \leq y \wedge \bigvee_{i=0}^{n+1} f(y) \sim i)^{*}$}
\RightLabel{$\forall : l$}
\UnaryInfC{$\forall x \bigvee_{i=0}^{n+1} f(x) \sim i \vdash \exists y (\alpha \leq y \wedge \bigvee_{i=0}^{n+1} f(y) \sim i)^{*}$}
\RightLabel{$\forall : r$}
\UnaryInfC{$\forall x \bigvee_{i=0}^{n+1} f(x) \sim i \vdash \exists y (\alpha \leq y \wedge \bigvee_{i=0}^{n+1} f(y) \sim i)^{*}$}
\RightLabel{$\forall : r$}
\UnaryInfC{$\forall x \bigvee_{i=0}^{n+1} f(x) \sim i \vdash \forall x \exists y (x \leq y \wedge \bigvee_{i=0}^{n+1} f(y) \sim i)^{*}$}
\RightLabel{cut}
\BinaryInfC{$\forall x \bigvee_{i=0}^{n+1} f(x) \sim i \vdash \exists p \exists q (p < q \wedge f(p) \sim f(q))$}
\end{prooftree}
$$

Basecase of ω

$$
\begin{prooftree}
\AxiomC{$\vdash \alpha \leq \alpha^{*}$}
\AxiomC{$f(\alpha) \sim 0 \vdash f(\alpha) \sim 0^{*}$}
\RightLabel{$\wedge : r$}
\BinaryInfC{$f(\alpha) \sim 0 \vdash \alpha \leq \alpha \wedge f(\alpha) \sim 0^{*}$}
\RightLabel{$\exists : r$}
\UnaryInfC{$f(\alpha) \sim 0 \vdash \exists y (\alpha \leq y \wedge f(y) \sim 0)^{*}$}
\RightLabel{$\forall : l$}
\UnaryInfC{$\forall x f(x) \sim 0 \vdash \exists y (\alpha \leq y \wedge f(y) \sim 0)^{*}$}
\RightLabel{$\forall : r$}
\UnaryInfC{$\forall x f(x) \sim 0 \vdash \forall x \exists y (x \leq y \wedge f(y) \sim 0)^{*}$}
\AxiomC{$\mu(0)$}
\dottedLine
\UnaryInfC{$\forall x \exists y (x \leq y \wedge f(y) = 0)^{*} \vdash \exists p \exists q (p < q \wedge f(p) \sim f(q))$}
\RightLabel{cut}
\BinaryInfC{$\forall x f(x) \sim 0 \vdash \exists p \exists q (p < q \wedge f(p) \sim f(q))$}
\end{prooftree}
$$

μ

Stepcase/Basecase of μ

$$
\begin{array}{cl}
s(\beta) \leq \alpha^* \vdash \beta < \alpha \qquad f(\beta) = i, f(\alpha) = i^* \vdash f(\beta) \sim f(\alpha) & \\
\hline
f(\beta) = i^* , s(\beta) \leq \alpha^*, f(\alpha) = i \vdash \beta < \alpha \wedge f(\beta) \sim f(\alpha) & cut \\
\hline
f(\beta) = i^* , (s(\beta) \leq \alpha \wedge f(\alpha) = i)^* \vdash \beta < \alpha \wedge f(\beta) \sim f(\alpha) & \wedge : r \\
\hline
(0 \leq \beta \wedge f(\beta) = i)^* , (s(\beta) \leq \alpha \wedge f(\alpha) = i)^* \vdash \beta < \alpha \wedge f(\beta) \sim f(\alpha) & \wedge : r \\
\hline
(0 \leq \beta \wedge f(\beta) = i)^* , (s(\beta) \leq \alpha \wedge f(\alpha) = i)^* \vdash \exists q(\beta < q \wedge f(\beta) \sim f(q)) & \exists : l \\
\hline
(0 \leq \beta \wedge f(\beta) = i)^* , (s(\beta) \leq \alpha \wedge f(\alpha) = i)^* \vdash \exists p \exists q(p < q \wedge f(p) \sim f(q)) & \exists : l \\
\hline
(0 \leq \beta \wedge f(\beta) = i)^* , \exists y(\beta \leq y \wedge f(y) = i)^* \vdash \exists p \exists q(p < q \wedge f(p) \sim f(q)) & \exists : r \\
\hline
(0 \leq \beta \wedge f(\beta) = i)^* , \forall x \exists y(x \leq y \wedge f(y) = i)^* \vdash \exists p \exists q(p < q \wedge f(p) \sim f(q)) & \forall : r \\
\hline
\exists y(0 \leq y \wedge f(y) = i)^* , \forall x \exists y(x \leq y \wedge f(y) = i)^* \vdash \exists p \exists q(p < q \wedge f(p) \sim f(q)) & \exists : r \\
\hline
\forall x \exists y(x \leq y \wedge f(y) = i)^* , \forall x \exists y(x \leq y \wedge f(y) = i)^* \vdash \exists p \exists q(p < q \wedge f(p) \sim f(q)) & \forall : r \\
\hline
\forall x \exists y(x \leq y \wedge f(y) = i)^* \vdash \exists p \exists q(p < q \wedge f(p) \sim f(q)) & w : r
\end{array}
$$

B.2 Refutation of $C(n)$ for $n = 2$

In this section we provide a fully unrolled resolution refutation of the clause set $C(2)$. The numbering refers to how the components connect together and the start of the proof can be found on the last page of this section.

$$\dfrac{\begin{array}{c}(11)\\ f(\mathbf{y_1}) \sim 1, f(\mathbf{y_2}) \sim 2 \vdash \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 0\end{array} \qquad \begin{array}{c}(4)\\ f(\mathbf{x}) \sim 0,\\ f(\mathbf{z}) \sim 1,\\ f(\mathbf{y}) \sim 2 \vdash\end{array}}{f(\mathbf{x}) \sim 1, f(\mathbf{y}) \sim 2 \vdash}$$

$$\dfrac{\dfrac{\begin{array}{c}(13)\\ f(\mathbf{y_2}) \sim 2 \vdash \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 0,\\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 1\end{array} \qquad \dfrac{\begin{array}{c}(10)\\ f(\mathbf{y_0}) \sim 0, f(\mathbf{y_2}) \sim 2 \vdash \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 1\end{array} \qquad \begin{array}{c}(4)\\ f(\mathbf{x}) \sim 0,\\ f(\mathbf{z}) \sim 1,\\ f(\mathbf{y}) \sim 2 \vdash\end{array}}{f(\mathbf{x}) \sim 0, f(\mathbf{y}) \sim 2 \vdash}}{\begin{array}{c}f(\mathbf{y_2}) \sim 2 \vdash \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 1\end{array}} \qquad f(\mathbf{x}) \sim 1, f(\mathbf{y}) \sim 2 \vdash}{f(\mathbf{y}) \sim 2 \vdash}$$

(1)

$$\dfrac{\begin{array}{c}(11)\\ f(\mathbf{y_1}) \sim 1, f(\mathbf{y_2}) \sim 2 \vdash \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 0\end{array} \qquad \begin{array}{c}(4)\\ f(\mathbf{x}) \sim 0,\\ f(\mathbf{z}) \sim 1,\\ f(\mathbf{y}) \sim 2 \vdash\end{array}}{f(\mathbf{y}) \sim 1, f(\mathbf{x}) \sim 2 \vdash}$$

$$\dfrac{\dfrac{\begin{array}{c}(12)\\ f(\mathbf{y_1}) \sim 1 \vdash \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 0,\\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 2\end{array} \qquad \dfrac{\begin{array}{c}(9)\\ f(\mathbf{y_0}) \sim 0, f(\mathbf{y_1}) \sim 1 \vdash \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 2\end{array} \qquad \begin{array}{c}(4)\\ f(\mathbf{x}) \sim 0,\\ f(\mathbf{z}) \sim 1,\\ f(\mathbf{y}) \sim 2 \vdash\end{array}}{f(\mathbf{x}) \sim 0, f(\mathbf{y}) \sim 1 \vdash}}{\begin{array}{c}f(\mathbf{y_1}) \sim 1 \vdash \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 2\end{array}} \qquad f(\mathbf{y}) \sim 1, f(\mathbf{x}) \sim 2 \vdash}{f(\mathbf{y}) \sim 1 \vdash}$$

(2)

$$\dfrac{\begin{array}{c}(10)\\ f(\mathbf{y_0})\sim 0, f(\mathbf{y_2})\sim 2\vdash\\ f(max(max(max(s(\mathbf{y_0}),s(\mathbf{y_0})),s(\mathbf{y_1})),s(\mathbf{y_2})))\sim 1\end{array}\qquad \begin{array}{c}(4)\\ f(\mathbf{x})\sim 0,\\ f(\mathbf{z})\sim 1,\\ f(\mathbf{y})\sim 2\vdash\end{array}}{f(\mathbf{y})\sim 0, f(\mathbf{x})\sim 2\vdash}$$

$$\dfrac{\dfrac{\begin{array}{c}(7)\\ f(\mathbf{y_0})\sim 0\vdash\\ f(max(max(max(s(\mathbf{y_0}),s(\mathbf{y_0})),s(\mathbf{y_1})),s(\mathbf{y_2})))\sim 1,\\ f(max(max(max(s(\mathbf{y_0}),s(\mathbf{y_0})),s(\mathbf{y_1})),s(\mathbf{y_2})))\sim 2\end{array}\qquad \dfrac{\begin{array}{c}(9)\\ f(\mathbf{y_0})\sim 0, f(\mathbf{y_1})\sim 1\vdash\\ f(max(max(max(s(\mathbf{y_0}),s(\mathbf{y_0})),s(\mathbf{y_1})),s(\mathbf{y_2})))\sim 2\end{array}\qquad \begin{array}{c}(4)\\ f(\mathbf{x})\sim 0,\\ f(\mathbf{z})\sim 1,\\ f(\mathbf{y})\sim 2\vdash\end{array}}{f(\mathbf{x})\sim 0, f(\mathbf{y})\sim 1\vdash}}{\begin{array}{c}f(\mathbf{y_0})\sim 0\vdash\\ f(max(max(max(s(\mathbf{y_0}),s(\mathbf{y_0})),s(\mathbf{y_1})),s(\mathbf{y_2})))\sim 2\end{array}}\qquad f(\mathbf{y})\sim 0, f(\mathbf{x})\sim 2\vdash}{\begin{array}{c}f(\mathbf{y})\sim 0\vdash\\ (3)\end{array}}$$

$$\dfrac{\dfrac{\begin{array}{c} f(\mathbf{y}) \sim 2, \\ f(\mathbf{x}) \sim 2, \\ s(y) \leq \mathbf{x} \vdash \end{array} \quad \begin{array}{c} (6) \\ \vdash s(\mathbf{y_1}) \leq max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2})) \end{array}}{\begin{array}{c} f(\mathbf{y_1}) \sim 1, \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 1 \vdash \end{array}} \quad \begin{array}{c} (7) \\ f(\mathbf{y_0}) \sim 0 \vdash \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 1, \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 2 \end{array}}{\begin{array}{c} f(\mathbf{y_0}) \sim 0, f(\mathbf{y_1}) \sim 1 \vdash \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 2 \\ (9) \end{array}}$$

$$\dfrac{\dfrac{\begin{array}{c} f(\mathbf{y}) \sim 2, \\ f(\mathbf{x}) \sim 2, \\ s(y) \leq \mathbf{x} \vdash \end{array} \quad \begin{array}{c} (5) \\ \vdash s(\mathbf{y_2}) \leq max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2})) \end{array}}{\begin{array}{c} f(\mathbf{y_2}) \sim 2, \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 2 \vdash \end{array}} \quad \begin{array}{c} (9) \\ f(\mathbf{y_0}) \sim 0, f(\mathbf{y_1}) \sim 1 \vdash \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 2 \end{array}}{\begin{array}{c} f(\mathbf{x}) \sim 0, \\ f(\mathbf{z}) \sim 1, \\ f(\mathbf{y}) \sim 2 \vdash \\ (4) \end{array}}$$

$$\frac{\vdash max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2})) \leq max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2})) \qquad max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2})) \leq max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2})) \vdash s(\mathbf{y_2}) \leq max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))}{\vdash s(\mathbf{y_2}) \leq max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))}$$

(5)

$$\dfrac{\dfrac{\begin{array}{c}\vdash max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2})) \leq \\ max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))\end{array} \qquad \begin{array}{c} max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2})) \leq \\ max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2})) \vdash \\ max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})) \leq \\ max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))\end{array}}{\begin{array}{c}\vdash max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})) \leq \\ max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))\end{array}} \qquad \begin{array}{c} max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})) \leq \\ max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2})) \vdash \\ s(\mathbf{y_1}) \leq \\ max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))\end{array}}{\vdash s(\mathbf{y_1}) \leq max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))}$$

(6)

$$
\dfrac{
  \dfrac{
    \begin{array}{c} f(\mathbf{y}) \sim 0, \\ f(\mathbf{x}) \sim 0, \\ s(y) \leq \mathbf{x} \vdash \end{array}
    \qquad
    \begin{array}{c} (8) \\ \vdash s(\mathbf{y_0}) \leq max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2})) \end{array}
  }{
    \begin{array}{c} f(\mathbf{y_0}) \sim 0, \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_0}))) \sim 0 \vdash \end{array}
  }
  \qquad
  \begin{array}{c} \vdash f(\alpha) \sim 0 \\ f(\alpha) \sim 1, \\ f(\alpha) \sim 2 \end{array}
}{
  \begin{array}{c} f(\mathbf{y_0}) \sim 0 \vdash \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 1, \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 2 \\ (7) \end{array}
}
$$

$$\dfrac{\dfrac{\begin{array}{c}\vdash max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2})) \leq \\ max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))\end{array} \qquad \begin{array}{c} max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2})) \leq \\ max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2})) \vdash \\ max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})) \leq \\ max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))\end{array}}{\begin{array}{c}\vdash max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})) \leq \\ max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))\end{array}} \qquad \begin{array}{c} max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})) \leq \\ max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2})) \vdash \\ max(s(\mathbf{y_0}), s(\mathbf{y_0})) \leq \\ max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))\end{array}}{\begin{array}{c}\vdash max(s(\mathbf{y_0}), s(\mathbf{y_0})) \leq \\ max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))\end{array}}$$

$$\dfrac{\begin{array}{c}\vdash max(s(\mathbf{y_0}), s(\mathbf{y_0})) \leq \\ max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))\end{array} \qquad \begin{array}{c} max(s(\mathbf{y_0}), s(\mathbf{y_0})) \leq \\ max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2})) \vdash \\ s(\mathbf{y_0}) \leq max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))\end{array}}{\begin{array}{c}\vdash s(\mathbf{y_0}) \leq \\ max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))\end{array}}$$

(8)

$$
\dfrac{
  \dfrac{
    \begin{array}{c} f(\mathbf{y}) \sim 2, \\ f(\mathbf{x}) \sim 2, \\ s(y) \leq \mathbf{x} \vdash \end{array}
    \quad
    \begin{array}{c} (6) \\ \vdash s(\mathbf{y_1}) \leq max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2})) \end{array}
  }{
    \begin{array}{c} f(\mathbf{y_1}) \sim 1, \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 1 \vdash \end{array}
  }
  \quad
  \begin{array}{c} (7) \\ f(\mathbf{y_0}) \sim 0 \vdash \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 1, \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 2 \end{array}
}{
  \begin{array}{c} f(\mathbf{y_0}) \sim 0, f(\mathbf{y_1}) \sim 1 \vdash \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 2 \\ (9) \end{array}
}
$$

$$
\dfrac{
\dfrac{
\begin{matrix} f(\mathbf{y}) \sim 2, \\ f(\mathbf{x}) \sim 2, \\ s(y) \leq \mathbf{x} \vdash \end{matrix}
\qquad
\begin{matrix} (5) \\ \vdash s(\mathbf{y_2}) \leq max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2})) \end{matrix}
}{
\begin{matrix} f(\mathbf{y_2}) \sim 2, \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 2 \vdash \end{matrix}
}
\qquad
\begin{matrix} (7) \\ f(\mathbf{y_0}) \sim 0 \vdash \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 1, \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 2 \end{matrix}
}{
\begin{matrix} f(\mathbf{y_0}) \sim 0, f(\mathbf{y_2}) \sim 2 \vdash \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 1 \\ (10) \end{matrix}
}
$$

$$
\dfrac{
  \dfrac{
    \begin{array}{c} f(\mathbf{y}) \sim 2, \\ f(\mathbf{x}) \sim 2, \\ s(y) \leq \mathbf{x} \vdash \end{array}
    \qquad
    \begin{array}{c} (5) \\ \vdash s(\mathbf{y_2}) \leq max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2})) \end{array}
  }{
    \begin{array}{c} f(\mathbf{y_2}) \sim 2, \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 2 \vdash \end{array}
  }
  \qquad
  \begin{array}{c} (12) \\ f(\mathbf{y_1}) \sim 1 \vdash \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 1, \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 2 \end{array}
}{
  \begin{array}{c} f(\mathbf{y_1}) \sim 1, f(\mathbf{y_2}) \sim 2 \vdash \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 1 \\ (11) \end{array}
}
$$

$$
\dfrac{
  \dfrac{
    \begin{array}{c} f(\mathbf{y}) \sim 1, \\ f(\mathbf{x}) \sim 1, \\ s(y) \leq \mathbf{x} \vdash \end{array}
    \qquad
    \begin{array}{c} (6) \\ \vdash s(\mathbf{y_1}) \leq max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2})) \end{array}
  }{
    \begin{array}{c} f(\mathbf{y_1}) \sim 1, \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_0}))) \sim 1 \vdash \end{array}
  }
  \qquad
  \begin{array}{c} \vdash f(\alpha) \sim 0 \\ f(\alpha) \sim 1, \\ f(\alpha) \sim 2 \end{array}
}{
  \begin{array}{c} f(\mathbf{y_1}) \sim 1 \vdash \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 1, \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 2 \end{array}
}
$$

(12)

$$
\dfrac{
\dfrac{
\begin{array}{c} f(\mathbf{y}) \sim 2, \\ f(\mathbf{x}) \sim 2, \\ s(y) \leq \mathbf{x} \vdash \end{array}
\qquad
\begin{array}{c} (6) \\ \vdash s(\mathbf{y_2}) \leq max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2})) \end{array}
}{
\begin{array}{c} f(\mathbf{y_2}) \sim 2, \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_0}))) \sim 1 \vdash \end{array}
}
\qquad
\begin{array}{c} \vdash f(\alpha) \sim 0 \\ f(\alpha) \sim 1, \\ f(\alpha) \sim 2 \end{array}
}{
\begin{array}{c} f(\mathbf{y_2}) \sim 2 \vdash \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 1, \\ f(max(max(max(s(\mathbf{y_0}), s(\mathbf{y_0})), s(\mathbf{y_1})), s(\mathbf{y_2}))) \sim 2 \end{array}
}
$$

(13)

$$\dfrac{\overset{(1)}{f(\mathbf{y}) \sim 2 \vdash} \qquad \dfrac{\overset{(2)}{f(\mathbf{y}) \sim 1 \vdash} \qquad \dfrac{\overset{(3)}{f(\mathbf{y}) \sim 0 \vdash} \qquad \overset{(4)}{\begin{array}{c} \vdash f(\mathbf{x}) \sim 0, \\ f(\mathbf{x}) \sim 1, \\ f(\mathbf{x}) \sim 2 \end{array}}}{\begin{array}{c} \vdash f(\mathbf{x}) \sim 1, \\ f(\mathbf{x}) \sim 2 \end{array}}}{\vdash f(\mathbf{x}) \sim 2}}{\vdash}$$

(START)

APPENDIX C

Gerneralized Non-injectivity Assertion: Proofs and Refutations

C.1 LKS$_\epsilon$ Proof of NiA

The formal proof found in this section of the appendix is constructed for the schema found in Ch. 9. The end sequent of this proof is the following:

$$\forall x \bigvee_{i=0}^{n+1} f(x) \sim i \vdash \exists q \; eq(m+1,q)$$

where n and m are the free parameters and $eq(m+1,q)$ is the defined predicate symbol found in Ch. 9. Essentially stating that given a function with a range containing $n+1$ then there must be $m+1$ different values in the domain which map to the same value in the domain. The formal proof has the following structure:

$$
\begin{array}{ccc}
 & & \nabla \\
 & & \dfrac{\forall x \exists y (x \leq y \wedge f(y) \sim n+1), f(\alpha) \sim n+1 \vdash eq(m,\alpha)}{\nu(n+1,\underline{m})} \\
 & & \vdots \\
\nabla & & \nabla \\
\dfrac{\forall x \exists y (x \leq y \wedge \bigvee_{i=0}^{n} f(y) \sim i) \vdash \exists q \; eq(m+1,q)}{\varphi(\underline{n}, m+1)} & & \dfrac{\forall x \exists y (x \leq y \wedge f(y) = n+1) \vdash \exists q \; eq(m+1,q)}{\mu(n+1,\underline{m+1})}
\end{array}
$$

$$
\begin{array}{c}
\vdots \\
\nabla \\
\dfrac{\forall x \exists y (x \leq y \wedge \bigvee_{i=0}^{n+1} f(y) \sim i) \vdash \exists q \; eq(m+1,q)}{\varphi(\underline{n+1}, m+1)} \\
\vdots \\
\nabla \\
\dfrac{\forall x \bigvee_{i=0}^{n+1} f(x) \sim i \vdash \exists q \; eq(m+1,q)}{\omega(\underline{n+1}, m+1)}
\end{array}
$$

In the proof outline, as well as in the formal proof, we mark the active parameter with an underline. The idea behind having two parameters in our proof schema is that the parameters are ordered from left to right and if we manipulate a formula with a parameter lower in the order, it is then not allowed to manipulate a formula with a parameter higher in the order. The highest parameter is the left most parameter. What we mean by manipulating a formula with a parameter is applying any of the rewriting rules defined on the defined predicate and defined term symbols. Thus, it is allowed to apply standard LK rules to formulae with any parameter, but one may not apply the rewrite rules.

φ

φ **stepcase**

$$\dfrac{\dfrac{\begin{array}{c} max(\alpha,\beta) \leq \gamma \vdash \\ \alpha \leq \gamma \end{array} \qquad \begin{array}{c} \bigvee_{i=0}^{n+1} f(\gamma) \sim i, \\ f(\gamma) \not\sim n+1 \vdash \\ \bigvee_{i=0}^{n} f(\gamma) \sim i \end{array}}{\begin{array}{c} max(\alpha,\beta) \leq \gamma, \bigvee_{i=0}^{n+1} f(\gamma) \sim i, \\ f(\gamma) \not\sim n+1 \vdash \\ \alpha \leq \gamma \wedge \bigvee_{i=0}^{n} f(\gamma) \sim i \end{array}} \wedge : r \qquad \begin{array}{c} max(\alpha,\beta) \leq \gamma \vdash \\ \beta \leq \gamma \end{array}}{\begin{array}{c} max(\alpha,\beta) \leq \gamma, \\ max(\alpha,\beta) \leq \gamma, \bigvee_{i=0}^{n+1} f(\gamma) \sim i, \\ \beta \leq \gamma \rightarrow f(\gamma) \not\sim n+1 \vdash \\ \alpha \leq \gamma \wedge \bigvee_{i=0}^{n} f(\gamma) \sim i \end{array}} \rightarrow : l$$

$$\dfrac{\begin{array}{c} max(\alpha,\beta) \leq \gamma, \\ max(\alpha,\beta) \leq \gamma, \bigvee_{i=0}^{n+1} f(\gamma) \sim i, \\ \beta \leq \gamma \rightarrow f(\gamma) \not\sim n+1 \vdash \\ \alpha \leq \gamma \wedge \bigvee_{i=0}^{n} f(\gamma) \sim i \end{array}}{\begin{array}{c} max(\alpha,\beta) \leq \gamma, \bigvee_{i=0}^{n+1} f(\gamma) \sim i, \\ \beta \leq \gamma \rightarrow f(\gamma) \not\sim n+1 \vdash \\ \alpha \leq \gamma \wedge \bigvee_{i=0}^{n} f(\gamma) \sim i \end{array}} c : l$$

$$\dfrac{\cdots}{\begin{array}{c} max(\alpha,\beta) \leq \gamma \wedge \bigvee_{i=0}^{n+1} f(\gamma) \sim i, \\ \beta \leq \gamma \rightarrow f(\gamma) \not\sim n+1 \vdash \\ \alpha \leq \gamma \wedge \bigvee_{i=0}^{n} f(\gamma) \sim i \end{array}} \wedge : l$$

$$\dfrac{\cdots}{\begin{array}{c} max(\alpha,\beta) \leq \gamma \wedge \bigvee_{i=0}^{n+1} f(\gamma) \sim i, \\ \forall y(\beta \leq y \rightarrow f(y) \not\sim n+1) \vdash \\ \alpha \leq \gamma \wedge \bigvee_{i=0}^{n} f(\gamma) \sim i \end{array}} \forall : l$$

$$\dfrac{\cdots}{\begin{array}{c} max(\alpha,\beta) \leq \gamma \wedge \bigvee_{i=0}^{n+1} f(\gamma) \sim i, \\ \forall y(\beta \leq y \rightarrow f(y) \not\sim n+1) \vdash \\ \exists y(\alpha \leq y \wedge \bigvee_{i=0}^{n} f(y) \sim i) \end{array}} \exists : r$$

$$\dfrac{\cdots}{\begin{array}{c} \exists y(max(\alpha,\beta) \leq y \wedge \bigvee_{i=0}^{n+1} f(y) \sim i), \\ \forall y(\beta \leq y \rightarrow f(y) \not\sim n+1) \vdash \\ \exists y(\alpha \leq y \wedge \bigvee_{i=0}^{n} f(y) \sim i) \end{array}} \exists : l$$

$$\dfrac{\cdots}{\begin{array}{c} \forall x \exists y(x \leq y \wedge \bigvee_{i=0}^{n+1} f(y) \sim i), \\ \forall y(\beta \leq y \rightarrow f(y) \not\sim n+1) \vdash \\ \exists y(\alpha \leq y \wedge \bigvee_{i=0}^{n} f(y) \sim i) \end{array}} \forall : l$$

$$\dfrac{\cdots}{\begin{array}{c} \forall x \exists y(x \leq y \wedge \bigvee_{i=0}^{n+1} f(y) \sim i), \\ \exists x \forall y(x \leq y \rightarrow f(y) \not\sim n+1) \vdash \\ \exists y(\alpha \leq y \wedge \bigvee_{i=0}^{n} f(y) \sim i) \end{array}} \exists : l$$

$$\dfrac{\cdots}{\begin{array}{c} \forall x \exists y(x \leq y \wedge \bigvee_{i=0}^{n+1} f(y) \sim i), \\ \exists x \forall y(x \leq y \rightarrow f(y) \not\sim n+1) \vdash \\ \forall x \exists y(x \leq y \wedge \bigvee_{i=0}^{n} f(y) \sim i) \end{array}} \forall : r$$

$$\dfrac{\cdots}{\begin{array}{c} \forall x \exists y(x \leq y \wedge \bigvee_{i=0}^{n+1} f(y) \sim i), \\ \exists x \forall y(x \leq y \rightarrow f(y) \not\sim n+1) \vdash \\ \forall x \exists y(x \leq y \wedge \bigvee_{i=0}^{n} f(y) \sim i) \end{array}}$$

$$\dfrac{\begin{array}{c}\forall x\exists y(x\leq y\wedge \bigvee_{i=0}^{n+1} f(y)\sim i),\\ \exists x\forall y(x\leq y\rightarrow f(y)\not\sim n+1)\vdash\\ \forall x\exists y(x\leq y\wedge \bigvee_{i=0}^{n} f(y)\sim i)\end{array}\qquad\begin{array}{c}\exists x\forall y(x\leq y\rightarrow f(y)\neq n+1)\vdash\\ \forall x\exists y(x\leq y\wedge f(y)=n+1)\end{array}}{\begin{array}{c}\forall x\exists y(x\leq y\wedge \bigvee_{i=0}^{n+1} f(y)\sim i)\vdash\\ \forall x\exists y(x\leq y\wedge \bigvee_{i=0}^{n} f(y)\sim i),\\ \forall x\exists y(x\leq y\wedge f(y)=n+1)\\ \vdots\end{array}}\ cut$$

$$\dfrac{\begin{array}{c}\vdots\\ \forall x\exists y(x\leq y\wedge \bigvee_{i=0}^{n+1} f(y)\sim i)\vdash\\ \forall x\exists y(x\leq y\wedge \bigvee_{i=0}^{n} f(y)\sim i),\\ \forall x\exists y(x\leq y\wedge f(y)=n+1)\end{array}\qquad\begin{array}{c}\varphi\left(\{\underline{n},m+1\}\right)\\ \hdashline \forall x\exists y(x\leq y\wedge \bigvee_{i=0}^{n} f(y)\sim i)\vdash\\ \exists q\; eq(m+1,q)\end{array}}{\begin{array}{c}\forall x\exists y(x\leq y\wedge \bigvee_{i=0}^{n+1} f(y)\sim i)\vdash\\ \exists q\; eq(m+1,q),\\ \forall x\exists y(x\leq y\wedge f(y)=n+1)\\ \vdots\end{array}}\ cut$$

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \forall x\exists y(x\leq y\wedge \bigvee_{i=0}^{n+1} f(y)\sim i)\vdash\\ \exists q\; eq(m+1,q),\\ \forall x\exists y(x\leq y\wedge f(y)=n+1)\end{array}\qquad\begin{array}{c}\mu\left(\{n+1,\underline{m+1}\}\right)\\ \hdashline \forall x\exists y(x\leq y\wedge f(y)=n+1)\vdash\\ \exists q\; eq(m+1,q)\end{array}}{\begin{array}{c}\forall x\exists y(x\leq y\wedge \bigvee_{i=0}^{n+1} f(y)\sim i)\vdash\\ \exists q\; eq(m+1,q),\\ \exists q\; eq(m+1,q)\end{array}}\ cut}{\begin{array}{c}\forall x\exists y(x\leq y\wedge \bigvee_{i=0}^{n+1} f(y)\sim i)\vdash\\ \exists q\; eq(m+1,q)\end{array}}\ c:r$$

φ basecase

$$\begin{array}{c}\mu\left(\{0,\underline{m+1}\}\right)\\ \hdashline \forall x\exists y(x\leq y\wedge f(y)\sim 0)\vdash\\ \exists q\; eq(m+1,q)\end{array}$$

ω

ω stepcase

$$\dfrac{\begin{array}{c}\varphi\left(\{\underline{n+1}, m+1\}\right)\\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\ \forall x\exists y(x\leq y\wedge\bigvee_{i=0}^{n+1} f(y)\sim i)\vdash\\ \exists q\; eq(m+1,q)\end{array} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash \alpha\leq\alpha \qquad \begin{array}{c}\bigvee_{i=0}^{n+1} f(\alpha)\sim i\vdash\\ \bigvee_{i=0}^{n+1} f(\alpha)\sim i\end{array}}{\begin{array}{c}\bigvee_{i=0}^{n+1} f(\alpha)\sim i\vdash\\ \alpha\leq\alpha\wedge\bigvee_{i=0}^{n+1} f(\alpha)\sim i\end{array}}\,\wedge:r}{\begin{array}{c}\bigvee_{i=0}^{n+1} f(\alpha)\sim i\vdash\\ \exists y(\alpha\leq y\wedge\bigvee_{i=0}^{n+1} f(y)\sim i)\end{array}}\,\exists:r}{\begin{array}{c}\forall x\bigvee_{i=0}^{n+1} f(x)\sim i\vdash\\ \exists y(\alpha\leq y\wedge\bigvee_{i=0}^{n+1} f(y)\sim i)\end{array}}\,\forall:l}{\begin{array}{c}\forall x\bigvee_{i=0}^{n+1} f(x)\sim i\vdash\\ \exists y(\alpha\leq y\wedge\bigvee_{i=0}^{n+1} f(y)\sim i)\end{array}}\,\forall:r}{\begin{array}{c}\forall x\bigvee_{i=0}^{n+1} f(x)\sim i\vdash\\ \forall x\exists y(x\leq y\wedge\bigvee_{i=0}^{n+1} f(y)\sim i)\end{array}}\,\forall:r}{\begin{array}{c}\forall x\bigvee_{i=0}^{n+1} f(x)\sim i\vdash\\ \exists q\; eq(m+1,q)\end{array}}\,cut$$

ω basecase

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash \alpha\leq\alpha \qquad \begin{array}{c}f(\alpha)\sim 0\vdash\\ f(\alpha)\sim 0\end{array}}{\begin{array}{c}f(\alpha)\sim 0\vdash\\ \alpha\leq\alpha\wedge f(\alpha)\sim 0\end{array}}\,\wedge:r}{\begin{array}{c}f(\alpha)\sim 0\vdash\\ \exists y(\alpha\leq y\wedge f(y)\sim 0)\end{array}}\,\exists:r}{\begin{array}{c}\forall x f(x)\sim 0\vdash\\ \exists y(\alpha\leq y\wedge f(y)\sim 0)\end{array}}\,\forall:l}{\begin{array}{c}\forall x f(x)\sim 0\vdash\\ \forall x\exists y(x\leq y\wedge f(y)\sim 0)\end{array}}\,\forall:r \qquad \begin{array}{c}\mu\left(\{0,\underline{m}\}\right)\\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\ \forall x\exists y(x\leq y\wedge f(y)=0)\vdash\\ \exists q\; eq(m+1,q)\end{array}}{\begin{array}{c}\forall x f(x)\sim 0\vdash\\ \exists q\; eq(m,q)\end{array}}\,cut$$

μ

μ step

$$
\dfrac{
s(\beta) \leq \alpha \vdash \beta < \alpha
\qquad
\dfrac{
\dfrac{
\begin{array}{c}
\nu\left(\{n+1, \underline{m}\}, \alpha\right) \\
\cdots\cdots\cdots\cdots\cdots\cdots \\
\forall x \exists y (x \leq y \wedge f(y) \sim n+1), f(\alpha) \sim n+1 \vdash eq(m, \alpha)
\end{array}
\qquad
f(\alpha) \sim n+1, f(\beta) \sim n+1 \vdash f(\beta) \sim f(\alpha)
}{
\forall x \exists y (x \leq y \wedge f(y) \sim n+1), f(\beta) \sim n+1, f(\alpha) \sim n+1, f(\alpha) \sim n+1 \vdash eq(m, \alpha) \wedge f(\beta) \sim f(\alpha)
} \wedge : r
}{
\forall x \exists y (x \leq y \wedge f(y) \sim n+1), f(\beta) \sim n+1, f(\alpha) \sim n+1 \vdash eq(m, \alpha) \wedge f(\beta) \sim f(\alpha)
} c : l
}{
\dfrac{
\forall x \exists y (x \leq y \wedge f(y) \sim n+1), f(\beta) \sim n+1, s(\beta) \leq \alpha, f(\alpha) \sim n+1 \vdash \beta < \alpha \wedge eq(m, \alpha) \wedge f(\beta) \sim f(\alpha)
}{
\forall x \exists y (x \leq y \wedge f(y) \sim n+1), f(\beta) \sim n+1, (s(\beta) \leq \alpha \wedge f(\alpha) \sim n+1) \vdash \beta < \alpha \wedge eq(m, \alpha) \wedge f(\beta) \sim f(\alpha)
} \wedge : l
} \wedge : r
$$

$$
\begin{array}{cl}
\forall x \exists y (x \leq y \land f(y) \sim n+1) & \\
, f(\beta) \sim n+1 & \\
, (s(\beta) \leq \alpha \land f(\alpha) \sim n+1) \vdash & \\
\beta < \alpha \land eq(m, \alpha) \land f(\beta) \sim f(\alpha) & \\
\hline
\forall x \exists y (x \leq y \land f(y) \sim n+1) & \land : l \\
, (0 \leq \beta \land f(\beta) \sim n+1) & \\
, (s(\beta) \leq \alpha \land f(\alpha) \sim n+1) \vdash & \\
\beta < \alpha \land eq(m, \alpha) \land f(\beta) \sim f(\alpha) & \\
\hline
\forall x \exists y (x \leq y \land f(y) \sim n+1) & \exists : r \\
, (0 \leq \beta \land f(\beta) \sim n+1) & \\
, (s(\beta) \leq \alpha \land f(\alpha) \sim n+1) \vdash & \\
\exists p (\beta < p \land eq(m, p) \land f(\beta) \sim f(p)) & \\
\hline
\forall x \exists y (x \leq y \land f(y) \sim n+1) & \twoheadrightarrow : r \\
, (0 \leq \beta \land f(\beta) \sim n+1) & \\
, (s(\beta) \leq \alpha \land f(\alpha) \sim n+1) \vdash & \\
eq(m+1, \beta) & \\
\hline
\forall x \exists y (x \leq y \land f(y) \sim n+1) & \exists : r \\
, (0 \leq \beta \land f(\beta) \sim n+1) & \\
, (s(\beta) \leq \alpha \land f(\alpha) \sim n+1) \vdash & \\
\exists q \; eq(m+1, q) & \\
\hline
\forall x \exists y (x \leq y \land f(y) \sim n+1) & \exists : l \\
, (0 \leq \beta \land f(\beta) \sim n+1) & \\
, \exists y (\beta \leq y \land f(y) \sim n+1) \vdash & \\
\exists q \; eq(m+1, q) & \\
\hline
\forall x \exists y (x \leq y \land f(y) \sim n+1) & \forall : l \\
, (0 \leq \beta \land f(\beta) \sim n+1) & \\
, \forall x \exists y (x \leq y \land f(y) \sim n+1) \vdash & \\
\exists q \; eq(m+1, q) & \\
\hline
\forall x \exists y (x \leq y \land f(y) \sim n+1) & \exists : l \\
, \exists y (0 \leq y \land f(y) \sim n+1) & \\
, \forall x \exists y (x \leq y \land f(y) \sim n+1) \vdash & \\
\exists q \; eq(m+1, q) & \\
\hline
\forall x \exists y (x \leq y \land f(y) \sim n+1) & \forall : l \\
, \forall x \exists y (x \leq y \land f(y) \sim n+1) & \\
, \forall x \exists y (x \leq y \land f(y) \sim n+1) \vdash & \\
\exists q \; eq(m+1, q) & \\
\hline
\forall x \exists y (x \leq y \land f(y) \sim n+1) & c : l \\
, \forall x \exists y (x \leq y \land f(y) \sim n+1) \vdash & \\
\exists q \; eq(m+1, q) & \\
\hline
\forall x \exists y (x \leq y \land f(y) \sim n+1) \vdash & c : l \\
\exists q \; eq(m+1, q) &
\end{array}
$$

μ base

$$
\begin{array}{cl}
s(\beta) \leq \alpha \vdash \beta < \alpha \qquad f(\beta) \sim n+1, f(\alpha) \sim n+1 \vdash f(\beta) \sim f(\alpha) & \\
\hline
f(\beta) \sim n+1, s(\beta) \leq \alpha, f(\alpha) \sim n+1 \vdash \beta < \alpha \wedge f(\beta) \sim f(\alpha) & \wedge : r \\
\hline
f(\beta) \sim n+1, (s(\beta) \leq \alpha \wedge f(\alpha) \sim n+1) \vdash \beta < \alpha \wedge f(\beta) \sim f(\alpha) & \wedge : l \\
\hline
(0 \leq \beta \wedge f(\beta) \sim n+1), (s(\beta) \leq \alpha \wedge f(\alpha) \sim n+1) \vdash \beta < \alpha \wedge f(\beta) \sim f(\alpha) & \wedge : l \\
\hline
(0 \leq \beta \wedge f(\beta) \sim n+1), (s(\beta) \leq \alpha \wedge f(\alpha) \sim n+1) \vdash \exists p(\beta < p \wedge f(\beta) \sim f(p)) & \exists : r \\
\hline
(0 \leq \beta \wedge f(\beta) \sim n+1), (s(\beta) \leq \alpha \wedge f(\alpha) \sim n+1) \vdash eq(0, \beta) & \twoheadrightarrow : r \\
\hline
(0 \leq \beta \wedge f(\beta) \sim n+1), (s(\beta) \leq \alpha \wedge f(\alpha) \sim n+1) \vdash \exists q\, eq(0, q) & \exists : r \\
\hline
(0 \leq \beta \wedge f(\beta) \sim n+1), \exists y(\beta \leq y \wedge f(y) \sim n+1) \vdash \exists q\, eq(0, q) & \exists : l \\
\hline
(0 \leq \beta \wedge f(\beta) \sim n+1), \forall x \exists y(x \leq y \wedge f(y) \sim n+1) \vdash \exists q\, eq(0, q) & \forall : l \\
\hline
\exists y(0 \leq y \wedge f(y) \sim n+1), \forall x \exists y(x \leq y \wedge f(y) \sim n+1) \vdash \exists q\, eq(0, q) & \exists : l \\
\hline
\forall x \exists y(x \leq y \wedge f(y) \sim n+1), \forall x \exists y(x \leq y \wedge f(y) \sim n+1) \vdash \exists q\, eq(0, q) & \forall : l \\
\hline
\forall x \exists y(x \leq y \wedge f(y) \sim n+1) \vdash \exists q\, eq(0, q) & c : l
\end{array}
$$

ν

ν step

$$
\begin{prooftree}
\AxiomC{$s(\alpha) \leq \beta \vdash \alpha < \beta$}
\AxiomC{$\nu\left(\{n+1, \underline{m}\}, \beta\right)$}
\dottedLine
\UnaryInfC{$\forall x \exists y (x \leq y \land f(y) \sim n+1), f(\beta) \sim n+1 \vdash eq(m,\beta)$}
\AxiomC{$f(\beta) \sim n+1, f(\alpha) \sim n+1 \vdash f(\alpha) \sim f(\beta)$}
\RightLabel{$\land : r$}
\BinaryInfC{$\forall x \exists y (x \leq y \land f(y) \sim n+1), f(\beta) \sim n+1, f(\beta) \sim n+1, f(\alpha) \sim n+1 \vdash eq(m,\beta) \land f(\alpha) \sim f(\beta)$}
\RightLabel{$c : l$}
\UnaryInfC{$\forall x \exists y (x \leq y \land f(y) \sim n+1), f(\beta) \sim n+1, f(\alpha) \sim n+1 \vdash eq(m,\beta) \land f(\alpha) \sim f(\beta)$}
\RightLabel{$\land : r$}
\BinaryInfC{$\forall x \exists y (x \leq y \land f(y) \sim n+1), s(\alpha) \leq \beta, f(\beta) \sim n+1, f(\alpha) \sim n+1 \vdash \alpha < \beta \land eq(m,\beta) \land f(\alpha) \sim f(\beta)$}
\RightLabel{$\land : l$}
\UnaryInfC{$\forall x \exists y (x \leq y \land f(y) \sim n+1), (s(\alpha) \leq \beta \land f(\beta) \sim n+1), f(\alpha) \sim n+1 \vdash \alpha < \beta \land eq(m,\beta) \land f(\alpha) \sim f(\beta)$}
\RightLabel{$\exists : r$}
\UnaryInfC{$\forall x \exists y (x \leq y \land f(y) \sim n+1), (s(\alpha) \leq \beta \land f(\beta) \sim n+1), f(\alpha) \sim n+1 \vdash \exists p (\alpha < p \land eq(m,p) \land f(\alpha) \sim f(p))$}
\RightLabel{$\twoheadrightarrow : r$}
\UnaryInfC{$\forall x \exists y (x \leq y \land f(y) \sim n+1), (s(\alpha) \leq \beta \land f(\beta) \sim n+1), f(\alpha) \sim n+1 \vdash eq(m+1,\alpha)$}
\RightLabel{$\exists : l$}
\UnaryInfC{$\forall x \exists y (x \leq y \land f(y) \sim n+1), \exists y (s(\alpha) \leq y \land f(y) \sim n+1), f(\alpha) \sim n+1 \vdash eq(m+1,\alpha)$}
\RightLabel{$\forall : l$}
\UnaryInfC{$\forall x \exists y (x \leq y \land f(y) \sim n+1), \forall x \exists y (x \leq y \land f(y) \sim n+1), f(\alpha) \sim n+1 \vdash eq(m+1,\alpha)$}
\RightLabel{$c : l$}
\UnaryInfC{$\forall x \exists y (x \leq y \land f(y) \sim n+1), f(\alpha) \sim n+1 \vdash eq(m+1,\alpha)$}
\end{prooftree}
$$

ν base

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{s(\alpha) \leq \beta \vdash \alpha < \beta \qquad f(\beta) \sim n+1, f(\alpha) \sim n+1 \vdash f(\alpha) \sim f(\beta)}{s(\alpha) \leq \beta, f(\beta) \sim n+1, f(\alpha) \sim n+1 \vdash \alpha < \beta \land f(\alpha) \sim f(\beta)}\ \land : r}{(s(\alpha) \leq \beta \land f(\beta) \sim n+1), f(\alpha) \sim n+1 \vdash \alpha < \beta \land f(\alpha) \sim f(\beta)}\ \land : r}{(s(\alpha) \leq \beta \land f(\beta) \sim n+1), f(\alpha) \sim n+1 \vdash \exists p(\alpha < p \land f(\alpha) \sim f(p))}\ \exists : r}{(s(\alpha) \leq \beta \land f(\beta) \sim n+1), f(\alpha) \sim n+1 \vdash eq(0, \alpha)}\ \twoheadrightarrow : r}{\exists y(s(\alpha) \leq y \land f(y) \sim n+1), f(\alpha) \sim n+1 \vdash eq(0, \alpha)}\ \exists : l}{\forall x \exists y(x \leq y \land f(y) \sim n+1), f(\alpha) \sim n+1 \vdash eq(0, \alpha)}\ \forall : l$$

APPENDIX D

Empirical Results: Output and Analysis

D.1 Refutation of the Clause set extracted From the NiA-schema for Instance Three

$1[0 : Inp]$ $||||$ $\Rightarrow eq(f(U), 2)$, $eq(f(U), 1)$, $eq(f(U), 0)$

$2[0 : Inp]$ $||||$ $\Rightarrow le(U, U)$

$3[0 : Inp]$ $||||$ $le(max(U, V), W)* \Rightarrow le(U, W$

$4[0 : Inp]$ $||||$ $le(max(U, V), W)* \Rightarrow le(V, W$

$5[0 : Inp]$ $||||$ $le(s(U), V) * +$, $eqf(U), 0)*$, $eqf(V), 0)* \Rightarrow$

$6[0 : Inp]$ $||||$ $le(s(U), V) * +$, $eqf(U), 1)*$, $eqf(V), 1)* \Rightarrow$

$7[0 : Inp]$ $||||$ $le(s(U), V) * +$, $eqf(U), 2)*$, $eqf(V), 2)* \Rightarrow$

$8[0 : Res : 2.0, 4.0]$ $||||$ $\Rightarrow le(U, max(V, U))$

$11[0 : Res : 2.0, 3.0]$ $||||$ $\Rightarrow le(U, max(U, V))$

$14[0 : Res : 11.0, 3.0]$ $||||$ $\Rightarrow le(U, max(max(U, V), W))$

$15[0 : Res : 11.0, 4.0]$ $||||$ $\Rightarrow le(U, max(max(V, U), W))$

$30[0 : Res : 2.0, 7.0] \;||||\; eq(f(U), 2)\ ,\ eq(f(s(U)), 2)* \Rightarrow$

$38[0 : Res : 11.0, 7.0] \;||||\; eq(f(U), 2)\ ,\ eq(f(max(s(U), V)), 2)* \Rightarrow$

$39[0 : Res : 14.0, 7.0] \;||||\; eq(f(U), 2)\ ,\ eq(f(max(max(s(U), V), W)), 2)* \Rightarrow$

$53[0 : Res : 2.0, 6.0] \;||||\; eq(f(U), 1)\ ,\ eq(f(s(U)), 1)* \Rightarrow$

$54[0 : Res : 8.0, 6.0] \;||||\; eq(f(U), 1)\ ,\ eq(f(max(V, s(U))), 1)* \Rightarrow$

$61[0 : Res : 11.0, 6.0] \;||||\; eq(f(U), 1)\ ,\ eq(f(max(s(U), V)), 1)* \Rightarrow$

$63[0 : Res : 15.0, 6.0] \;||||\; eq(f(U), 1)\ ,\ eq(f(max(max(V, s(U)), W)), 1)* \Rightarrow$

$76[0 : Res : 2.0, 5.0] \;||||\; eq(f(U), 0)\ ,\ eq(f(s(U)), 0)* \Rightarrow$

$77[0 : Res : 8.0, 5.0] \;||||\; eq(f(U), 0)\ ,\ eq(f(max(V, s(U))), 0)* \Rightarrow$

$93[0 : Res : 1.2, 76.1] \;||||\; eq(f(U), 0) \Rightarrow eq(f(s(U)), 2)\ ,\ eq(f(s(U)), 1)$

$164[0 : Res : 1.2, 77.1] \;||||\; eq(f(U), 0) \Rightarrow eq(f(max(V, s(U))), 2)\ ,\ eq(f(max(V, s(U))), 1)$

$339[0 : Res : 164.2, 61.1] \;||||\; eq(f(U), 0)\ ,\ eq(f(V), 1) \Rightarrow eq(f(max(s(V), s(U))), 2)$

$341[0 : Res : 164.2, 63.1] \;||||\; eq(f(U), 0)\ ,\ eq(f(V), 1) \Rightarrow eq(f(max(max(W, s(V)), s(U))), 2)$

$742[0 : Res : 341.2, 39.1] \;||||\; eq(f(U), 0)*\ ,\ eqf(V), 1)*\ ,\ eqf(W), 2)* \Rightarrow$

$746[0 : MRR : 339.2, 742.2] \;||||\; eq(f(U), 0) * +\ ,\ eqf(V), 1)* \Rightarrow$

$760[0 : MRR : 93.2, 746.1] \;||||\; eq(f(U), 0) \Rightarrow eq(f(s(U)), 2)$

$761[0 : MRR : 164.2, 746.1] \;||||\; eq(f(U), 0) \Rightarrow eq(f(max(V, s(U))), 2)$

$779[0 : Res : 761.1, 38.1] \;||||\; eq(f(U), 0)*\ ,\ eqf(V), 2)* \Rightarrow$

$786[0 : MRR : 760.1, 779.1] \;||||\; eq(f(U), 0)* \Rightarrow$

$801[0 : MRR : 1.2, 786.0] \;||||\; \Rightarrow eq(f(U), 2)\ ,\ eq(f(U), 1)$

$802[0 : Res : 801.1, 53.1] \;||||\; eq(f(U), 1) \Rightarrow eq(f(s(U)), 2)$

$803[0 : Res : 801.1, 54.1] \;||||\; eq(f(U), 1) \Rightarrow eq(f(max(V, s(U))), 2)$

$820[0 : Res : 803.1, 38.1] \;||||\; eq(f(U), 1)* \;,\; eqf(V), 2)* \Rightarrow$

$827[0 : MRR : 802.1, 820.1] \;||||\; eq(f(U), 1)* \Rightarrow$

$842[0 : MRR : 801.1, 827.0] \;||||\; \Rightarrow eq(f(U), 2)$

$844[0 : MRR : 30.1, 30.0, 842.0] \;||||\; \Rightarrow$

D.2 Refutation of the Clause set extracted From the NiA-schema for Instance Four

The refutation provided in this section is almost identical to the output from SPASS except for a few minor changes to the syntax to aid reading.

$1[0 : Inp] \;||||\; \Rightarrow\; eq(f(U), 3) \;,\; eq(f(U), 2) \;,\; eq(f(U), 1) \;,\; eq(f(U), 0)*$

$2[0 : Inp] \;||||\; \Rightarrow\; le(U, U)*$

$3[0 : Inp] \;||||\; le(max(U, V), W)* \;\Rightarrow\; le(U, W)$

$4[0 : Inp] \;||||\; le(max(U, V), W)* \;\Rightarrow\; le(V, W)$

$5[0 : Inp] \;||||\; le(s(U), V) *+ \;,\; eq(f(U), 0)* \;,\; eq(f(V), 0)* \;\Rightarrow$

$6[0 : Inp] \;||||\; le(s(U), V) *+ \;,\; eq(f(U), 1)* \;,\; eq(f(V), 1)* \;\Rightarrow$

$7[0 : Inp] \;||||\; le(s(U), V) *+ \;,\; eq(f(U), 2)* \;,\; eq(f(V), 2)* \;\Rightarrow$

$8[0 : Inp] \;||||\; le(s(U), V) *+ \;,\; eq(f(U), 3)* \;,\; eq(f(V), 3)* \;\Rightarrow$

$9[0 : Res : 2.0, 4.0] \;||||\; \Rightarrow\; le(U, max(V, U))$

$10[0 : Res : 9.0, 4.0] \;||||\; \Rightarrow\; le(U, max(V, max(W, U)))$

$12[0 : Res : 2.0, 3.0] \;||||\; \Rightarrow\; le(U, max(U, V))$

$13[0 : Res : 9.0, 3.0] \;||||\; \Rightarrow\; le(U, max(V, max(U, W)))$

$15[0 : Res : 12.0, 3.0] \;||||\; \Rightarrow\; le(U, max(max(U, V), W))$

$16[0:Res:12.0,4.0] \,||||\, \Rightarrow\; le(U,max(max(V,U),W))$

$19[0:Res:15.0,3.0] \,||||\, \Rightarrow\; le(U,max(max(max(U,V),W),X))$

$20[0:Res:15.0,4.0] \,||||\, \Rightarrow\; le(U,max(max(max(V,U),W),X))$

$23[0:Res:2.0,8.0] \,||||\, eq(f(U),3)\;,\; eq(f(s(U)),3)* \;\Rightarrow$

$25[0:Res:10.0,8.0] \,||||\, eq(f(U),3)\;,\; eq(f(max(V,max(W,s(U)))),3)* \;\Rightarrow$

$27[0:Res:12.0,8.0] \,||||\, eq(f(U),3)\;,\; eq(f(max(s(U),V)),3)* \;\Rightarrow$

$28[0:Res:15.0,8.0] \,||||\, eq(f(U),3)\;,\; eq(f(max(max(s(U),V),W)),3)* \;\Rightarrow$

$42[0:Res:2.0,7.0] \,||||\, eq(f(U),2)\;,\; eq(f(s(U)),2)* \;\Rightarrow$

$43[0:Res:9.0,7.0] \,||||\, eq(f(U),2)\;,\; eq(f(max(V,s(U))),2)* \;\Rightarrow$

$44[0:Res:10.0,7.0] \,||||\, eq(f(U),2)\;,\; eq(f(max(V,max(W,s(U)))),2)* \;\Rightarrow$

$50[0:Res:12.0,7.0] \,||||\, eq(f(U),2)\;,\; eq(f(max(s(U),V)),2)* \;\Rightarrow$

$52[0:Res:16.0,7.0] \,||||\, eq(f(U),2)\;,\; eq(f(max(max(V,s(U)),W)),2)* \;\Rightarrow$

$54[0:Res:19.0,4.0] \,||||\, \Rightarrow\; le(U,max(max(max(max(V,U),W),X),Y))$

$59[0:Res:20.0,7.0] \,||||\, eq(f(U),2)\;,\; eq(f(max(max(max(V,s(U)),W),X)),2)* \;\Rightarrow$

$69[0:Res:2.0,6.0] \,||||\, eq(f(U),1)\;,\; eq(f(s(U)),1)* \;\Rightarrow$

$70[0:Res:9.0,6.0] \,||||\, eq(f(U),1)\;,\; eq(f(max(V,s(U))),1)* \;\Rightarrow$

$74[0:Res:13.0,6.0] \,||||\, eq(f(U),1)\;,\; eq(f(max(V,max(s(U),W))),1)* \;\Rightarrow$

$79[0:Res:16.0,6.0] \,||||\, eq(f(U),1)\;,\; eq(f(max(max(V,s(U)),W)),1)* \;\Rightarrow$

$89[0:Res:2.0,5.0] \,||||\, eq(f(U),0)\;,\; eq(f(s(U)),0)* \;\Rightarrow$

$90[0:Res:9.0,5.0] \,||||\, eq(f(U),0)\;,\; eq(f(max(V,s(U))),0)* \;\Rightarrow$

$98[0:Res:12.0,5.0] \,||||\, eq(f(U),0)\;,\; eq(f(max(s(U),V)),0)* \;\Rightarrow$

$123[0:Res:1.3,89.1] \,||||\, eq(f(U),0) \;\Rightarrow\; eq(f(s(U)),3)\;,\; eq(f(s(U)),2)\;,\; eq(f(s(U)),1)$

$159[0:Res:54.0,8.0]\ \|\|\|\ eq(f(U),3)\ ,\ eq(f(max(max(max(max(V,s(U)),W),X),Y)),3)*\ \Rightarrow$

$196[0:Res:1.3,90.1]\ \|\|\|\ eq(f(U),0)\ \Rightarrow\ eq(f(max(V,s(U))),3)$

$eq(f(max(V,s(U))),2)\ ,\ eq(f(max(V,s(U))),1)$

$197[0:Res:1.3,98.1]\ \|\|\|\ eq(f(U),0)\ \Rightarrow\ eq(f(max(s(U),V)),3)$

$eq(f(max(s(U),V)),2)\ ,\ eq(f(max(s(U),V)),1)$

$423[0:Res:196.3,79.1]\ \|\|\|\ eq(f(U),0)\ ,\ eq(f(V),1)\ \Rightarrow$

$eq(f(max(max(W,s(V)),s(U))),3)\ ,\ eq(f(max(max(W,s(V)),s(U))),2)$

$450[0:Res:197.3,74.1]\ \|\|\|\ eq(f(U),0)\ ,\ eq(f(V),1)\ \Rightarrow$

$eq(f(max(s(U),max(s(V),W))),3)\ ,\ eq(f(max(s(U),max(s(V),W))),2)$

$955[0:Res:423.3,59.1]\ \|\|\|\ eq(f(U),0)\ ,\ eq(f(V),1)\ ,\ eq(f(W),2)$

$\Rightarrow\ eq(f(max(max(max(X,s(W)),s(V)),s(U))),3)$

$1009[0:Res:450.3,44.1]\ \|\|\|\ eq(f(U),0)\ ,\ eq(f(V),1)\ ,\ eq(f(W),2)$

$\Rightarrow\ eq(f(max(s(U),max(s(V),s(W)))),3)$

$2272[0:Res:955.3,159.1]\ \|\|\|\ eq(f(U),0)*\ ,\ eq(f(V),1)*\ ,\ eq(f(W),2)*$

$eq(f(X),3)*\ \Rightarrow$

$2273[0:MRR:1009.3,2272.3]\ \|\|\|\ eq(f(U),0)*+\ ,\ eq(f(V),1)*$

$eq(f(W),2)*\ \Rightarrow$

$2301[0:MRR:450.3,2273.2]\ \|\|\|\ eq(f(U),0)\ ,\ eq(f(V),1)$

$\Rightarrow\ eq(f(max(s(U),max(s(V),W))),3)$

$2450[0:Res:2301.2,25.1]\ \|\|\|\ eq(f(U),0)*\ ,\ eq(f(V),1)*$

$eq(f(W),3)*\ \Rightarrow$

$2459[0:MRR:2301.2,2450.2]$ |||| $eq(f(U),0)*+ ,\ eq(f(V),1)* \Rightarrow$

$2577[0:MRR:123.3,2459.1]$ |||| $eq(f(U),0) \Rightarrow eq(f(s(U)),3)$

$eq(f(s(U)),2)$

$2578[0:MRR:196.3,2459.1]$ |||| $eq(f(U),0) \Rightarrow$
$eq(f(max(V,s(U))),3) ,\ eq(f(max(V,s(U))),2)$

$2613[0:Res:2578.2,50.1]$ |||| $eq(f(U),0) ,\ eq(f(V),2) \Rightarrow eq(f(max(s(V),s(U))),3)$

$2615[0:Res:2578.2,52.1]$ |||| $eq(f(U),0) ,\ eq(f(V),2)$
$\Rightarrow eq(f(max(max(W,s(V)),s(U))),3)$

$2676[0:Res:2615.2,28.1]$ |||| $eq(f(U),0)* ,\ eq(f(V),2)* ,\ eq(f(W),3)* \Rightarrow$

$2684[0:MRR:2613.2,2676.2]$ |||| $eq(f(U),0)*+ ,\ eq(f(V),2)* \Rightarrow$

$2714[0:MRR:2577.2,2684.1]$ |||| $eq(f(U),0) \Rightarrow eq(f(s(U)),3)$

$2715[0:MRR:2578.2,2684.1]$ |||| $eq(f(U),0) \Rightarrow eq(f(max(V,s(U))),3)$

$2749[0:Res:2715.1,27.1]$ |||| $eq(f(U),0)* ,\ eq(f(V),3)* \Rightarrow$

$2764[0:MRR:2714.1,2749.1]$ |||| $eq(f(U),0)* \Rightarrow$

$2795[0:MRR:1.3,2764.0]$ |||| $\Rightarrow eq(f(U),3) ,\ eq(f(U),2) ,\ eq(f(U),1)$

$2796[0:Res:2795.2,69.1]$ |||| $eq(f(U),1) \Rightarrow eq(f(s(U)),3) ,\ eq(f(s(U)),2)$

$2797[0:Res:2795.2,70.1]$ |||| $eq(f(U),1) \Rightarrow eq(f(max(V,s(U))),3)$
$eq(f(max(V,s(U))),2)$

$2831[0:Res:2797.2,50.1]$ |||| $eq(f(U),1) ,\ eq(f(V),2)$
$\Rightarrow eq(f(max(s(V),s(U))),3)$

$2833[0:Res:2797.2,52.1]$ |||| $eq(f(U),1) ,\ eq(f(V),2)$
$\Rightarrow eq(f(max(max(W,s(V)),s(U))),3)$

$2896[0:Res:2833.2,28.1]$ |||| $eq(f(U),1)* ,\ eq(f(V),2)* ,\ eq(f(W),3)* \Rightarrow$

$2904[0:MRR:2831.2,2896.2]$ |||| $eq(f(U),1)*+ ,\ eq(f(V),2)* \Rightarrow$

$2934[0 : MRR : 2796.2, 2904.1] \|\|\| eq(f(U), 1) \Rightarrow eq(f(s(U)), 3)$

$2935[0 : MRR : 2797.2, 2904.1] \|\|\| eq(f(U), 1) \Rightarrow eq(f(max(V, s(U))), 3)$

$2969[0 : Res : 2935.1, 27.1] \|\|\| eq(f(U), 1)* , eq(f(V), 3)* \Rightarrow$

$2984[0 : MRR : 2934.1, 2969.1] \|\|\| eq(f(U), 1)* \Rightarrow$

$3015[0 : MRR : 2795.2, 2984.0] \|\|\| \Rightarrow eq(f(U), 3) , eq(f(U), 2)$

$3016[0 : Res : 3015.1, 42.1] \|\|\| eq(f(U), 2) \Rightarrow eq(f(s(U)), 3)$

$3017[0 : Res : 3015.1, 43.1] \|\|\| eq(f(U), 2) \Rightarrow eq(f(max(V, s(U))), 3)$

$3050[0 : Res : 3017.1, 27.1] \|\|\| eq(f(U), 2)* , eq(f(V), 3)* \Rightarrow$

$3065[0 : MRR : 3016.1, 3050.1] \|\|\| eq(f(U), 2)* \Rightarrow$

$3096[0 : MRR : 3015.1, 3065.0] \|\|\| \Rightarrow eq(f(U), 3)$

$3098[0 : MRR : 23.1, 23.0, 3096.0] \|\|\| \Rightarrow$

D.3 Derived Clause and Spass Refutation for Instance Four of the NiA-schema

Clause Set

$eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 3)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 2)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 1)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 0)$

$\neg eq(f(A), 3) \vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 2)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 1)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 0)$

$\neg eq(f(A), 2) \vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 3)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 1)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 0)$

$\neg eq(f(A), 1) \vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 3)$

$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 2)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 0)$

$\neg eq(f(A), 0) \vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 3)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 2)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 1)$

$\neg eq(f(A), 3) \vee \neg eq(f(B), 2)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 1)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 0)$

$\neg eq(f(A), 3) \vee \neg eq(f(B), 1)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 2)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 0)$

$\neg eq(f(A), 3) \vee \neg eq(f(B), 0)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 2)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 1)$

$\neg eq(f(A), 2) \vee \neg eq(f(B), 1)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 3)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 0)$

$\neg eq(f(A), 2) \vee \neg eq(f(B), 0)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 3)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 1)$

$\neg eq(f(A), 1) \vee \neg eq(f(B), 0)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 3)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 2)$

$\neg eq(f(A), 3) \vee \neg eq(f(B), 2) \vee \neg eq(f(C), 1)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 0)$

$\neg eq(f(A), 3) \vee \neg eq(f(B), 2) \vee \neg eq(f(C), 0)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 1)$

$\neg eq(f(A), 3) \vee \neg eq(f(B), 1) \vee \neg eq(f(C), 0)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 2)$

$\neg eq(f(A), 2) \vee \neg eq(f(B), 1) \vee \neg eq(f(C), 0)$
$\vee eq(f(max(max(max(s(D), s(C)), s(B)), s(A))), 3)$

$\neg eq(f(A), 3) \vee \neg eq(f(B), 1) \vee \neg eq(f(C), 0) \vee \neg eq(f(D), 2)$

Refutation

$1[0:Inp] \;||||\; \Rightarrow eq(f(max(max(max(s(U), s(V)), s(W)), s(X))), 3) *$
$eq(f(max(max(max(s(U), s(V)), s(W)), s(X))), 2)$
$eq(f(max(max(max(s(U), s(V)), s(W)), s(X))), 1)$
$eq(f(max(max(max(s(U), s(V)), s(W)), s(X))), 0$

$2[0:Inp] \;||||\; eq(f(U), 3) \Rightarrow eq(f(max(max(max(s(V), s(W)), s(X)), s(U))), 2) *$
$eq(f(max(max(max(s(V), s(W)), s(X)), s(U))), 1)$
$eq(f(max(max(max(s(V), s(W)), s(X)), s(U))), 0$

$3[0:Inp] \;||||\; eq(f(U), 2) \Rightarrow eq(f(max(max(max(s(V), s(W)), s(X)), s(U))), 3) *$
$eq(f(max(max(max(s(V), s(W)), s(X)), s(U))), 1)$
$eq(f(max(max(max(s(V), s(W)), s(X)), s(U))), 0$

$4[0:Inp] \;||||\; eq(f(U), 1) \Rightarrow eq(f(max(max(max(s(V), s(W)), s(X)), s(U))), 3) *$
$eq(f(max(max(max(s(V), s(W)), s(X)), s(U))), 2)$
$eq(f(max(max(max(s(V), s(W)), s(X)), s(U))), 0$

$5[0:Inp] \;||||\; eq(f(U), 0) \Rightarrow eq(f(max(max(max(s(V), s(W)), s(X)), s(U))), 3) *$
$eq(f(max(max(max(s(V), s(W)), s(X)), s(U))), 2)$
$eq(f(max(max(max(s(V), s(W)), s(X)), s(U))), 1$

$6[0:Inp] \;||||\; eq(f(U), 3) eq(f(V), 2) \Rightarrow eq(f(max(max(max(s(W), s(X)), s(V)), s(U))), 1) *$
$eq(f(max(max(max(s(W), s(X)), s(V)), s(U))), 0$

$7[0:Inp] \;||||\; eq(f(U), 3) eq(f(V), 1) \Rightarrow eq(f(max(max(max(s(W), s(X)), s(V)), s(U))), 2) *$
$eq(f(max(max(max(s(W), s(X)), s(V)), s(U))), 0$

$8[0:Inp] \;||||\; eq(f(U), 3) eq(f(V), 0) \Rightarrow eq(f(max(max(max(s(W), s(X)), s(V)), s(U))), 2) *$
$eq(f(max(max(max(s(W), s(X)), s(V)), s(U))), 1$

$9[0:Inp] \;||||\; eq(f(U), 2) eq(f(V), 1) \Rightarrow eq(f(max(max(max(s(W), s(X)), s(V)), s(U))), 3) *$
$eq(f(max(max(max(s(W), s(X)), s(V)), s(U))), 0$

$10[0:Inp] \;||||\; eq(f(U), 2) eq(f(V), 0) \Rightarrow eq(f(max(max(max(s(W), s(X)), s(V)), s(U))), 3) *$
$eq(f(max(max(max(s(W), s(X)), s(V)), s(U))), 1$

$11[0:Inp] \;||||\; eq(f(U), 1) eq(f(V), 0) \Rightarrow eq(f(max(max(max(s(W), s(X)), s(V)), s(U))), 3) *$
$eq(f(max(max(max(s(W), s(X)), s(V)), s(U))), 2$

$12[0:Inp] \;||||\; eq(f(U), 3) eq(f(V), 2) eq(f(W), 1) \Rightarrow$

$eq(f(max(max(max(s(X), s(W)), s(V)), s(U))), 0)$

$13[0 : Inp]$ |||| $eq(f(U), 3)eq(f(V), 2)eq(f(W), 0) \Rightarrow$
$eq(f(max(max(max(s(X), s(W)), s(V)), s(U))), 1)$

$14[0 : Inp]$ |||| $eq(f(U), 3)eq(f(V), 1)eq(f(W), 0) \Rightarrow$
$eq(f(max(max(max(s(X), s(W)), s(V)), s(U))), 2)$

$15[0 : Inp]$ |||| $eq(f(U), 2)eq(f(V), 1)eq(f(W), 0) \Rightarrow$
$eq(f(max(max(max(s(X), s(W)), s(V)), s(U))), 3)$

$16[0 : Inp]$ |||| $eq(f(U), 3) * eq(f(V), 1) * eq(f(W), 0) * eq(f(X), 2)* \Rightarrow$

$17[0 : MRR : 12.3, 16.2]$ |||| $eq(f(U), 3) * eq(f(V), 2) * eq(f(W), 1)* \Rightarrow$

$18[0 : MRR : 13.3, 17.2]$ |||| $eq(f(U), 3) * eq(f(V), 2) * eq(f(W), 0)* \Rightarrow$

$19[0 : MRR : 14.3, 18.1]$ |||| $eq(f(U), 3) * eq(f(V), 1) * eq(f(W), 0)* \Rightarrow$

$20[0 : MRR : 15.3, 19.0]$ |||| $eq(f(U), 2) * eq(f(V), 1) * eq(f(W), 0)* \Rightarrow$

$21[0 : MRR : 6.2, 6.3, 17.2, 18.2]$ |||| $eq(f(U), 3) * eq(f(V), 2)* \Rightarrow$

$22[0 : MRR : 7.2, 7.3, 21.1, 19.2]$ |||| $eq(f(U), 3) * eq(f(V), 1)* \Rightarrow$

$23[0 : MRR : 8.2, 8.3, 21.1, 22.1]$ |||| $eq(f(U), 3) * eq(f(V), 0)* \Rightarrow$

$24[0 : MRR : 9.2, 9.3, 22.0, 20.2]$ |||| $eq(f(U), 2) * eq(f(V), 1)* \Rightarrow$

$25[0 : MRR : 10.2, 10.3, 23.0, 24.1]$ |||| $eq(f(U), 2) * eq(f(V), 0)* \Rightarrow$

$26[0 : MRR : 11.2, 11.3, 23.0, 25.0]$ |||| $eq(f(U), 1) * eq(f(V), 0)* \Rightarrow$

$27[0 : MRR : 2.1, 2.2, 2.3, 21.1, 22.1, 23.1]$ |||| $eq(f(U), 3)* \Rightarrow$

$28[0 : MRR : 3.1, 3.2, 3.3, 27.0, 24.1, 25.1]$ |||| $eq(f(U), 2)* \Rightarrow$

$29[0 : MRR : 4.1, 4.2, 4.3, 27.0, 28.0, 26.1]$ |||| $eq(f(U), 1)* \Rightarrow$

$30[0 : MRR : 5.1, 5.2, 5.3, 27.0, 28.0, 29.0]$ |||| $eq(f(U), 0)* \Rightarrow$

$31[0 : MRR : 1.0, 1.1, 1.2, 1.3, 27.0, 28.0, 29.0, 30.0]$ |||| $\Rightarrow$

D.4 Refutation Found by VAMPIR for the *g*NiA-schema With the Free Parameters Instantiated to n=2 and m=3

$fof(f1, axiom, ((![X0] : (eq(f(X0), 1)|eq(f(X0), 0)))),$
$file('/tmp/SystemOnTPTPFormReply32094/SOT_jn8KUb', sos)).$

$fof(f2, axiom, ((![X0] : (le(X0, X0)))),$
$file('/tmp/SystemOnTPTPFormReply32094/SOT_jn8KUb', sos_001)).$

$fof(f3, axiom, ((![X2, X0, X1] : (le(max(X0, X1), X2)|le(X0, X2)))),$
$file('/tmp/SystemOnTPTPFormReply32094/SOT_jn8KUb', sos_002)).$

$fof(f4, axiom, ((![X2, X0, X1] : (le(max(X0, X1), X2)|le(X1, X2)))),$
$file('/tmp/SystemOnTPTPFormReply32094/SOT_jn8KUb', sos_003)).$

$fof(f5, axiom, ((![X2, X0, X1] : (le(s(X2), X1)|\ eq(f(X1), 0)|$
$eq(f(X2), 0)|\ le(s(X1), X0)|\ eq(f(X0), 0)))),$
$file('/tmp/SystemOnTPTPFormReply32094/SOT_jn8KUb', sos_004)).$

$fof(f6, axiom, ((![X2, X0, X1] : (le(s(X2), X1)|\ eq(f(X1), 1)|$
$eq(f(X2), 1)|\ le(s(X1), X0)|\ eq(f(X0), 1)))),$
$file('/tmp/SystemOnTPTPFormReply32094/SOT_jn8KUb', sos_005)).$

$fof(f7, plain, ((![X0, X1] : (le(X0, max(X0, X1))))),$
$inference(resolution, [], [f3, f2])).$

$fof(f10, plain, ((![X0, X1] : (le(X0, max(X1, X0))))),$
$inference(resolution, [], [f4, f2])).$

$fof(f11, plain, ((![X4, X2, X3] : (le(X2, max(max(X3, X2), X4))))),$
$inference(resolution, [], [f4, f7])).$

$fof(f14, plain, ((![X4, X5, X3] : (le(X3, max(X4, max(X3, X5)))))),$
$inference(resolution, [], [f10, f3])).$

$fof(f15, plain, ((![X2, X0, X3, X1] : (le(X0, max(max(X1, max(X2, X0)), X3))))),$
$inference(resolution, [], [f11, f4])).$

$fof(f34, plain, ((![X4, X2, X3] : (le(s(max(s(X2), X3)), X4)|$
$eq(f(X2), 0)|\ eq(f(max(s(X2), X3)), 0)|\ eq(f(X4), 0)))),$
$inference(resolution, [], [f5, f7])).$

$fof(f41, plain, ((![X35, X33, X34] : (le(s(max(X33, s(X34))), X35)|$
$eq(f(X34), 0)| eq(f(max(X33, s(X34))), 0)| eq(f(X35), 0)))),$
$inference(resolution, [], [f5, f10])).$

$fof(f45, plain, ((![X52, X50, X53, X51] :$
$(le(s(max(X50, max(s(X51), X52))), X53)| eq(f(X51), 0)|$
$eq(f(max(X50, max(s(X51), X52))), 0)| eq(f(X53), 0)))),$
$inference(resolution, [], [f5, f14])).$

$fof(f88, plain, ((![X4, X2, X3] : (le(s(max(s(X2), X3)), X4)|$
$eq(f(X2), 1)| eq(f(max(s(X2), X3)), 1)| eq(f(X4), 1)))),$
$inference(resolution, [], [f6, f7])).$

$fof(f799, plain, ((![X449, X451, X450] :$
$(eq(f(max(X451, s(max(s(X449), X450)))), 0)|$
$eq(f(max(s(X449), X450)), 0)| eq(f(X449), 0)))),$
$inference(resolution, [], [f34, f10])).$

$fof(f1012, plain, ((![X494, X496, X495] :$
$(eq(f(max(X496, s(max(X495, s(X494))))), 0)|$
$eq(f(max(X495, s(X494))), 0)| eq(f(X494), 0)))),$
$inference(resolution, [], [f41, f10])).$

$fof(f1302, plain, ((![X848, X847, X845, X846] :$
$(eq(f(max(X847, max(s(max(s(X845), X846)), X848))), 1)|$
$eq(f(max(s(X845), X846)), 1)| eq(f(X845), 1)))),$
$inference(resolution, [], [f88, f14])).$

$fof(f1188, plain, ((![X4, X2, X3] : (eq(f(max(s(max(s(X2), X3)), X4)), 1)|$
$eq(f(max(s(X2), X3)), 1)| eq(f(X2), 1)))),$
$inference(resolution, [], [f88, f7])).$

$fof(f1193, plain, ((![X26, X24, X23, X27, X25] :$
$(eq(f(max(max(X25, max(X26, s(max(s(X23), X24)))), X27)), 1)|$
$eq(f(max(s(X23), X24)), 1)| eq(f(X23), 1)))),$
$inference(resolution, [], [f88, f15])).$

$fof(f3142, plain, ((![X2, X0, X1] : (eq(f(max(s(max(s(X0), X1)), X2)), 0)|$
$eq(f(X0), 1)| eq(f(max(s(X0), X1)), 1)))),$
$inference(resolution, [], [f1188, f1])).$

$fof(f5241, plain, ((![X2, X0, X3, X1] : (eq(f(max(X2, max(s(max(s(X0), X1)), X3))), 0)$

$| \ eq(f(X0), 1)| \ eq(f(max(s(X0), X1)), 1)))),$
$inference(resolution, [], [f1302, f1])).$

$fof(f5289, plain, ((![X10, X8, X7, X9] : (\ eq(f(max(s(X9), X10)), 0)|$
$eq(f(max(s(X7), X8)), 1)| \ eq(f(X7), 1)| \ eq(f(X9), 0)))),$
$inference(resolution, [], [f3142, f799])).$

$fof(f7263, plain, ((![X2, X0, X1] : (\ eq(f(s(max(X1, max(s(X0), X2)))), 0)|$
$eq(f(max(X1, max(s(X0), X2))), 0)| \ eq(f(X0), 0)))),$
$inference(resolution, [], [f45, f2])).$

$fof(f18474, plain, ((![X30, X28, X31, X29, X32] : (\ eq(f(max(s(X28), X29)), 1)|$
$eq(f(X28), 1)| \ eq(f(X30), 0)| \ eq(f(X31), 1)| \ eq(f(max(s(X31), X32)), 1)))),$
$inference(resolution, [], [f5289, f5241])).$

$fof(f18476, plain, ((![X2, X0, X1] : (\ eq(f(max(s(X0), X1)), 1)| \ eq(f(X0), 1)|$
$eq(f(X2), 0)| \ eq(f(X0), 1)))),$
$inference(condensation, [], [f18474])).$

$fof(f18477, plain, ((![X2, X0, X1] : (\ eq(f(max(s(X0), X1)), 1)| \ eq(f(X0), 1)| \ eq(f(X2), 0)))),$
$inference(duplicate_literal_removal, [], [f18476])).$

$fof(f18533, plain, ((![X4, X2, X0, X3, X1] : (\ eq(f(max(s(X0), X1)), 1)| \ eq(f(X0), 1)$
$|eq(f(max(max(X2, max(X3, s(max(s(X0), X1)))), X4)), 0)))),$
$inference(resolution, [], [f1193, f1])).$

$fof(f18534, plain, ((![X0, X1] : (\ eq(f(max(s(X0), X1)), 1)| \ eq(f(X0), 1)))),$
$inference(subsumption_resolution, [], [f18533, f18477])).$

$fof(f18535, plain, ((![X0, X1] : (eq(f(max(s(X0), X1)), 0)| \ eq(f(X0), 1)))),$
$inference(resolution, [], [f18534, f1])).$

$fof(f18541, plain, ((![X12, X10, X11] : (\ eq(f(max(X11, s(X12))), 0)|$
$eq(f(X10), 1)| \ eq(f(X12), 0)))),$
$inference(resolution, [], [f18535, f1012])).$

$fof(f18571, plain, ((![X12, X13, X11] : (\ eq(f(X11), 1)| \ eq(f(X12), 0)| \ eq(f(X13), 1)))),$
$inference(resolution, [], [f18541, f18535])).$

$fof(f18572, plain, ((![X0, X1] : (\ eq(f(X1), 0)| \ eq(f(X0), 1)))),$
$inference(condensation, [], [f18571])).$

$fof(f18582, plain, ((![X17, X18] : (\ eq(f(X17), 1)| \ eq(f(X18), 1)))),$

$inference(resolution, [], [f18572, f18535]))$.

$fof(f18583, plain, ((![X0] : (eq(f(X0), 1)))),$
$inference(condensation, [], [f18582]))$.

$fof(f18590, plain, ((![X0] : (eq(f(X0), 0)))),$
$inference(subsumption_resolution, [], [f1, f18583]))$.

$fof(f18648, plain, ((![X2, X0, X1] : (eq(f(s(max(X1, max(s(X0), X2)))), 0)|$
$eq(f(max(X1, max(s(X0), X2))), 0)))),$
$inference(subsumption_resolution, [], [f7263, f18590]))$.

$fof(f18649, plain, ((![X2, X0, X1] : (eq(f(max(X1, max(s(X0), X2))), 0)))),$
$inference(subsumption_resolution, [], [f18648, f18590]))$.

$fof(f18650, plain, (false),$
$inference(subsumption_resolution, [], [f18649, f18590]))$.

D.5 *g*NiA-schema Clause set for two Symbols and 3 Occurrences

Clause set Without Term Schema (TPTP Syntax)

$1 : \ eq(f(A), 0) \mid eq(f(A), 1)$
$2 : \ le(A, A)$
$3 : \sim le(max(A, B), C) \mid le(A, C)$
$4 : \sim le(max(A, B), C) \mid le(B, C)$
$5 : \sim eq(f(A), 0) \mid \ \sim eq(f(B), 0) \mid \ \sim eq(f(C), 0) \mid$
$\sim le(s(B), A) \mid \ \sim le(s(C), B)$
$6 : \sim eq(f(A), 1) \mid \ \sim eq(f(B), 1) \mid \ \sim eq(f(C), 1) \mid$
$\sim le(s(B), A) \mid \ \sim le(s(C), B)$

Clause set with Term Schema (TPTP Syntax)

Note, this clause set does not require the definition of the max function because all the configuration of the max function required to refute the clause set are already constructed. Also, Axy where $x, y \in [0, 1]$ are the variables representing the 4 points required to construct a face of a cube. This problem is a simplified version of the problem found in the next section of this appaendix.

$1 : \ eq(f(A), 0) \mid eq(f(A), 1)$
$2 : \sim eq(f(A00), 0) \mid \ \sim eq(f(max(s(A00), s(A10))), 0) \mid$
$\sim eq(f(max(s(max(s(A00), s(A10))), s(max(s(A01), s(A11))))), 0)$
$3 : \sim eq(f(A01), 0) \mid \ \sim eq(f(max(s(A01), s(A11))), 0) \mid$
$\sim eq(f(max(s(max(s(A00), s(A10))), s(max(s(A01), s(A11))))), 0)$
$4 : \sim eq(f(A10), 1) \mid \ \sim eq(f(max(s(A00), s(A10))), 1) \mid$

$\sim eq(f(max(s(max(s(A00), s(A10))), s(max(s(A01), s(A11))))), 1)$
$5 : \sim eq(f(A11), 1) \mid \sim eq(f(max(s(A01), s(A11))), 1) \mid$
$\sim eq(f(max(s(max(s(A00), s(A10))), s(max(s(A01), s(A11))))), 1)$

SPASS Refutation of Clause set with Term Schema (TPTP Syntax)

$1[0 : Inp] \|\|\| \Rightarrow eq(f(U), 1) \ , \ eq(f(U), 0)*$

$2[0 : Inp] \|\|\| eq(f(U), 0) \ , \ eq(f(max(s(U), s(V))), 0) \ ,$
$eq(f(max(s(max(s(U), s(V))), s(max(s(W), s(X))))), 0)* \Rightarrow$

$3[0 : Inp] \|\|\| eq(f(U), 0) \ , \ eq(f(max(s(U), s(V))), 0) \ ,$
$eq(f(max(s(max(s(W), s(X))), s(max(s(U), s(V))))), 0)* \Rightarrow$

$4[0 : Inp] \|\|\| eq(f(U), 1) \ , \ eq(f(max(s(V), s(U))), 1) \ ,$
$eq(f(max(s(max(s(V), s(U))), s(max(s(W), s(X))))), 1)* \Rightarrow$

$5[0 : Inp] \|\|\| eq(f(U), 1) \ , \ eq(f(max(s(V), s(U))), 1) \ ,$
$eq(f(max(s(max(s(W), s(X))), s(max(s(V), s(U))))), 1)* \Rightarrow$

$6[0 : Res : 1.1, 2.2] \|\|\| eq(f(U), 0) \ , \ eq(f(max(s(U), s(V))), 0) \Rightarrow$
$eq(f(max(s(max(s(U), s(V))), s(max(s(W), s(X))))), 1)$

$7[0 : Res : 1.1, 3.2] \|\|\| eq(f(U), 0) \ , \ eq(f(max(s(U), s(V))), 0) \Rightarrow$
$eq(f(max(s(max(s(W), s(X))), s(max(s(U), s(V))))), 1)$

$10[0 : Res : 7.2, 4.2] \|\|\| eq(f(U), 0) \ , \ eq(f(max(s(U), s(V))), 0) *+ \ ,$
$eq(f(W), 1) \ , \ eq(f(max(s(X), s(W))), 1)* \Rightarrow$

$11[0 : Res : 1.1, 10.1] \|\|\| eq(f(U), 0) \ , \ eq(f(V), 1) \ , \ eq(f(max(s(W), s(V))), 1) *+ \Rightarrow$
$eq(f(max(s(U), s(X))), 1)$

$12[0 : Res : 6.2, 11.2] \|\|\| eq(f(U), 0) \ , \ eq(f(max(s(U), s(V))), 0) *+ \ ,$
$eq(f(W), 0) \ , \ eq(f(max(s(X), s(Y))), 1)* \Rightarrow eq(f(max(s(W), s(Z))), 1)$

$16[0 : Res : 1.1, 12.1] \|\|\| eq(f(U), 0) \ , \ eq(f(V), 0) \ , \ eq(f(max(s(W), s(X))), 1)* \Rightarrow$
$eq(f(max(s(U), s(Y))), 1)* \ , \ eq(f(max(s(V), s(Z))), 1)$

$17[0 : Con : 16.1] \|\|\| eq(f(U), 0) \ , \ eq(f(max(s(V), s(W))), 1)*+ \Rightarrow eq(f(max(s(U), s(X))), 1)$

$19[0 : Res : 7.2, 17.1] \|\|\| eq(f(U), 0) \ , \ eq(f(max(s(U), s(V))), 0) *+ \ , \ eq(f(W), 0) \Rightarrow$
$eq(f(max(s(W), s(X))), 1)$

$20[0 : Res : 1.1, 19.1] \;\|\|\|\; eq(f(U), 0)\ ,\ eq(f(V), 0) \Rightarrow$
$eq(f(max(s(U), s(W))), 1)*\ ,\ eq(f(max(s(V), s(X))), 1)$

$21[0 : Con : 20.1] \;\|\|\|\; eq(f(U), 0) \Rightarrow eq(f(max(s(U), s(V))), 1)$

$23[0 : Res : 21.1, 5.2] \;\|\|\|\; eq(f(max(s(U), s(V))), 0) * +\ ,\ eq(f(W), 1)\ ,$
$eq(f(max(s(X), s(W))), 1)* \Rightarrow$

$24[0 : Res : 1.1, 23.0] \;\|\|\|\; eq(f(U), 1)\ ,\ eq(f(max(s(V), s(U))), 1)* \Rightarrow$
$eq(f(max(s(W), s(X))), 1)$

$25[0 : MRR : 4.2, 24.2] \;\|\|\|\; eq(f(U), 1)\ ,\ eq(f(max(s(V), s(U))), 1)* \Rightarrow$

$28[0 : Res : 21.1, 25.1] \;\|\|\|\; eq(f(U), 0)*\ ,\ eq(f(V), 1)* \Rightarrow$

$30[0 : MRR : 21.1, 28.1] \;\|\|\|\; eq(f(U), 0)* \Rightarrow$

$31[0 : MRR : 1.1, 30.0] \;\|\|\|\; \Rightarrow eq(f(U), 1)$

$32[0 : MRR : 25.1, 25.0, 31.0] \;\|\|\|\; \Rightarrow$

D.6 *g*NiA-schema Clause set for two Symbols and Four Occurrences

Clause set Without Term Schema (TPTP Syntax)

$1 :\ eq(f(A), 0) \mid eq(f(A), 1)$
$2 :\ le(A, A)$
$3 : \sim le(max(A, B), C) \mid le(A, C)$
$4 : \sim le(max(A, B), C) \mid le(B, C)$
$5 : \sim eq(f(A), 0) \mid\ \sim eq(f(B), 0) \mid\ \sim eq(f(C), 0) \mid\ \sim eq(f(D), 0) \mid$
$\sim le(s(B), A) \mid\ \sim le(s(C), B) \mid\ \sim le(s(D), C)$
$6 : \sim eq(f(A), 1) \mid\ \sim eq(f(B), 1) \mid\ \sim eq(f(C), 1) \mid\ \sim eq(f(D), 1) \mid$
$\sim le(s(B), A) \mid\ \sim le(s(C), B) \mid\ \sim le(s(D), C)$

Clause set with Term Schema (TPTP Syntax)

Note, this clause set does not require the definition of the max function because all the configuration of the max function required to refute the clause set are already constructed. Also, $Axyz$ where $x, y, z \in [0, 1]$ are the variables representing the 8 points required to construct the cube found in Fig. 10.1.

$1:\ eq(f(A),0) \mid eq(f(A),1)$

$2:\sim eq(f(A000),0) \mid\ \sim eq(f(max(s(A000),s(A100))),0) \mid$
$\sim eq(f(max(s(max(s(A000),s(A100))),s(max(s(A010),s(A110))))),0) \mid$
$\sim eq(f(max(s(max(s(max(s(A000),s(A100))),s(max(s(A010),s(A110))))),$
$s(max(s(max(s(A001),s(A101))),s(max(s(A011),s(A111))))))),0)$

$3:\sim eq(f(A010),0) \mid\ \sim eq(f(max(s(A010),s(A110))),0) \mid$
$\sim eq(f(max(s(max(s(A000),s(A100))),s(max(s(A010),s(A110))))),0) \mid$
$\sim eq(f(max(s(max(s(max(s(A000),s(A100))),s(max(s(A010),s(A110))))),$
$s(max(s(max(s(A001),s(A101))),s(max(s(A011),s(A111))))))),0)$

$4:\sim eq(f(A001),0) \mid\ \sim eq(f(max(s(A001),s(A101))),0) \mid$
$\sim eq(f(max(s(max(s(A001),s(A101))),s(max(s(A011),s(A111))))),0) \mid$
$\sim eq(f(max(s(max(s(max(s(A000),s(A100))),s(max(s(A010),s(A110))))),$
$s(max(s(max(s(A001),s(A101))),s(max(s(A011),s(A111))))))),0)$

$5:\sim eq(f(A011),0) \mid\ \sim eq(f(max(s(A011),s(A111))),0) \mid$
$\sim eq(f(max(s(max(s(A001),s(A101))),s(max(s(A011),s(A111))))),0) \mid$
$\sim eq(f(max(s(max(s(max(s(A000),s(A100))),s(max(s(A010),s(A110))))),$
$s(max(s(max(s(A001),s(A101))),s(max(s(A011),s(A111))))))),0)$

$6:\sim eq(f(A100),1) \mid\ \sim eq(f(max(s(A000),s(A100))),1) \mid$
$\sim eq(f(max(s(max(s(A000),s(A100))),s(max(s(A010),s(A110))))),1) \mid$
$\sim eq(f(max(s(max(s(max(s(A000),s(A100))),s(max(s(A010),s(A110))))),$
$s(max(s(max(s(A001),s(A101))),s(max(s(A011),s(A111))))))),1)$

$7:\sim eq(f(A110),1) \mid\ \sim eq(f(max(s(A010),s(A110))),1) \mid$
$\sim eq(f(max(s(max(s(A000),s(A100))),s(max(s(A010),s(A110))))),1) \mid$
$\sim eq(f(max(s(max(s(max(s(A000),s(A100))),s(max(s(A010),s(A110))))),$
$s(max(s(max(s(A001),s(A101))),s(max(s(A011),s(A111))))))),1)$

$8 : \sim eq(f(A101), 1) \mid \sim eq(f(max(s(A001), s(A101))), 1) \mid$
$\sim eq(f(max(s(max(s(A001), s(A101))), s(max(s(A011), s(A111))))), 1) \mid$
$\sim eq(f(max(s(max(s(max(s(A000), s(A100))), s(max(s(A010), s(A110))))),$
$s(max(s(max(s(A001), s(A101))), s(max(s(A011), s(A111))))))), 1)$

$5 : \sim eq(f(A111), 0) \mid \sim eq(f(max(s(A011), s(A111))), 0) \mid$
$\sim eq(f(max(s(max(s(A001), s(A101))), s(max(s(A011), s(A111))))), 0) \mid$
$\sim eq(f(max(s(max(s(max(s(A000), s(A100))), s(max(s(A010), s(A110))))),$
$s(max(s(max(s(A001), s(A101))), s(max(s(A011), s(A111))))))), 0)$

SPASS Refutation of Clause set with Term Schema (TPTP Syntax)

$1[0 : Inp] \|\| \Rightarrow eq(f(U), 1) \ , \ eq(f(U), 0)*$
$2[0 : Inp] \|\| eq(f(U), 0) \ , \ eq(f(max(s(U), s(V))), 0) \ ,$
$eq(f(max(s(max(s(U), s(V))), s(max(s(W), s(X))))), 0) \ ,$
$eq(f(max(s(max(s(max(s(U), s(V))), s(max(s(W), s(X))))),$
$s(max(s(max(s(Y), s(Z))), s(max(s(X1), s(X2))))))), 0)* \Rightarrow$
$3[0 : Inp] \|\| eq(f(U), 0) \ , \ eq(f(max(s(U), s(V))), 0) \ ,$
$eq(f(max(s(max(s(W), s(X))), s(max(s(U), s(V))))), 0) \ ,$
$eq(f(max(s(max(s(max(s(W), s(X))), s(max(s(U), s(V))))),$
$s(max(s(max(s(Y), s(Z))), s(max(s(X1), s(X2))))))), 0)* \Rightarrow$
$5[0 : Inp] \|\| eq(f(U), 0) \ , \ eq(f(max(s(U), s(V))), 0) \ ,$
$eq(f(max(s(max(s(W), s(X))), s(max(s(U), s(V))))), 0) \ ,$
$eq(f(max(s(max(s(max(s(Y), s(Z))), s(max(s(X1), s(X2))))),$
$s(max(s(max(s(W), s(X))), s(max(s(U), s(V))))))), 0)* \Rightarrow$
$6[0 : Inp] \|\| eq(f(U), 1) \ , \ eq(f(max(s(V), s(U))), 1) \ ,$
$eq(f(max(s(max(s(V), s(U))), s(max(s(W), s(X))))), 1) \ ,$
$eq(f(max(s(max(s(max(s(V), s(U))), s(max(s(W), s(X))))),$
$s(max(s(max(s(Y), s(Z))), s(max(s(X1), s(X2))))))), 1)* \Rightarrow$
$7[0 : Inp] \|\| eq(f(U), 1) \ , \ eq(f(max(s(V), s(U))), 1) \ ,$
$eq(f(max(s(max(s(W), s(X))), s(max(s(V), s(U))))), 1) \ ,$
$eq(f(max(s(max(s(max(s(W), s(X))), s(max(s(V), s(U))))),$
$s(max(s(max(s(Y), s(Z))), s(max(s(X1), s(X2))))))), 1)* \Rightarrow$
$9[0 : Inp] \|\| eq(f(U), 1) \ , \ eq(f(max(s(V), s(U))), 1) \ ,$
$eq(f(max(s(max(s(W), s(X))), s(max(s(V), s(U))))), 1) \ ,$
$eq(f(max(s(max(s(max(s(Y), s(Z))), s(max(s(X1), s(X2))))),$
$s(max(s(max(s(W), s(X))), s(max(s(V), s(U))))))), 1)* \Rightarrow$
$10[0 : Res : 1.1, 2.3] \|\| eq(f(U), 0) \ , \ eq(f(max(s(U), s(V))), 0) \ ,$
$eq(f(max(s(max(s(U), s(V))), s(max(s(W), s(X))))), 0) \Rightarrow$
$eq(f(max(s(max(s(max(s(U), s(V))), s(max(s(W), s(X))))),$
$s(max(s(max(s(Y), s(Z))), s(max(s(X1), s(X2))))))), 1)*$
$11[0 : Res : 1.1, 3.3] \|\| eq(f(U), 0) \ , \ eq(f(max(s(U), s(V))), 0) \ ,$

$eq(f(max(s(max(s(W),s(X))),s(max(s(U),s(V))))),0) \Rightarrow$
$eq(f(max(s(max(s(max(s(W),s(X))),s(max(s(U),s(V))))),$
$s(max(s(max(s(Y),s(Z))),s(max(s(X1),s(X2))))))),1)*$
$13[0:Res:1.1,5.3]$ $\|\|$ $eq(f(U),0)$, $eq(f(max(s(U),s(V))),0)$,
$eq(f(max(s(max(s(W),s(X))),s(max(s(U),s(V))))),0) \Rightarrow$
$eq(f(max(s(max(s(max(s(Y),s(Z))),s(max(s(X1),s(X2))))),$
$s(max(s(max(s(W),s(X))),s(max(s(U),s(V))))))),1)*$
$17[0:Res:13.3,6.3]$ $\|\|$ $eq(f(U),0)$, $eq(f(max(s(U),s(V))),0)$,
$eq(f(max(s(max(s(W),s(X))),s(max(s(U),s(V))))),0)*+$, $eq(f(Y),1)$,
$eq(f(max(s(Z),s(Y))),1)$, $eq(f(max(s(max(s(Z),s(Y))),s(max(s(X1),s(X2))))),1)* \Rightarrow$
$18[0:Res:1.1,17.2]$ $\|\|$ $eq(f(U),0)$, $eq(f(max(s(U),s(V))),0)$, $eq(f(W),1)$,
$eq(f(max(s(X),s(W))),1)$, $eq(f(max(s(max(s(X),s(W))),s(max(s(Y),s(Z))))),1)*$
$+$
$\Rightarrow eq(f(max(s(max(s(X1),s(X2))),s(max(s(U),s(V))))),1)*$
$22[0:Res:13.3,18.4]$ $\|\|$ $eq(f(U),0)$, $eq(f(max(s(U),s(V))),0)$,
$eq(f(max(s(max(s(W),s(X))),s(max(s(U),s(V))))),0)*+$, $eq(f(Y),0)$,
$eq(f(max(s(Y),s(Z))),0)$, $eq(f(max(s(X1),s(X2))),1)$,
$eq(f(max(s(max(s(X3),s(X4))),s(max(s(X1),s(X2))))),1)* \Rightarrow$
$eq(f(max(s(max(s(X5),s(X6))),s(max(s(Y),s(Z))))),1)*$
$50[0:Res:1.1,22.2]$ $\|\|$ $eq(f(U),0)$, $eq(f(max(s(U),s(V))),0)$, $eq(f(W),0)$
, $eq(f(max(s(W),s(X))),0)$, $eq(f(max(s(Y),s(Z))),1)$,
$eq(f(max(s(max(s(X1),s(X2))),s(max(s(Y),s(Z))))),1)* \Rightarrow$
$eq(f(max(s(max(s(X3),s(X4))),s(max(s(U),s(V))))),1)*$,
$eq(f(max(s(max(s(X5),s(X6))),s(max(s(W),s(X))))),1)*$
$51[0:Con:50.2]$ $\|\|$ $eq(f(U),0)$, $eq(f(max(s(U),s(V))),0)$,
$eq(f(max(s(W),s(X))),1)$, $eq(f(max(s(max(s(Y),s(Z))),s(max(s(W),s(X))))),1)*$
$+$
$\Rightarrow eq(f(max(s(max(s(X1),s(X2))),s(max(s(U),s(V))))),1)*$
$53[0:Res:11.3,51.3]$ $\|\|$ $eq(f(U),0)$, $eq(f(max(s(U),s(V))),0)$,
$eq(f(max(s(max(s(W),s(X))),s(max(s(U),s(V))))),0)*+$, $eq(f(Y),0)$,
$eq(f(max(s(Y),s(Z))),0)$, $eq(f(max(s(max(s(X1),s(X2))),s(max(s(X3),s(X4))))),1)*$
$\Rightarrow eq(f(max(s(max(s(X5),s(X6))),s(max(s(Y),s(Z))))),1)*$
$56[0:Res:1.1,53.2]$ $\|\|$ $eq(f(U),0)$, $eq(f(max(s(U),s(V))),0)$, $eq(f(W),0)$
, $eq(f(max(s(W),s(X))),0)$, $eq(f(max(s(max(s(Y),s(Z))),s(max(s(X1),s(X2))))),1)* \Rightarrow$
$eq(f(max(s(max(s(X3),s(X4))),s(max(s(U),s(V))))),1)*$,
$eq(f(max(s(max(s(X5),s(X6))),s(max(s(W),s(X))))),1)*$
$57[0:Con:56.2]$ $\|\|$ $eq(f(U),0)$, $eq(f(max(s(U),s(V))),0)$,
$eq(f(max(s(max(s(W),s(X))),s(max(s(Y),s(Z))))),1)*+ \Rightarrow$
$eq(f(max(s(max(s(X1),s(X2))),s(max(s(U),s(V))))),1)*$
$62[0:Res:13.3,57.2]$ $\|\|$ $eq(f(U),0)$, $eq(f(max(s(U),s(V))),0)$,
$eq(f(max(s(max(s(W),s(X))),s(max(s(U),s(V))))),0)*+$, $eq(f(Y),0)$,
$eq(f(max(s(Y),s(Z))),0) \Rightarrow eq(f(max(s(max(s(X1),s(X2))),s(max(s(Y),s(Z))))),1)*$
$63[0:Res:1.1,62.2]$ $\|\|$ $eq(f(U),0)$, $eq(f(max(s(U),s(V))),0)$, $eq(f(W),0)$

$, \ eq(f(max(s(W),s(X))),0) \Rightarrow eq(f(max(s(max(s(Y),s(Z))),s(max(s(U),s(V))))),1){*}\ ,$
$eq(f(max(s(max(s(X1),s(X2))),s(max(s(W),s(X))))),1){*}$
$64[0:Con:63.2]\ \|\|\|\|\ eq(f(U),0)\ ,\ eq(f(max(s(U),s(V))),0) \Rightarrow$
$eq(f(max(s(max(s(W),s(X))),s(max(s(U),s(V))))),1){*}$
$65[0:Res:64.2,6.3]\ \|\|\|\|\ eq(f(max(s(U),s(V))),0)\ ,$
$eq(f(max(s(max(s(U),s(V))),s(max(s(W),s(X))))),0){*}{+}\ ,$
$eq(f(Y),1)\ ,\ eq(f(max(s(Z),s(Y))),1)\ ,$
$eq(f(max(s(max(s(Z),s(Y))),s(max(s(X1),s(X2))))),1){*} \Rightarrow$
$86[0:Res:1.1,65.1]\ \|\|\|\|\ eq(f(max(s(U),s(V))),0)\ ,\ eq(f(W),1)\ ,$
$eq(f(max(s(X),s(W))),1)\ ,\ eq(f(max(s(max(s(X),s(W))),s(max(s(Y),s(Z))))),1){*}$
$+$
$\Rightarrow eq(f(max(s(max(s(U),s(V))),s(max(s(X1),s(X2))))),1){*}$
$90[0:Res:64.2,86.3]\ \|\|\|\|\ eq(f(U),0)\ ,\ eq(f(max(s(U),s(V))),0){*}{+}\ ,$
$eq(f(max(s(W),s(X))),0)\ ,\ eq(f(Y),1)\ ,\ eq(f(max(s(Z),s(Y))),1){*}$
$\Rightarrow eq(f(max(s(max(s(W),s(X))),s(max(s(X1),s(X2))))),1){*}$
$91[0:Res:1.1,90.1]\ \|\|\|\|\ eq(f(U),0)\ ,\ eq(f(max(s(V),s(W))),0)\ ,\ eq(f(X),1)\ ,$
$eq(f(max(s(Y),s(X))),1){*} \Rightarrow eq(f(max(s(U),s(Z))),1){*}\ ,$
$eq(f(max(s(max(s(V),s(W))),s(max(s(X1),s(X2))))),1){*}$
$92[0:Con:91.0]\ \|\|\|\|\ eq(f(max(s(U),s(V))),0){+}eq(f(W),1)\ ,\ eq(f(max(s(X),s(W))),1){*} \Rightarrow$
$eq(f(max(s(max(s(U),s(V))),s(max(s(Y),s(Z))))),1){*}$
$93[0:Res:1.1,92.0]\ \|\|\|\|\ eq(f(U),1)\ ,\ eq(f(max(s(V),s(U))),1){*}{+} \Rightarrow$
$eq(f(max(s(W),s(X))),1)\ ,\ eq(f(max(s(max(s(W),s(X))),s(max(s(Y),s(Z))))),1){*}$
$94[0:Res:10.3,93.1]\ \|\|\|\|\ eq(f(U),0)\ ,\ eq(f(max(s(U),s(V))),0)\ ,$
$eq(f(max(s(max(s(U),s(V))),s(max(s(W),s(X))))),0){*}{+}\ ,$
$eq(f(max(s(max(s(Y),s(Z))),s(max(s(X1),s(X2))))),1){*} \Rightarrow eq(f(max(s(X3),s(X4))),1)\ ,$
$eq(f(max(s(max(s(X3),s(X4))),s(max(s(X5),s(X6))))),1){*}$
$102[0:Res:1.1,94.2]\ \|\|\|\|\ eq(f(U),0)\ ,\ eq(f(max(s(U),s(V))),0)\ ,$
$eq(f(max(s(max(s(W),s(X))),s(max(s(Y),s(Z))))),1){*}{+} \Rightarrow$
$eq(f(max(s(max(s(U),s(V))),s(max(s(X1),s(X2))))),1){*}\ ,$
$eq(f(max(s(X3),s(X4))),1)\ ,\ eq(f(max(s(max(s(X3),s(X4))),s(max(s(X5),s(X6))))),1){*}$
$106[0:Res:64.2,102.2]\ \|\|\|\|\ eq(f(U),0)\ ,\ eq(f(max(s(U),s(V))),0){*}\ ,\ eq(f(W),0)\ ,$
$eq(f(max(s(W),s(X))),0) \Rightarrow eq(f(max(s(max(s(W),s(X))),s(max(s(Y),s(Z))))),1){*}\ ,$
$eq(f(max(s(X1),s(X2))),1)\ ,\ eq(f(max(s(max(s(X1),s(X2))),s(max(s(X3),s(X4))))),1){*}$
$107[0:Con:106.0]\ \|\|\|\|\ eq(f(U),0)\ ,\ eq(f(max(s(U),s(V))),0){+} \Rightarrow$
$eq(f(max(s(max(s(U),s(V))),s(max(s(W),s(X))))),1){*}\ ,\ eq(f(max(s(Y),s(Z))),1)\ ,$
$eq(f(max(s(max(s(Y),s(Z))),s(max(s(X1),s(X2))))),1){*}$
$108[0:Res:1.1,107.1]\ \|\|\|\|\ eq(f(U),0) \Rightarrow eq(f(max(s(U),s(V))),1)\ ,$
$eq(f(max(s(max(s(U),s(V))),s(max(s(W),s(X))))),1){*}\ ,$
$eq(f(max(s(Y),s(Z))),1)\ ,\ eq(f(max(s(max(s(Y),s(Z))),s(max(s(X1),s(X2))))),1){*}$
$109[0:Con:108.3]\ \|\|\|\|\ eq(f(U),0) \Rightarrow eq(f(max(s(U),s(V))),1)\ ,$
$eq(f(max(s(max(s(U),s(V))),s(max(s(W),s(X))))),1){*}$
$111[0:Res:109.2,93.1]\ \|\|\|\|\ eq(f(U),0)\ ,\ eq(f(max(s(V),s(W))),1){*}{+} \Rightarrow$
$eq(f(max(s(U),s(X))),1){*}\ ,\ eq(f(max(s(Y),s(Z))),1)\ ,$

$eq(f(max(s(max(s(Y),s(Z))),s(max(s(X1),s(X2))))),1)*$
$119[0:Res:109.2,111.1]$ |||| $eq(f(U),0)$, $eq(f(V),0) \Rightarrow$
$eq(f(max(s(U),s(W))),1)*$, $eq(f(max(s(V),s(X))),1)*$, $eq(f(max(s(Y),s(Z))),1)$,
$eq(f(max(s(max(s(Y),s(Z))),s(max(s(X1),s(X2))))),1)*$
$120[0:Con:119.1]$ |||| $eq(f(U),0)+ \Rightarrow eq(f(max(s(U),s(V))),1)*$,
$eq(f(max(s(W),s(X))),1)$, $eq(f(max(s(max(s(W),s(X))),s(max(s(Y),s(Z))))),1)*$
$122[0:Res:1.1,120.0]$ |||| $\Rightarrow eq(f(U),1)$, $eq(f(max(s(U),s(V))),1)*$
, $eq(f(max(s(W),s(X))),1)$, $eq(f(max(s(max(s(W),s(X))),s(max(s(Y),s(Z))))),1)*$
$123[0:Con:122.0]$ |||| $\Rightarrow eq(f(max(s(U),s(V))),1)$,
$eq(f(max(s(max(s(U),s(V))),s(max(s(W),s(X))))),1)*$
$127[0:Res:123.1,9.3]$ |||| $eq(f(U),1)$, $eq(f(max(s(V),s(U))),1)$,
$eq(f(max(s(max(s(W),s(X))),s(max(s(V),s(U))))),1)* \Rightarrow$
$eq(f(max(s(max(s(Y),s(Z))),s(max(s(X1),s(X2))))),1)*$
$128[0:MRR:7.3,127.3]$ |||| $eq(f(U),1)$, $eq(f(max(s(V),s(U))),1)$,
$eq(f(max(s(max(s(W),s(X))),s(max(s(V),s(U))))),1)* \Rightarrow$
$136[0:Res:123.1,128.2]$ |||| $eq(f(U),1)$, $eq(f(max(s(V),s(U))),1)* \Rightarrow eq(f(max(s(W),s(X))),1)*$
$137[0:MRR:128.2,136.2]$ |||| $eq(f(U),1)$, $eq(f(max(s(V),s(U))),1)* \Rightarrow$
$143[0:Res:123.1,137.1]$ |||| $eq(f(max(s(U),s(V))),1)* \Rightarrow eq(f(max(s(W),s(X))),1)*$
$144[0:MRR:123.1,143.0]$ |||| $\Rightarrow eq(f(max(s(U),s(V))),1)*$
$146[0:MRR:137.1,144.0]$ |||| $eq(f(U),1)* \Rightarrow$
$147[0:UnC:146.0,144.0]$ |||| $\Rightarrow$

D.7 *g*NiA-schema Clause set for three Symbols and 3 Occurrences

Clause set Without Term Schema (TPTP Syntax)

$1: \ eq(f(A),0) \mid eq(f(A),1) \mid eq(f(A),2)$
$2: \ le(A,A)$
$3: \sim le(max(A,B),C) \mid le(A,C)$
$4: \sim le(max(A,B),C) \mid le(B,C)$
$5: \sim eq(f(A),0) \mid \ \sim eq(f(B),0) \mid \ \sim eq(f(C),0) \mid$
$\sim le(s(B),A) \mid \ \sim le(s(C),B)$
$6: \sim eq(f(A),1) \mid \ \sim eq(f(B),1) \mid \ \sim eq(f(C),1) \mid$
$\sim le(s(B),A) \mid \ \sim le(s(C),B)$ $7: \sim eq(f(A),2) \mid \ \sim eq(f(B),2) \mid \ \sim eq(f(C),2) \mid$
$\sim le(s(B),A) \mid \ \sim le(s(C),B)$

Clause set with Term Schema (TPTP Syntax)

Note, this clause set does not require the definition of the max function because all the configuration of the max function required to refute the clause set are already constructed. Also, Axy where $x,y \in [0,1,2]$ are the variables representing the 9 points required to construct the clause set.

$1: \; eq(f(A), 0) \mid eq(f(A), 1) \mid eq(f(A), 2)$
$2: \; \sim eq(f(A00), 0) \mid \; \sim eq(f(max(s(A00), max(s(A01), s(A02)))), 0) \mid$
$\sim eq(f(max(s(max(s(A00), max(s(A01), s(A02)))),$
$max(s(max(s(A10), max(s(A11), s(A12)))),$
$s(max(s(A20), max(s(A21), s(A22))))))), 0)$
$3: \; \sim eq(f(A10), 0) \mid \; \sim eq(f(max(s(A10), max(s(A11), s(A12)))), 0) \mid$
$\sim eq(f(max(s(max(s(A00), max(s(A01), s(A02)))),$
$max(s(max(s(A10), max(s(A11), s(A12)))),$
$s(max(s(A20), max(s(A21), s(A22))))))), 0)$
$4: \; \sim eq(f(A20), 0) \mid \sim eq(f(max(s(A20), max(s(A21), s(A22)))), 0) \mid$
$\sim eq(f(max(s(max(s(A00), max(s(A01), s(A02)))),$
$max(s(max(s(A10), max(s(A11), s(A12)))),$
$s(max(s(A20), max(s(A21), s(A22))))))), 0)$
$5: \; \sim eq(f(A01), 1) \mid \; \sim eq(f(max(s(A00), max(s(A01), s(A02)))), 1) \mid$
$\sim eq(f(max(s(max(s(A00), max(s(A01), s(A02)))),$
$max(s(max(s(A10), max(s(A11), s(A12)))),$
$s(max(s(A20), max(s(A21), s(A22))))))), 1)$
$6: \; \sim eq(f(A11), 1) \mid \; \sim eq(f(max(s(A10), max(s(A11), s(A12)))), 1) \mid$
$\sim eq(f(max(s(max(s(A00), max(s(A01), s(A02)))),$
$max(s(max(s(A10), max(s(A11), s(A12)))),$
$s(max(s(A20), max(s(A21), s(A22))))))), 1)$
$7: \; \sim eq(f(A21), 1) \mid \; \sim eq(f(max(s(A20), max(s(A21), s(A22)))), 1) \mid$
$\sim eq(f(max(s(max(s(A00), max(s(A01), s(A02))))$
$, max(s(max(s(A10), max(s(A11), s(A12)))),$
$s(max(s(A20), max(s(A21), s(A22))))))), 1)$
$8: \; \sim eq(f(A02), 2) \mid \; \sim eq(f(max(s(A00), max(s(A01), s(A02)))), 2) \mid$
$\sim eq(f(max(s(max(s(A00), max(s(A01), s(A02)))),$
$max(s(max(s(A10), max(s(A11), s(A12)))),$
$s(max(s(A20), max(s(A21), s(A22))))))), 2)$
$9: \; \sim eq(f(A12), 2) \mid \; \sim eq(f(max(s(A10), max(s(A11), s(A12)))), 2) \mid$
$\sim eq(f(max(s(max(s(A00), max(s(A01), s(A02)))),$
$max(s(max(s(A10), max(s(A11), s(A12))))$
$, s(max(s(A20), max(s(A21), s(A22))))))), 2)$
$10: \; \sim eq(f(A22), 2) \mid \; \sim eq(f(max(s(A20), max(s(A21), s(A22)))), 2) \mid$
$\sim eq(f(max(s(max(s(A00), max(s(A01), s(A02)))),$
$max(s(max(s(A10), max(s(A11), s(A12)))),$
$s(max(s(A20), max(s(A21), s(A22))))))), 2)$

SPASS was able to find a refutation of this clause set as well, however, the length of the refutation forced us not to include it in this work. The SPASS refutation has 5784 lines and a depth of 62. We conjecture that, from this derived schematic clause set pattern, SPASS as well as other theorem provers will not be able to refute a clause set for higher instances. Regrettably when the refutations get so large it is also very hard to parse the output and derive a meaningful result from it. To give an example of how hard the clause set is to read, The following line from the refutation is the largest clause SPASS derived in order to refute the clause set.

$15654[6 : Res : 15648.7, 15577.2] \;||\; eq(f(U), 0)\; eq(f(V), 0)\; eq(f(W), 0)$
$eq(f(X), 0)- > eq(f(max(s(U), max(s(Y), s(Z)))), 2)\; eq(f(max(s(U), max(s(Y), s(Z)))), 1)$
$eq(f(max(s(max(s(X1), max(s(X2), s(X3)))), max(s(max(s(U), max(s(Y), s(Z)))),$
$s(max(s(X4), max(s(X5), s(X6))))))), 2)*$
$eq(f(max(s(V), max(s(X7), s(X8)))), 2)\; eq(f(max(s(V), max(s(X7), s(X8)))), 1)*$
$eq(f(max(s(W), max(s(X9), s(X10)))), 2) eq(f(max(s(W), max(s(X9), s(X10)))), 1)$
$eq(f(max(s(max(s(X11), max(s(X12), s(X13)))), max(s(max(s(W), max(s(X9), s(X10)))),$
$s(max(s(X14), max(s(X15), s(X16))))))), 2)$
$eq(f(max(s(X), max(s(X17), s(X18)))), 2)\; eq(f(max(s(X), max(s(X17), s(X18)))), 1)$
$eq(f(max(s(max(s(X), max(s(X17), s(X18)))), max(s(max(s(max(s(X11), max(s(X12), s(X13)))),$
$max(s(max(s(W), max(s(X9), s(X10)))), s(max(s(X14), max(s(X15), s(X16))))))),$
$s(max(s(X19), max(s(X20), s(X21))))))), 2) * .$

This clause has 14 literals, four in the antecedent, 10 in the consequent and 27 different variables. When a refutation requires such massive clauses, it becomes clause to impossible for a human to take this information and abstract a recursive pattern from it. This issue plague the analysis of the Früstenberg proof using second order arithmetic because the clauses had too many literals and too many variables. It became impossible to read through.

Bibliography

[1] Peter Aczel. An introduction to inductive definitions. In Jon Barwise, editor, *HANDBOOK OF MATHEMATICAL LOGIC*, volume 90 of *Studies in Logic and the Foundations of Mathematics*, pages 739 – 782. Elsevier, 1977.

[2] M. Adler and N. Immerman. An n! lower bound on formula size. In *Logic in Computer Science, 2001. Proceedings. 16th Annual IEEE Symposium on*, pages 197–206, 2001.

[3] Vincent Aravantinos, Ricardo Caferra, and Nicolas Peltier. A decidable class of nested iterated schemata. In *Proceedings of the 5th international conference on Automated Reasoning*, IJCAR'10, pages 293–308, Berlin, Heidelberg, 2010. Springer-Verlag.

[4] Vincent Aravantinos, Ricardo Caferra, and Nicolas Peltier. Decidability and undecidability results for propositional schemata. *Journal of Artificial Intelligence Research*, 40(1):599–656, January 2011.

[5] Vincent Aravantinos, Ricardo Caferra, and Nicolas Peltier. Linear temporal logic and propositional schemata, back and forth. In *Proceedings of the 2011 Eighteenth International Symposium on Temporal Representation and Reasoning*, TIME '11, pages 80–87, Washington, DC, USA, 2011. IEEE Computer Society.

[6] Vincent Aravantinos, Mnacho Echenim, and Nicolas Peltier. A resolution calculus for first-order schemata. *Fundamenta Informaticae*, 2013.

[7] Matthias Baaz. Note on the generalization of calculations. *Theoretical Computer Science*, 224(1–2):3 – 11, 1999.

[8] Matthias Baaz, Stefan Hetzl, Alexander Leitsch, Clemens Richter, and Hendrik Spohr. Cut-elimination: Experiments with ceres. In Franz Baader and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning*, volume 3452 of *Lecture Notes in Computer Science*, pages 481–495. Springer Berlin Heidelberg, 2005.

[9] Matthias Baaz, Stefan Hetzl, Alexander Leitsch, Clemens Richter, and Hendrik Spohr. Proof transformation by ceres. In JonathanM. Borwein and WilliamM. Farmer, editors, *Mathematical Knowledge Management*, volume 4108 of *Lecture Notes in Computer Science*, pages 82–93. Springer Berlin Heidelberg, 2006.

[10] Matthias Baaz, Stefan Hetzl, Alexander Leitsch, Clemens Richter, and Hendrik Spohr. Proof transformation by ceres. In Jonathan M. Borwein and William M. Farmer, editors, *Mathematical Knowledge Management (MKM) 2006*, volume 4108 of *Lecture Notes in Artificial Intelligence*, pages 82–93. Springer, 2006.

[11] Matthias Baaz, Stefan Hetzl, Alexander Leitsch, Clemens Richter, and Hendrik Spohr. Ceres: An analysis of Fürstenberg's proof of the infinity of primes. *Theoretical Computer Science*, 403(2-3):160–175, August 2008.

[12] Matthias Baaz and Alexander Leitsch. On skolemization and proof complexity. *Fundamenta Informaticae*, 20(4):353–379, December 1994.

[13] Matthias Baaz and Alexander Leitsch. Fast cut-elimination by projection. In Dirk van Dalen and Marc Bezem, editors, *Computer Science Logic*, volume 1258 of *Lecture Notes in Computer Science*, pages 18–33. Springer Berlin Heidelberg, 1997.

[14] Matthias Baaz and Alexander Leitsch. Cut-elimination and redundancy-elimination by resolution. *Journal of Symbolic Computation*, 29:149–176, 2000.

[15] Matthias Baaz and Alexander Leitsch. *Methods of Cut-Elimination*. Springer Publishing Company, Incorporated, 2013.

[16] Matthias Baaz and Richard Zach. Short proofs of tautologies using the schema of equivalence. In Egon Börger, Yuri Gurevich, and Karl Meinke, editors, *Computer Science Logic*, volume 832 of *Lecture Notes in Computer Science*, pages 33–35. Springer Berlin Heidelberg, 1994.

[17] Anthony Bonato, Peter J Cameron, Dejan Delić, and Stéphan Thomassé. Generalized pigeonhole properties of graphs and oriented graphs. *European Journal of Combinatorics*, 23(3):257 – 274, 2002.

[18] E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem*. Springer, 1997.

[19] J.C. Bradfield. The modal mu-calculus alternation hierarchy is strict. In Ugo Montanari and Vladimiro Sassone, editors, *CONCUR '96: Concurrency Theory*, volume 1119 of *Lecture Notes in Computer Science*, pages 233–246. Springer Berlin Heidelberg, 1996.

[20] AaronR. Bradley, Zohar Manna, and HennyB. Sipma. What's decidable about arrays? In E.Allen Emerson and KedarS. Namjoshi, editors, *Verification, Model Checking, and Abstract Interpretation*, volume 3855 of *Lecture Notes in Computer Science*, pages 427–442. Springer Berlin Heidelberg, 2006.

[21] James Brotherston. Cyclic proofs for first-order logic with inductive definitions. In Bernhard Beckert, editor, *Automated Reasoning with Analytic Tableaux and Related Methods*, volume 3702 of *Lecture Notes in Computer Science*, pages 78–92. Springer Berlin Heidelberg, 2005.

[22] Sam Buss and Toniann Pitassi. Resolution and the weak pigeonhole principle. In Mogens Nielsen and Wolfgang Thomas, editors, *Computer Science Logic*, volume 1414 of *Lecture Notes in Computer Science*, pages 149–156. Springer Berlin Heidelberg, 1998.

[23] Samuel R. Buss. Polynomial size proofs of the propositional pigeonhole principle. *Journal of Symbolic Logic*, 52:916–927, 1987.

[24] Samuel R. Buss and Gyorgy Turan. Resolution proofs of generalized pigeonhole principles. *Theoretical Computer Science*, 62(3):311 – 317, 1988.

[25] David Cerna. A tableaux-based decision procedure for multi-parameter propositional schemata. In StephenM. Watt, JamesH. Davenport, AlanP. Sexton, Petr Sojka, and Josef Urban, editors, *Intelligent Computer Mathematics*, volume 8543 of *Lecture Notes in Computer Science*, pages 61–75. Springer International Publishing, 2014.

[26] Hubert Comon. Inductionless induction. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning (in 2 volumes)*, pages 913–962. Elsevier and MIT Press, 2001.

[27] D. Cooper. Theorem proving in arithmetic without multiplication. In *Machine Intelligence*, 1972.

[28] John Corcoran. Schemata: The concept of schema in the history of logic. *Bulletin of Symbolic Logic*, (2):219–240.

[29] cvetan Dunchev. *Automation of cut-elimination in proof schemata*. PhD thesis, Technical University of Vienna, 2012.

[30] Cvetan Dunchev, Alexander Leitsch, Mikheil Rukhaia, and Daniel Weller. Ceres for first-order schemata. *CoRR*, abs/1303.4257, 2013.

[31] Cvetan Dunchev, Alexander Leitsch, Mikheil Rukhaia, and Daniel Weller. Cut-elimination and proof schemata. *Journal of Language, Logic, and Computation*, 2014.

[32] Egon, Börger, Erich Grädel, and Yuri Gurevich. The classical decision problem (perspectives in mathematical logic). *Bulletin of the London Mathematical Society*, 30:317–335, 5 1998.

[33] Christopher J. Fewster and Daniel Siemssen. Enumerating permutations by their run structure. 2014. arXiv:1403.1723 [math.CO].

[34] Gerhard Gentzen. Untersuchungen über das logische Schließen I. *Mathematische Zeitschrift*, 39(1):176–210, December 1935.

[35] Gerhard Gentzen. Fusion of several complete inductions. In M.E. Szabo, editor, *The Collected Papers of Gerhard Gentzen*, volume 55 of *Studies in Logic and the Foundations of Mathematics*, pages 309 – 311. Elsevier, 1969.

[36] Jürgen Giesl and Deepak Kapur. Decidable classes of inductive theorems. In *Proceedings of the First International Joint Conference on Automated Reasoning*, IJCAR ’01, pages 469–484, London, UK, UK, 2001. Springer-Verlag.

[37] Stefan Hetzl, Alexander Leitsch, and Daniel Weller. Ceres in higher-order logic. *Annals of Pure and Applied Logic*, 162(12):1001–1034, 2011.

[38] Stefan Hetzl, Alexander Leitsch, Daniel Weller, and Bruno Woltzenlogel Paleo. Herbrand sequent extraction. In *IN INTELLIGENT COMPUTER MATHEMATICS*, pages 462–477. Springer, 2008.

[39] Deepak Kapur and Mahadavan Subramaniam. Extending decision procedures with induction schemes. In David McAllester, editor, *Automated Deduction - CADE-17*, volume 1831 of *Lecture Notes in Computer Science*, pages 324–345. Springer Berlin Heidelberg, 2000.

[40] Laura Kovács and Andrei Voronkov. First-order theorem proving and vampire. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification*, volume 8044 of *Lecture Notes in Computer Science*, pages 1–35. Springer Berlin Heidelberg, 2013.

[41] Jan Krajíček and Pavel Pudlák. The number of proof lines and the size of proofs in first order logic. *Archive for Mathematical Logic*, 27(1):69–84, 1988.

[42] Alexander Leitsch. The resolution calculus. *Journal of Logic, Language and Information*, 7(4):499–502, October 1998.

[43] Alexander Leitsch, Giselle Reis, and Bruno Woltzenlogel Paleo. Towards CERes in intuitionistic logic. In Patrick Cégielski and Arnaud Durand, editors, *Computer Science Logic (CSL’12) - 26th International Workshop/21st Annual Conference of the EACSL*, volume 16 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 485–499, Dagstuhl, Germany, 2012. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[44] Alexis Maciel, Toniann Pitassi, and Alan R. Woods. A new proof of the weak pigeonhole principle. *Journal of Computer and System Sciences*, 64(4):843 – 872, 2002.

[45] W. McCune. Prover9 and mace4. `http://www.cs.unm.edu/~mccune/prover9/`, 2005–2010.

[46] William Mccune. Solution of the robbins problem. *Journal of Automated Reasoning*, 19(3):263–276, 1997.

[47] Raymond Mcdowell and Dale Miller. Cut-elimination for a logic with definitions and induction. *Theoretical Computer Science*, 232:2000, 1997.

[48] V.P. Orevkov. Proof schemata in Hilbert-type axiomatic theories. *Journal of Soviet Mathematics*, 55(2):1610–1620, 1991.

[49] R. J. Parikh. Some results on the length of proofs. *Transactions of the American Mathematical Society*, 177:pp. 29–36, 1973.

[50] Toniann Pitassi, Paul Beame, and Russell Impagliazzo. Exponential lower bounds for the pigeonhole principle. *computational complexity*, 3(2):97–140, 1993.

[51] AlexanderA. Razborov. Proof complexity of pigeonhole principles. In Werner Kuich, Grzegorz Rozenberg, and Arto Salomaa, editors, *Developments in Language Theory*, volume 2295 of *Lecture Notes in Computer Science*, pages 100–116. Springer Berlin Heidelberg, 2002.

[52] Clemens Richter. *Proof Transformations by Resolution*. PhD thesis, Technical University of Vienna, 2006.

[53] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, January 1965.

[54] Gaisi Takeuti. *Proof Theory*, volume 81 of *Studies in logic and the foundations of mathematics*. American Elsevier Pub., 1975.

[55] Christian Urban. *Classical Logic and Computation*. PhD thesis, University of Cambridge, 2000.

[56] Christoph Weidenbach, Dilyana Dimova, Arnaud Fietzke, Rohit Kumar, Martin Suda, and Patrick Wischnewski. Spass version 3.5. In *Proceedings of the 22Nd International Conference on Automated Deduction*, CADE-22, pages 140–145, Berlin, Heidelberg, 2009. Springer-Verlag.

Index