CADE-21

The 21st Conference on Automated Deduction

Workshop on Disproving - Non-Theorems, Non-Validity, Non-Provability

(DISPROVING'07)

Editors:

Wolfgang Ahrendt, Peter Baumgartner, Hans de Nivelle

Bremen, Germany, July 16th 2007

CADE-21 Organization:

Conference Chair: Michael Kohlhase (Jacobs University Bremen)
Program Chair: Frank Pfenning (Carnegie Mellon University)
Workshop Chair: Christoph Benzmüller (University of Cambridge)
Local Organization: Event4 Event Management

Preface

The *Disproving Workshop* was held at the 21st International Conference on Automated Deduction, in Bremen, Germany. The name *automated theorem proving* or *automated deduction* derives from the fact that the field traditionally focused on the art of automatically finding proofs. Initially, researchers were mainly motivated by the wish to build computer systems that could automatically solve hard, mathematical problems. When searching for a very hard proof, it is quite acceptable for a system to eat up all resources and to never to give up. After all that is what we, researchers are also doing all the time.

However in the last years, one has become aware of the fact that for many applications, one needs to take more of an engineer's approach. In particular, one needs to be aware of resources. In order to use resources efficiently, it is essential to be able to efficiently recognize non-theorems. As an example, consider a situation where an automated theorem proving system is used as assistant for automatically solving subtasks in a larger, interactive project. In this context, the requirements to the automated theorem prover are quite different than in mathematics. In case, the user is working on a faulty conjecture, he should find out as early as possible. In addition, in case it cannot find a proof, the prover should provide as much information as possible, so that the user can correct the conjecture. The papers collected in this volume address this topic from different angles.

Our invited speaker, Cesare Tinelli, starts by speaking about satisfiability modulo theories. This is the problem of determining satisfiability of first-order formulas with respect to a logical background theory T. He will also address the problem of the generation of adequate models in case a first-order formula cannot be proven.

Alan Bundy discusses a framework in which ontologies can be repaired. Concretely, he studies the question of how an inconsistent ontology, (consisting of a theory part and an observation part) can be repaired, so that the theory part is no longer inconsistent with the observation part. This extends the range of disproving, which clearly contains the problem of repairing non-theorems, to also include the problem of repairing inconsistent ontologies.

Arjeh Cohen, Jan Willem Knopper and Scott H. Murray address the problem of generating proofs of graph non-isomorphism. The motivation is as follows: Nowadays, with the increasing use of remote computing, questions about graph-isomorphism are often delegated over the Internet to implementations written by experts (or people who claim to be). This raises questions of trust and reliability. In case the remote algorithm says 'yes, the graphs are isomorphic', one can demand that it will return the isomorphism. In this paper, the other side is discussed: What proof the algorithm can give back in case of non-isomorphism.

Our second invited speaker, Koen Claessen, observes a paradigm shift in automated theorem proving (ATP): among other aspects this shift involves that proof tasks for ATP systems are nowadays often generated from other formalisms and are quite different wrt. size, difficulty, redundancy etc than traditional "hand-crafted" problems. Koen argues this paradigm shift is paralleled by changing needs from an also changing user community, and ATP systems can be really useful then only if they can provide reasons for failure to find a proof, e.g. by (approximating) counter examples.

Jan Otop disproved two related conjectures that are in the TPTP, using a specialized algorithm. A right alternative ring is a ring in which for multiplication, the associative law is weakened to $\forall xy\ x(yy) = (xy)y$. The disproven conjectures are: Does for weak alternative rings hold that $2((xy)z - x(yz))^3 = 0$, and $6((xy)z - x(yz))^3 = 0$?

D. Galmiche en D. Larchey-Wendling and Y. Salhi study profs and counter models in Gödel-Dummett Logics. Gödel-Dummett logics are logics between classical and intuitionistic logic. The authors develop a new approach for deciding hypersequents through a kind of semantic graph called *bicolored graph.* Validity can be characterized through a particular kind of chain in this graph.

Hans de Nivelle studies redundancy in geometric resolution. He does this through a framework of proof permutations somewhat similar to those that are used in constructive proofs of cut elimination. Geometric resolution is a proof procedure that attempts to refute a set of formulas by exhaustively trying to construct a model for them. If a model is found, then the original hypothesis has been disproven, if no model can be found, then the original hypothesis is proven.

We are indebted to the members of the program committee for their reviewing efforts. We made use of the *EasyChair* conference management by Andrei Voronkov, which simplified the effort of organizing the reviewing process. Special thanks also deserve our invited speakers Cesare Tinelli and Koen Claessen (the former being joint invited speaker with the Verify workshop).

Wolfgang Ahrendt,
Peter Baumgartner,
Hans de Nivelle

Members of the program committee:

- Wolfgang Ahrendt, Chalmers University of Technology, Göteborg, Sweden.
- Franz Baader, Technische Universität Dresden, Germany.
- Peter Baumgartner, NICTA, Canberra, Australia.
- Simon Colton, Imperial College, London, UK.
- Chris Fermüller, Technische Universität Wien, Vienna, Austria.
- Bernhard Gramlich, Technische Universität Wien, Vienna, Austria.
- William McCune, University of New Mexico, USA.
- Hans de Nivelle, University of Wroclaw, Wroclaw, Poland.
- Michael Norrish, NICTA, Canberra, Australia.

- Silvio Ranise, LORIA and INRIA-Lorraine, Villers-les-Nancy, France.
- Renate Schmidt, University of Manchester, Manchester, UK.
- Carsten Schürmann, IT University of Copenhagen, Copenhagen, Denmark.
- Graham Steel, Centre for Intelligent Systems and their Applications, School of Informatics, Edinburgh, UK.

Contents

Trends and Challenges in Satisfiability Modulo Theories

Cesare Tinelli[⋆]

Department of Computer Science
The University of Iowa
tinelli@cs.uiowa.edu

Satisfiability Modulo Theories (SMT) is concerned with the problem of determining the satisfiability of first-order formulas with respect to a given logical theory T. A distinguishing feature of SMT is the use of inference methods tailored to the particular theory T. By being theory-specific and restricting their language to certain classes of formulas (such as, typically but not exclusively, ground formulas), such methods can be implemented into solvers that are more efficient in practice than general-purpose theorem provers. SMT techniques have been traditionally developed to support deductive software verification, but they have also applications in model checking, certifying compilers, automated test generation, and other formal methods.

This talk gives an overview of SMT and its applications, and highlights some long-standing challenges for a wider applications of SMT techniques within formal methods, as well as some fresh challenges introduced by new potential uses. A major challenge is providing adequate model generation features for disproving verification conditions.

[⋆] The author's research described in this talk was made possible with the partial support of grants #0237422 and #0551646 from the National Science Foundation.

Where's My Stuff? An Ontology Repair Plan *

Alan Bundy

June 15, 2007

Abstract

Appropriate representation is the key to successful reasoning. Hence, if intelligent agents are to cope with changing goals in a changing environment, they must be able to adapt their representations, i.e., to detect that a current representation is inadequate, to diagnose its shortcomings and to repair it. In this paper we address the most basic kind of representational shortcoming: inconsistency. We focus on how certain kinds of inconsistency can be repaired using a *repair plan* that we entitle *Where's My Stuff?*. We apply this repair plan manually to four examples from the domain of Physics. In each case an inconsistent ontology is repaired into a consistent one. This extends the interest of the Disproving workshop beyond the "reparation of non-theorems" to the reparation of inconsistent ontologies. The Physics domain has the advantage that many faulty ontologies have been recorded by historians of science, together with the evidence that identified their faults and the ontological repairs that were proposed to mend them. These records provide plenty of data for developing and evaluating ontology repair plans.

1 Introduction

In [Bundy *et al*, 2006, McNeill & Bundy, forthcoming] we have described the Ontology Repair System (ORS), which repairs faulty, first-order ontologies by diagnosing the execution failures of multi-agent plans. These repairs were not just belief revisions, but changes to the underlying signatures, e.g., adding or removing predicate or function arguments, splitting or conflating predicates or functions. Adding arguments and splitting functions are examples of *refinement*, in which ontologies are enriched. An inherent problem with these refinement operations is that they are only partially defined. For instance, when an additional argument is added to a function it is not always clear what value each instance should take. When a function is split into two, it is not always clear to which of the new functions each occurrence of the old one should be mapped.

In current work we are trying to develop a theory of ontology repair and to extend it to new domains. In this paper we report two advances.

- The aggregation of repair operations into *repair plans*, which helps address the partial definedness of the refinement operations.
- The development of the *Where's My Stuff*[1] repair plan and its manual application[2] to four examples from the Physics domain.

Our claim is that the *Where's My Stuff* plan can successfully account for the ontological repairs required in several historic advances in Physics. The evidence for this claim is the manually worked examples in the rest of this paper.

*The research reported in this paper was supported by EPSRC grant GR/S01771. Thanks to Graham Steel, three anonymous Disproving workshop referees and members of the *DReAM* Group for feedback on an earlier draft. Thanks to Alan Smaill for assistance with λProlog.

[1]With apologies to Amazon.

[2]An implementation in λProlog is under development.

Our immediate aim is the exploration of mechanisms for ontology repair. Physics is a convenient development domain due to the abundance of historical records of fault diagnosis and repair in Physical ontologies. Our objective is to build a prototype, possibly interactive, computer program which can emulate[3] a wide variety of these historical ontology repair episodes. In future, this work might lead to some practical application, but this is not our current objective.

By *repair plan* we mean a compound, possibly hierarchical, system of repair operations, with associated preconditions and effects, in the way that a proof plan [Bundy, 1991] is a compound system of rules of inference. The *Where's My Stuff* plan is intended to be the first move in gathering a portfolio of such repair plans, which we hope will collectively cover a large number of ontology repairs, at least in the Physics domain. For instance, we are currently developing a repair plan based on adding additional arguments to functions with unexpected variation. The triggering pattern for this new plan will overlap with that for *Where's My Stuff*, providing a rival ontology repair in some cases. Our hope is that just tens of similar repair plans will provide significant coverage of historical ontology repairs in Physics; we expect complete coverage to be impossible due to the need for idiosyncratic or domain-specific repairs in some cases.

Typed higher-order ontologies appear to be required in the Physics domain, since many of the concepts, for instance, calculus, are essentially higher-order and many of the functions only make sense when applied to objects of certain types, e.g., Orb_Vel takes an astronomical object and returns a real number. So, in this paper, *ontology* will mean a theory in typed, higher-order logic. We use the word "ontology" rather than "theory" because: (a) our emphasis on *signature* modification might be obscured by the word "theory"; (b) we intend, eventually, to apply the mechanisms we develop to ontology repair in the semantic web etc. and; (c) this frees up the word "theory" to refer to the set of *theorems* of the ontology. Higher-order logic is also required to describe the modifications to the functions of the ontology, i.e., to describe the mechanisms themselves.

An ontology O is a pair $\langle Sig(O), Ax(O)\rangle$, where $Sig(O)$ is the *signature* and $Ax(O)$ are the *axioms*. The *language* $L(O)$ of O is set of formulae generated by the grammar $Sig(O)$. The *theory* $Th(O)$ of O is the set of theorems generated from the closure of $Ax(O)$ over the rules of inference of simply-typed lambda calculus, e.g., expressed as a sequent calculus. We will write $O \vdash \phi$ when $\phi \in Th(O)$. If ontology O is faulty then the repaired ontology will be denoted $\nu(O)$, where ν is the function that converts the faulty ontologies into repaired ones, i.e., it is the instantiated repair plan. Since ontology repair could, in principle, involve changes to the underlying logic, we reserve the right to depart from these definitions (although not in this paper) and have deliberately left them a little open-ended. Note, in particular, the need for both a typed and higher-order logic, in contrast to the first-order, sorted logic of ORS. As argued above, these extensions are needed for the Physics domain.

We envisage these repair plans being implemented by higher-order deductive machinery. The preconditions of each repair plan will contain some patterns expressed as higher-order formulae. To trigger the repair plan, these patterns will be matched to the original ontology. When they match, then some higher-order output patterns will be instantiated and thereby form the repaired ontology. We have started to develop a prototype implementation in the higher-order logic programming language λProlog [Miller & Nadathur, 1988]. Since the typed, higher-order Physics ontologies required for this implementation do not already exist, we are constructing them on an 'as needed' basis for each of our test examples. Our examples below are intended to illustrate both the triggering and the repair mechanisms. The uniform presentation is intended to illustrate the essentially algorithmic nature of these processes, to convince the reader that they can be readily implemented. Human intervention is, however, currently required to prepare the ontology to facilitate the triggering process. There is a brief discussion of the deductive and search issues involved in such preparations, but the details are left to further work.

[3] I.e. "to rival with some degree of success", but not to provide a historically valid model.

2 The Where's My Stuff Ontology Repair Plan

The *Where's My Stuff* ontology repair plan is triggered by a mismatch between the predicted and the actual value of some Physics function on some object. Let us call this function *stuff*, a higher-order, variadic, function variable[4] from physical objects or systems to some values that can be added, usually the reals, but we will see, in §5 and §6, that there are other possibilities, so addition must be polymorphic. The predicted value is a deductive inference from the original ontology of Physics theory, say $O_t \vdash \mathit{stuff}(\vec{c}) = v_1$, i.e., v_1 is the value, predicted by the theoretical ontology O_t, when *stuff* is applied to the n arguments in the vector $\vec{c}$. However, the observed value is v_2, i.e., $O_s \vdash \mathit{stuff}(\vec{c}) = v_2$, where O_s is the sensory theory whose axioms record the experimental observations and whose theorems are deductions from these observations. The predicted and observed values differ, i.e., $v_1 \neq v_2$, resulting in a contradiction if the ontologies O_t and O_s are combined. So, to summarise, the triggering pattern is:

$$O_t \vdash \mathit{stuff}(\vec{c}) = v_1, \quad O_s \vdash \mathit{stuff}(\vec{c}) = v_2, \quad O_t \vdash v_1 \neq v_2 \tag{1}$$

Note that $=$ is polymorphic; it depends on the type of the value returned by *stuff*, e.g., reals. It also needs to be a bit fuzzy, since there is always some noise in experimental data. One way to achieve this would be to associate error bars with any value and count two values equal if the intersection of the intervals defined by these error bars was non-empty.

The repair is to split *stuff*, into three new terms: *stuff*, $\mathit{stuff}\sigma_{vis}$ and $\mathit{stuff}\sigma_{invis}$, where σ_{vis} and σ_{invis} are substitutions that replace one or more higher-order function variables with new functions of the same type. $\mathit{stuff}\sigma_{vis}$ and $\mathit{stuff}\sigma_{invis}$ are intended to be the visible and invisible parts of *stuff*, respectively. Then *stuff* is re-defined as the total of these two new functions:

$$\forall \vec{c}{:}\vec{\tau}.\ \mathit{stuff}(\vec{c}) \quad ::= \quad \mathit{stuff}\sigma_{vis}(\vec{c}) + \mathit{stuff}\sigma_{invis}(\vec{c}) \tag{2}$$

where the τ_i are the types of the c_i and $+$ is polymorphic, depending on the types of the values returned by *stuff*, e.g. reals. We have chosen to retain the name of the old *stuff* function as the name of the new total function. Alternatively, we could have retained it for the new visible part of *stuff* or we need not have retained it at all, choosing, say, a substitution σ_{total} to create a new term for the total function. These choices are just a matter of taste.

This new definition (2) is added to the axioms of the repaired theoretical ontology, $Ax(\nu(O_t))$. The remaining repairs to O_t and O_s depend on whether $v_1 > v_2$ or $v_1 < v_2$, where $>$ and $<$ are polymorphic total orders. If $v_1 > v_2$ then the remaining axioms of the repaired theory are copied unchanged[5] from the original axioms.

$$Ax(\nu(O_t)) \quad ::= \quad \{\phi \mid \phi \in Ax(O_t) \vee \phi = (2)\} \tag{3}$$

where $\phi = (2)$ is a convenient abuse of notation intended to abbreviate that ϕ might be the *stuff* definition axiom given in (2) above. (3) defines the axioms of the repaired ontology as being the old axioms plus (2). In particular, in the repaired ontology, $\nu(O_t) \vdash \mathit{stuff}(\vec{c}) = v_1$, i.e., the problematic prediction is preserved unchanged, but, as we shall see, it ceases to be problematic.

The sensory ontology O_s, however, *is* changed. We now take the observations of *stuff* to be observations of $\mathit{stuff}\sigma_{vis}$.

$$Ax(\nu(O_s)) \quad ::= \quad \{\phi\{\mathit{stuff}/\mathit{stuff}\sigma_{vis}\} \mid \phi \in Ax(O_s)\}$$

In particular, $O_s \vdash \mathit{stuff}\sigma_{vis}(\vec{c}) = v_2$. So, this observation no longer conflicts with the prediction. We can no longer directly observe values of *stuff*, but only of its visible part. At some future point, we hope we will devise methods to measure $\mathit{stuff}\sigma_{invis}$, but not at the current stage of repair.

[4]We use the anti-Prolog variable/constant convention: lower case letters are variables and upper case are constants.

[5]Since we have retained the old *stuff* name for the new total function.

If $v_1 > v_2$ then the roles of *stuff* and *stuff* σ_{vis} are reversed: *stuff* is renamed to *stuff* σ_{vis} in O_t but retained unchanged in O_s, i.e.,

$$\begin{aligned} Ax(\nu(O_t)) &::= \{\phi\{stuff/stuff\,\sigma_{vis}\} \mid \phi \in Ax(O_t) \vee \phi = (2)\} \\ Ax(\nu(O_s)) &::= \{\phi \mid \phi \in Ax(O_s)\} \end{aligned}$$

So, the triggering formulae (1) are transformed to one of the following:

$$\nu(O_t) \vdash stuff(\vec{c}) = v_1, \quad \nu(O_s) \vdash stuff\,\sigma_{vis}(\vec{c}) = v_2, \quad if\ O_t \vdash v_1 > v_2 \tag{4}$$
$$\nu(O_t) \vdash stuff\,\sigma_{vis}(\vec{c}) = v_1, \quad \nu(O_s) \vdash stuff(\vec{c}) = v_2, \quad if\ O_t \vdash v_1 < v_2 \tag{5}$$

each of which breaks the previous derivation of a contradiction[6]. Note that this conditional branching on whether $v_1 > v_2$ ensures that *stuff* σ_{invis} is always positive. Below, we will show examples of both (4) and (5).

Note how the *Where's My Stuff* repair plan overcomes the problem introduced in the last sentence of the first paragraph of §1: a function is split into three, but we are told exactly which occurrences of the original function turn into one of the new functions and which to leave unchanged. The repair also requires the addition of a new axiom and the mapping of some old derivations into new ones. Despite the compound structure and special properties of this repair plan, it is surprisingly widely applicable to the emulation of historical ontology repairs in Physics. In the next four sections we apply it, manually, to four such repairs, drawn from different areas of Physics and from different historical periods.

3 Application to the Latent-Heat Paradox

We start by applying it to the discovery of latent-heat by Joseph Black around 1750. [Wiser & Carey, 1983] discusses a period when heat and temperature were conflated, which presented a conceptual barrier that Black had to overcome before he could formulate the concept of latent heat. This conflation creates a paradox: as water is frozen it is predicted to lose heat, but its heat, as measured by temperature, remains constant. Black had to split the concept of heat into energy and temperature.

We can model this situation with the following formulae:

$$O_t \vdash Heat(H_2O, Start(Freeze)) = Heat(H_2O, Start(Freeze)) \tag{6}$$
$$O_s \vdash Heat(H_2O, Start(Freeze)) = Heat(H_2O, End(Freeze)) \tag{7}$$
$$O_t \vdash Heat(H_2O, Start(Freeze)) \neq Heat(H_2O, End(Freeze)) \tag{8}$$

where H_2O is the water being frozen, $Freeze$ is the time interval during which the freezing takes place, $Start$ returns the first moment of this period and End the last. (6) comes from the reflexive law of equality, (7) comes from the observed constant temperature during freezing and (8) is deduced from the then current physical theory that heat decreases strictly monotonically when objects are cooled.

These formulae match the repair plan trigger (1) with the following substitution:

$$\{Heat/stuff,\ \langle H_2O, Heat(H_2O, Start(Freeze))\rangle/\vec{c},\ Heat(H_2O, Start(Freeze))/v_1,\ Heat(H_2O, End(Freeze))/$$

To effect the repair we will define $\sigma_{vis} = \{Temp/stuff\}$ and $\sigma_{invis} = \{LHF/stuff\}$, respectively, in anticipation of their intended meanings, where LHF can be read as the latent heat of fusion. These choices instantiate (2) to:

$$\forall o{:}obj, t{:}mom.\ Heat(o,t) \quad ::= \quad Temp(o,t) + LHF(o,t)$$

which is not quite what is required, but is along the right lines. Some further indirect observations of LHF are required to witness its behaviour under different states of o so that it can be further

[6]Which is not to say that some other contradiction does not still lurk undetected.

repaired, e.g., the removal of its t argument. The $Temp$ part of the new definition needs to be further refined so that its contribution of energy depends both on temperature and mass. These further refinements will be the subject of future ontology repair plans.

In the repaired ontologies, since $Heat(H_2O, Start(Freeze)) > Heat(H_2O, End(Freeze))$, the repaired triggering formulae are transformed to :

$$\begin{aligned}\nu(O_t) &\vdash Heat(H_2O, Start(Freeze)) = Heat(H_2O, Start(Freeze)) \\ \nu(O_s) &\vdash Temp(H_2O, Start(Freeze)) = Temp(H_2O, End(Freeze))\end{aligned}$$

which breaks the derivation of the detected contradiction, as required.

4 Application to the Bouncing-Ball Paradox

Our second example is based on an experiment described in [diSessa, 1983]. In [Bundy *et al*, 2006] we described it thus:

> "...consider the experiment conducted by Andreas diSessa on first-year MIT physics students [diSessa, 1983]. The students were asked to imagine a situation in which a ball is dropped from a height onto the floor. Initially, the ball has potential but not kinetic energy. Just before it hits the floor it has kinetic but not potential energy. As it hits the floor it has neither. Where did the energy go?"

The paradox arises because students typically idealise the ball as a particle without extent. However, the energy is stored in the compression of the ball[7] and this cannot be represented unless the idealisation of the ball has extent.

We can model this situation with the following formulae:

$$\begin{aligned}O_t &\vdash TE(Ball, End(Drop)) = TE(Ball, Start(Drop)) && (9)\\ O_s &\vdash TE(Ball, End(Drop)) = 0 && (10)\\ O_t &\vdash TE(Ball, Start(Drop)) \neq 0 && (11)\end{aligned}$$

where $Ball$ is the ball, $Drop$ is the time interval from its release to contact with the ground and $TE(Ball, t)$ is the total energy of the ball at time moment t. (9) comes from the law of conservation of energy; (10) comes from the observation that the ball is stationary and at zero height at the point of contact with the ground, so has neither potential nor kinetic energy; and (11) comes from the inference that the original energy of the ball consists of potential energy which is not zero. The substitution required to instantiate the trigger (1) with these three formulae is:

$$\{TE/stuff,\ \langle Ball, End(Drop)\rangle/\vec{c},\ TE(Ball, Start(Drop))/v_1,\ 0/v_2\}$$

To effect the repair we will define $\sigma_{vis} = \{TE_{part}/stuff\}$ and $\sigma_{invis} = \{EE/stuff\}$, so the new definition of TE that is proposed is:

$$\forall o{:}obj, t{:}mom.\ TE(o,t) ::= TE_{part}(o,t) + EE(o,t)$$

where $TE_{part}(o,t)$ is the total energy of a particle, defined as sum of its potential and kinetic energy, and $EE(o,t)$ is some invisible energy to be discovered. This invisible energy will turn out to be the elastic potential energy of the ball viewed as a spring: $EE(Ball, End(Drop))$. But the need to identify a source for this invisible energy could be the incentive to re-idealise the ball as an object with a type that has such an additional source of energy available, e.g., a spring.

In the repaired ontologies, since $TE(Ball, Start(Drop)) > 0$, the repaired triggering formulae are:

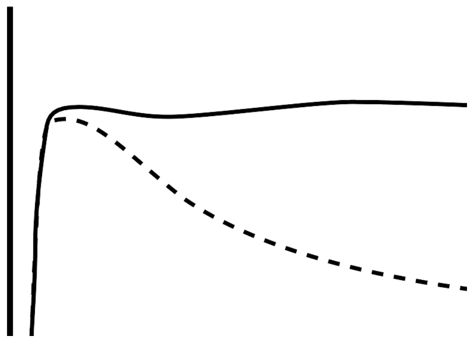
$$\begin{aligned}\nu(O_t) &\vdash TE(Ball, End(Drop)) = TE(Ball, Start(Drop)) \\ \nu(O_s) &\vdash TE_{part}(Ball, End(Drop)) = 0\end{aligned}$$

which breaks the previous derivation of a contradiction, as required.

[7]And also of the floor, but we will ignore this factor in this exercise.

5 Application to Dark Matter

Our third example is the invention[8] of dark matter. The evidence for dark matter arises comes from various sources, for instance, from an anomaly in the orbital velocities of stars in spiral galaxies[9] identified by Rubin in 1975. Given the observed distribution of mass in these galaxies, we can use Newtonian Mechanics to predict that the orbital velocity of each star should be inversely proportional to the square root of its distance from the galactic centre (called its *radius*). However, observation of these stars show their orbital velocities to be roughly constant and independent of their radius. Figure 1 illustrates the predicted and actual graphs. In order to account for this discrepancy, it is hypothesised that galaxies also contain a halo of, so called, *dark matter*, which is invisible to our radiation detectors, such as telescopes, because it does not radiate, so can only be measured indirectly.



This diagram is taken from `http://en.wikipedia.org/wiki/Galaxy_rotation_problem`. The x-axis is the radii of the stars and the y-axis is their orbital velocities. The dotted line represents the predicted graph and the solid line is the observed graph.

Figure 1: Predicted *vs* Observed Stellar Orbital Velocities

We can trigger the preconditions (1) of the *Where's My Stuff* plan with the following formulae:

$$O_t \vdash \lambda s \in Spiral.\ \langle Rad(s), Orb\_Vel(s)\rangle = Graph_A \qquad (12)$$
$$O_s \vdash \lambda s \in Spiral.\ \langle Rad(s), Orb\_Vel(s)\rangle = Graph_B \qquad (13)$$
$$O_t \vdash Graph_A \neq Graph_B \qquad (14)$$

where $Orb\_Vel(s)$ is the orbital velocity of star s, $Rad(s)$ is the radius of s from the centre of its galaxy and *Spiral* is a particular spiral galaxy, represented as the set of stars it contains. Formula (12) is the predicted orbital velocity graph based on the observed distribution of the visible stars and their masses in a spiral galaxy: the orbital velocity decreases inversely with the square root of the radius. Formula (13) is the observed orbital velocity graph: it is almost a constant function over most of the values of s. Note the use of λ abstraction to create graph objects as unary functions. These two graphs are unequal (14), within the range of legitimate experimental variation.

$Graph_A$ is deduced by Newtonian Mechanics from the observed distribution of mass in the spiral, i.e., it is a function, say, *M2OV* (mass to orbital velocity) [10] of the mass distribution

[8] discovery?

[9] `http://en.wikipedia.org/wiki/Dark_matter`

[10] Alternatively, the actual formula might be inserted here, but it is enough for our purposes to know that such a formula exists, and the actual formula would clutter and obscure the picture. It involves complex calculus requiring computer calculation to give a solution.

graph: $\lambda s \in Spiral.\ \langle Rad(s), Mass(s)\rangle$:

$$O \vdash \lambda s \in Spiral.\ \langle Rad(s), Orb\_Vel(s)\rangle = M2OV(\lambda s \in Spiral.\ \langle Rad(s), Mass(s)\rangle) = Graph_A \quad (15)$$

Of course, the actual $M2OV$ calculation was done in the reverse direction: the mass distribution of the dark matter was calculated so that the predicted orbital velocities would fit the observational evidence.

These three formulae instantiate the trigger preconditions (1) with the following substitution:

$$\{\lambda s \in g.\ \langle Rad(s), Orb\_Vel(s)\rangle / stuff,\ \langle Spiral\rangle/\vec{c},\ Graph_A/v_1,\ Graph_B/v_2\}$$

Note that the repair plan works perfectly well with higher-order objects as the values v_1 and v_2, provided polymorphic $+$ and $\neq$ can be defined as having meaning over this data-type: in this case a piecewise addition over the individual values for each star and a fuzzy, negated equality between graphs.

To effect the repair we will define $\sigma_{vis} = \{Spiral_{vis}/g\}$ and $\sigma_{invis} = \{Spiral_{invis}/g\}$, so the instantiation of definition (2) suggested by this triggering is:

$$\begin{aligned} &\lambda s \in Spiral.\ \langle Rad(s), Orb\_Vel(s)\rangle \\ &\quad ::= \lambda s \in Spiral_{vis}.\ \langle Rad(s), Orb\_Vel(s)\rangle + \lambda s \in Spiral_{invis}.\ \langle Rad(s), Orb\_Vel(s)\rangle \end{aligned}$$

where $Spiral_{vis}$ is the visible part of the galaxy, that can be detected from its radiation, and $Spiral_{invis}$ is its dark matter part. Note that $Spiral = Spiral_{vis} \cup Spiral_{invis}$, which is also an instantiation of definition (2), if you treat $\cup$ as addition for sets, but we cannot see how to trigger this simpler redefinition using the trigger (1).

In the repaired ontologies, since $Graph_A < Graph_B$, the repaired triggering formulae are:

$$\begin{aligned} \nu(O_t) &\vdash \lambda s \in Spiral_{vis}.\ \langle Rad(s), Orb\_Vel(s)\rangle = Graph_A \\ \nu(O_s) &\vdash \lambda s \in Spiral.\ \langle Rad(s), Orb\_Vel(s)\rangle = Graph_B \end{aligned}$$

which breaks the previous derivation of a contradiction, as required.

6 Application to the Precession of the Perihelion of Mercury

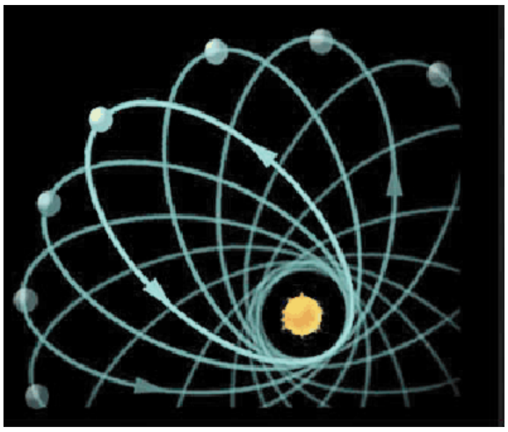
Our fourth, and last, example was suggested to us by the sociologist of science, Harry Collins: the precession of the perihelion of Mercury, i.e., the gradual rotation of the elliptical form of the orbit (see Figure 2). The orbits of the planets in the Solar System precess in this way. This is predicted by Newtonian Mechanics. However, Mercury's orbit does not precess by quite the right amount (`http://physics.ucr.edu/~wudka/Physics7/Notes_www/node98.html`). Nowadays, we understand this as an accurate prediction of Einstein's General Theory of Relativity, but for a long while it was believed to be caused by the gravitational attraction of an additional planet, named Vulcan, that was even closer to the Sun. Observation eventually ruled this out, but it is this (erroneous) prediction that we wish to model. Alternatively, we could have shown how a similar ontological repair could emulate the successful discovery of Pluto, but it is important to emphasise that our repair plan can be used to emulate ultimately *unsuccessful* ontology repairs, as well as successful ones.

We can represent the orbit of Mercury with the function $\lambda t.\ Posn(Mercury, t)$, where $Posn(o, t)$ is the 3D coordinate of object o at time t according to some implicit frame of reference. The triggering formulae are then:

$$O_t \vdash \lambda t.\ Posn(Mercury, t) = Orbit_p \quad (16)$$

$$O_s \vdash \lambda t.\ Posn(Mercury, t) = Orbit_o \quad (17)$$

$$O_t \vdash Orbit_p \neq Orbit_o \quad (18)$$



This diagram is taken from `http://physics.ucr.edu/~wudka/Physics7/Notes_www/node98.html#fig:prec.mercury`. *It shows the elliptical orbits of Mercury themselves rotating.*

Figure 2: Precession of the Perihelion of Mercury

where (16) is the predicted orbit of Mercury, (17) is the observed orbit and (18) asserts that these are not equal. Unfortunately, these triggers will not give us the right repair. What we would like is that *Solar_System* appeared in the term that instantiated *stuff* in (1). Then we could use (2) to define:

$$Solar\_System \quad ::= \quad Solar\_System_{vis} \cup Solar\_System_{invis}$$

where $Solar\_System_{invis} = \{Vulcan\}$. Unfortunately, we can't see a way of legitimately introducing *Solar_System* into the LHS of (16), say. However, all is not lost. The predicted orbit of Mercury, $Orbit_p$, is calculated by considering the mass distribution of the Solar System.

$$\begin{aligned} \lambda t.\ Posn(Mercury, t) &= M2O(\lambda s \in Solar\_System, t.\ \langle Posn(s,t), Mass(s)\rangle) \\ &= Orbit_p \end{aligned}$$

where *M2O* is the calculation of the orbit of Mercury from the distribution of mass in the Solar System over time, i.e., taking into account the gravitational influences of the sun and the other planets. As with *M2OV* in formula (15) in §5, *M2O* will be some complex function involving calculus and requiring computer calculation, but for our purposes the details do not matter; it is enough that some such function exists.

By putting this calculation into reverse[11] ($M2O^{-1}$), we can create an alternative set of trigger formulae that *do* contain *Solar_System*, as required, namely:

$$\begin{aligned} O_t &\vdash \lambda o \in Solar\_System, t.\ \langle Posn(o,t), Mass(o)\rangle = M2O^{-1}(Orbit_p) \\ O_s &\vdash \lambda o \in Solar\_System, t.\ \langle Posn(o,t), Mass(o)\rangle = M2O^{-1}(Orbit_o) \qquad (19) \\ O_t &\vdash M2O^{-1}(Orbit_p) \neq M2O^{-1}(Orbit_o) \end{aligned}$$

This set of triggers also has the nice property of predicting the mass distribution of the Solar System from the observed orbit of Mercury (this is what (19) means), and hence predicting the position of Vulcan. These three formulae instantiate the trigger preconditions (1) with the following substitution:

$$\{\lambda o \in s, t.\ \langle Posn(o,t), Mass(o)\rangle / stuff,\ \langle Solar\_System\rangle/\vec{c},\ M2O^{-1}(Orbit_p)/v_1,\ M2O^{-1}(Orbit_o)/v_2\}$$

[11] Assuming this *is* a function.

To effect the repair we will define $\sigma_{vis} = \{Solar\_System_{vis}/s\}$ and $\sigma_{invis} = \{Solar\_System_{invis}/s\}$, so the instantiation of definition (2) suggested by this triggering is:

$$\begin{aligned}&\nu(O_t) \vdash \lambda o \in Solar\_System, t.\ \langle Posn(o,t), Mass(o)\rangle \\ &\quad = \quad \lambda o \in Solar\_System_{vis}, t.\ \langle Posn(o,t), Mass(o)\rangle + \lambda o \in Solar\_System_{invis}, t.\ \langle Posn(o,t), Mass(o)\rangle\end{aligned}$$

as required. In the repaired ontologies, since $M2O^{-1}(Orbit_p) < M2O^{-1}(Orbit_o)$, the repaired triggering formulae are:

$$\begin{aligned}\nu(O_t) &\ \vdash\ \lambda o \in Solar\_System_{vis}, t.\ \langle Posn(o,t), Mass(o)\rangle = M2O^{-1}(Orbit_p) \\ \nu(O_s) &\ \vdash\ \lambda o \in Solar\_System, t.\ \langle Posn(o,t), Mass(o)\rangle = M2O^{-1}(Orbit_o)\end{aligned}$$

which breaks the previous derivation of a contradiction, as required.

This example demonstrates another dimension of choice: we may have to search back through the derivation of the contradicted prediction in order to find the most appropriate form for the trigger. We think this kind of choice was really also present in the previous examples, but we got lucky and managed to avoid making it explicit. For instance, in the Bouncing Ball example in §4, the trigger was phrased in terms of total energy, although this can hardly be directly observed. What can be observed is positions, distances and mass, from which the energies can be inferred and the contradiction derived. If this had been made more explicit then we think this dimension of search would have been revealed here too. Similar remarks can also be made about the dark matter example. Notice how this again forces us to mix theory and observation to deduce the theory-contradicting observation.

7 Conclusion

In this paper we have described the *Where's My Stuff* ontology repair plan. It consists of a higher-order, triggering pattern describing a particular kind of disagreement between theoretical predictions and experimental observations: a function *stuff* is predicted to have one value, but is observed to have another. When this pattern can be instantiated to a situation occurring in particular theoretical and sensory ontologies then this triggers some ontological repairs. The repairs have two main parts. Firstly, function refinement is applied to divide *stuff* into three functions: one for visible stuff, one for invisible stuff and one for their total. The original *stuff* is replaced by the total stuff in one ontology, but by the visible stuff in the other. Secondly, a new definition is added to the repaired theoretical ontology which defines the total function as the sum of the other two. These repairs disrupt the derivation of the contradiction.

We have applied this repair plan to four examples: latent heat, deSessa's bouncing ball , the invention of dark matter and the precession of the perihelion of Mercury. In each case we see that the repair plan provides a significant part of the required repair, but leaves some areas to be fixed. We also see that there are choices that require heuristic guidance. These choices arise in at least two ways: choices over which functions to refine into visible, invisible and their total, and choices over how far back to go in the derivations of the prediction and observation to identify the trigger formulae.

The dark matter and Mercury examples are particularly interesting, as *stuff* has to be instantiated to a compound λ term. It also shows the need for polymorphic $+$, $=$ and $>$, since these need to be interpreted differently depending on the data-type.

It's instructive to compare this repair plan with standard belief revision[12]. In belief revision the problem is to add some new belief ϕ to an existing ontology O. The interesting case is when just adding ϕ as a new axiom creates inconsistency. In this case, ϕ is usually assumed to take precedence over O, and O is adjusted by reducing its theory, up to and including removing O altogether. All this is done within a fixed signature. The *Where's My Stuff* repair plan, is triggered because an inconsistency will be created if we add, say, an observation ϕ to ontology O_t.

[12] `http://en.wikipedia.org/wiki/Belief_revision`

But ϕ does not take precedence. In fact, the signature is changed [13] so that *both* ϕ and O_t can be retained, but with a modified understanding of what each means. One is now seen to apply to a larger totality of stuff than previously assumed and the other just to the original, narrower conception of visible stuff. In this way, the inconsistency is made to melt away, but we come away with a richer conception of the world's complexity, including new questions about how to investigate the newly hypothesised invisible stuff.

Implementing a repair mechanism based on this repair plan requires higher-order matching and deduction, as well as some search control. The higher-order logic-programming language λProlog is well suited as an implementation, since it embodies all three elements. A λProlog implementation is currently under development.

We've been surprised to discover just how general the *Where's My Stuff* plan is. When we started this work we thought we knew a couple of examples, and had rejected two others: the bouncing ball and the precession of Mercury's orbit. But these rejected ones turned out to be examples too. Generality is just what we want, but, of course, it won't be enough. For instance, if it is applied too often we will get 'epicycles'. When we have 10 kinds of matter ranging from very light to very dark, then we will know it is time to apply the Occam's Razor repair plan $\smile$. We wonder what other plans are out there waiting to be revealed.

References

[Bundy, 1991] Bundy, Alan. (1991). A science of reasoning. In Lassez, J.-L. and Plotkin, G., (eds.), *Computational Logic: Essays in Honor of Alan Robinson*, pages 178–198. MIT Press. Also available from Edinburgh as DAI Research Paper 445.

[Bundy *et al*, 2006] Bundy, A., McNeill, F. and Walton, C. (2006). On repairing reasoning reversals via representational refinements. In *Proceedings of the 19th International FLAIRS Conference*, pages 3–12. AAAI Press. Invited talk.

[diSessa, 1983] diSessa, A. (1983). Phenomenology and the evolution of intuition. In Stevens, A. and Gentner, D., (eds.), *Mental Models*, pages 15–33. Erlbaum.

[McNeill & Bundy, forthcoming] McNeill, F. and Bundy, A. (forthcoming). Dynamic, automatic, first-order ontology repair by diagnosis of failed plan execution. *IJSWIS*. Special issue on ontology matching.

[Miller & Nadathur, 1988] Miller, D. and Nadathur, G. (1988). An overview of λProlog. In Bowen, R. K. & Kowalski, (ed.), *Proceedings of the Fifth International Logic Programming Conference/ Fifth Symposium on Logic Programming*. MIT Press.

[Wiser & Carey, 1983] Wiser, M. and Carey, S. (1983). When heat and temperature were one. In Stevens, A. and Gentner, D., (eds.), *Mental Models*, pages 267–297. Erlbaum.

[13] To the best of our knowledge, our use of *signature* changes to affect ontological repairs was first introduced in our earlier ORS work. In particular, signature change has not been used in belief revision.

AUTOMATIC PROOF OF GRAPH NONISOMORPHISM

ARJEH M. COHEN, JAN WILLEM KNOPPER, AND SCOTT H. MURRAY

Abstract. We describe automated methods for constructing nonisomorphism proofs for pairs of graphs. The proofs can be human-readable or machine-readable. We have developed a proof generator for graph nonisomorphism, which allows users to input graphs and construct a proof of (non)isomorphism.

1. Introduction

With the growth in computer power and internet access, an increasing number of problems are solved on remote machines by programs written by experts in a particular field. In this situation, the user may have no knowledge of the algorithm used, its implementation, or indeed how the remote machine is maintained. A mere yes-or-no answer cannot be trusted: we need additional verification that the answer is correct. For mathematical problems, the most obvious form of verification is a proof of correctness. In this article, we construct such proofs for the problem of graph isomorphism.

If two graphs are isomorphic, and we are given an isomorphism, then it is easy to prove this by checking the isomorphism. Proving that a pair of graphs are *not* isomorphic is more difficult. We show how to generate such a proof automatically. Our proofs are intended to be human-readable but could be modified to give machine-readable proofs as in [4]. We use a lot of computer time to find a short and understandable proof. Hence it can take much longer to generate a proof than to determine nonisomorphism. Although we are primarily interested in practical computations, we occasionally use the concept of polynomial-time algorithms [6, Chapter 36].

In Section 2, we look at invariants: functions that take the same value on isomorphic graphs, but may take different values on nonisomorphic graphs. In many cases, invariants give short and easy-to-verify proofs of nonisomorphism. For example, two graphs with different numbers of vertices clearly cannot be isomorphic, so this is an easily-checked invariant.

When no simple invariants can be found to distinguish two graphs, we resort to general graph-isomorphism algorithms building on the methods of [5]. We have implemented the algorithm of Luks [11], and modified it to output a human-readable proof. We have also modified the nauty implementation [12] of McKay's algorithm [14] to produce such a proof. We only discuss McKay's algorithm, since it gave a shorter proof than Luks' in every case we tried. The modified version of McKay's algorithm can also prove the correctness of the identification of the automorphism group of a graph.

We have developed a proof generator for graph nonisomorphism [16], described in Section 4. This will automatically construct a proof of (non)isomorphism, and can also be used to compose a proof interactively by choosing invariants or calling one of the modified algorithms. The algorithms are implemented in GAP [7], apart from the modifications to nauty, which is in C [9]. The user interface is written in Java [17]. The proof generator, with installation instructions, can be found online at [16] or in the RIACA software repository.

Date: June 22, 2007.

Because of the exponential growth in the lengths of the proofs produced, our modified version of McKay's algorithm is only practical for relatively small graphs. Invariants can frequently distinguish much larger graphs, however.

The proofs have a hierarchical structure, with many small lemmas (see the example in Section 4). It is possible to hide the proof of certain lemmas to take into account the different levels of mathematical expertise among users. The user can also click on a hidden part of the proof to reveal it.

Although we do not focus on complexity in this paper, it may be worth mentioning that graph nonisomorphism is neither known nor believed to be in NP, that graph isomorphism has time complexity $O(\exp(n^{1/2+o(1)}))$ (cf. [2, 11]), and that graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses (cf. [10]), where n refers to the number of vertices of G. The proofs we give are not based on these advanced algorithms.

2. Invariants

In order to check whether two graphs are isomorphic, the following invariants are checked in order:

(1) number of vertices
(2) number of edges
(3) degree multiset
(4) diameter
(5) girth
(6) distance multiplicity
(7) subgraph invariance
(8) extended subgraph invariance
(9) characteristic polynomial of the adjacency matrix and Seidel matrix
(10) Smith normal form of the adjacency matrix
(11) powers of the adjacency matrix
(12) number of triangles per vertex, edge (multiset)
(13) number of $K_{2,1,1}$-graphs per edge (multiset)
(14) edge distance multiplicity
(15) multiset of all edge invariants per edge

The precise definitions of most of these invariants can be found in [3].

The order of the invariants is chosen to balance understandability with ease of calculation. In larger graphs some of the invariants high in the tree become harder to humanly verify, but still can give information about the graph.

Note that some invariants are straightforward to calculate but harder to prove correct. Some effort is made to reduce the output, for example if the number of vertices with a certain degree differs in two graphs it is not needed to mention the number of vertices with a different degree.

3. McKay's algorithm

3.1. **Introduction.** The current implementation of McKay's algorithm [13, 14], called nauty [12], is one of the most efficient practical graph isomorphism solvers available. We have modified this program to give additional output, which allows us to construct a human-readable proof.

Nauty's default routine for establishing nonisomorphism involves computing a *canonical labelling* for each graph. That is, a labelling of the vertices by integers with the property that two graphs are isomorphic iff this labelling induces an isomorphism. The problem with this for constructing a formal proof is that the definition of the canonical labelling is almost as involved as the algorithm itself.

We chose instead to prove nonisomorphism by constructing automorphism groups. A disadvantage of using the automorphism group is that a new graph must be constructed from the two earlier graphs and that the resulting graph is twice as big as the original graphs. Let $G = (V, E, \gamma)$ and $G' = (V', E', \gamma')$ be two connected graphs and suppose $v \in V$ and $v' \in V'$. We create a new graph G'' by relabeling V' so that V and V' are disjoint, adding an edge $\{v, v'\}$ and creating a new coloring function γ'' that colors the vertices in V like γ and the vertices in V' like γ' except for v and v' which are given a new color c that is different from all other colors. In other words, $G'' = (V'', E'', \gamma'')$ where $V'' = V \cup V'$, $E'' = E \cup E' \cup \{\{v, v'\}\}$, $\gamma''(u) = \gamma(u)$ for $u \in V \setminus \{v\}$, $\gamma''(u') = \gamma'(u')$ for $u' \in V' \setminus \{v'\}$, and $\gamma''(v) = \gamma''(v') = c$.

We can now determine whether there is an isomorphism $G \to G'$ that takes v to v', by running the automorphism algorithm to compute the group of automorphisms of G'' (they leave the edge $\{v, v'\}$ fixed). If the resulting group of automorphisms contains an element that exchanges v and v' then that element gives an isomorphism between G and G'. Now fix a vertex $v \in V$.

Suppose there exists an isomorphism between G and G'. Let σ be such an isomorphism. Let $v' \in V'$ be the image of v under σ. Now construct G'' with this v'. If the automorphism algorithm is called with G'', then σ can be retrieved from the automorphism group.

Suppose we want to prove that there exist no isomorphisms that transform G to G'. If an isomorphism σ exists, then for some $v' \in V$ the computed group of automorphisms of G'' must contain an element that transfers v to v'. If we can prove that for all $v' \in V'$ the automorphism group of the corresponding G'' contains no such element then this is a proof that the graphs are not isomorphic.

If we know automorphisms of G' then we can use these to reduce the number of checks. Suppose now that τ is an automorphism of G' that does not fix $v' \in V'$, so there is a vertex $u' = \tau(v')$ distinct from v'. Construct G'' with v' and calculate the group of automorphisms A. Now the group of automorphisms for G'' constructed with u' is $B = \{\tau\sigma\tau^{-1} \mid \sigma \in A\}$. It is easy to see that the number of automorphisms that transform v to v' in A is equal to the number of automorphisms that transform v to u' in B. This means that for a nonisomorphism proof it is sufficient to prove the nonexistence only for one vertex v' in each orbit under a group of known automorphisms of G'. A further reduction is discussed in the first paragraph of Section 4.

3.2. **Algorithms and variables.** Let $G = (V, E)$ be a finite graph. A *partition* is defined as a vertex coloring $\pi : V \to C$ with an ordering on $\pi(V) = C$. For example, by $\pi = [1 \mid 2\ 4 \mid 3]$, we mean $\pi(1) < \pi(2) = \pi(4) < \pi(3)$. A set consisting of vertices with the same color is called a *cell*. A partition is called *discrete* if all vertices have a different color, for example $[1 \mid 2 \mid 3 \mid 4]$ is discrete. Let π and π' be partitions of a set of vertices V. Then π is called *finer* than π' if every cell of π is a subset of a cell of π' and $\pi'(v) > \pi'(v') \Rightarrow \pi(v) > \pi(v')$; π' is then called *coarser* than π. Note that π is both finer and coarser than itself. If π is finer (or coarser) than π' and $\pi \neq \pi'$ then π is called *strictly finer* (or *strictly coarser*) than π'. The number of cells of π is denoted by $|\pi|$. Let $v \in V$ and $W \subseteq V$. Define $\mathrm{adj}_W(v)$ to be the number of elements of W which are adjacent to v in G. If $\pi = [V_1 \mid \ldots \mid V_k]$ is a partition of V and $v \in V_i$, for some i, then we define $\pi \circ v$ to be $(V_1, \ldots, V_{i-1}, \{v\}, V_i \setminus \{v\}, V_{i+1}, \ldots, V_k)$ if $|V_i| > 1$, and π otherwise.

Refinement function. Let $G = (V, E)$ be a graph and $\pi = (V_1, \ldots, V_k)$ a partition of V. For a sequence $\alpha = (V_{i_1}, \ldots, V_{i_l})$ of distinct cells of π, let $\mathcal{R}(G, \pi, \alpha)$ be a partition of V, with the following properties:

(1) $\mathcal{R}(G, \pi, \alpha)$ is finer than π
(2) $\mathcal{R}(G^\sigma, \pi^\sigma, \alpha^\sigma) = \mathcal{R}(G, \pi, \alpha)^\sigma$, for all $\sigma \in \mathrm{Sym}(V)$.

A function defined this way is called a *refinement function*. Now we will give an example of a refinement function (cf. Algorithm 1 from [13] and Algorithm 2.5 in [14]). This is the standard algorithm that is used in nauty. For some types of graphs other refinement functions might give better results. Define $\pi \perp v$ to be the refinement $\mathcal{R}(G, \pi \circ v, \{v\})$.

The idea behind the algorithm is looking at the number of edges between cells of a partition. Let V_i be cells of a partition π. Let V_j be another cell of π. Now calculate the value of $\mathrm{adj}_{V_i}(\cdot)$ for the points in V_j. If the value is not the same for all points, then it is possible to make a finer partition, in which V_j is split according to the different values.

This function can be used to narrow down the number of possibilities. The number of cells can be increased in a way that is invariant under automorphisms. When using a reference discrete partition it is also possible to check if the $\mathrm{adj}_v(\cdot)$ values are the same. If they are not, then no map from a partition finer than the current partition and the reference partition can be an automorphism. This has been implemented in the proof constructor.

The basic search tree. Let $G = (V, E, \gamma)$ be a colored graph. A discrete partition gives a labeling of G. With two discrete partitions of the same vertices it is possible to construct the vertex map that takes a vertex to the vertex in the second partition with the same index (recall that a discrete partition is actually a list). It is then possible to check whether this map is an automorphism. Now let p be a discrete partition finer than $\pi_0 = \gamma(V)$. If we check for each discrete partition p' finer than π_0, whether the vertex map between p and p' is an automorphism then we have found all automorphisms in the automorphism group of G.

Checking all discrete partitions is not efficient. Fortunately it is possible to reduce the number of checks by refinement and further it is sufficient to not generate the full automorphism group but only generate its generators. This means that known automorphisms can be used to reduce the number of possibilities. We describe the basic search tree and the methods to reduce the number of checks. Algorithm 2 exhibits the code involved.

We now define the *search tree* $T(G, \pi)$ on the nodes labeled by partitions of V. The root is π. A node in the tree with a discrete partition is a leaf. Let the partition π be a node in the tree that is not discrete. Then the children of π are the partitions $\pi \perp w$ for each w in the first cell of π with maximal length.

By comparing all leaves with the first leaf, the complete automorphism group can be obtained.

3.3. **Implementation and our modifications.** A pair of discrete partitions of the same graph gives a map from V to V. If such a map keeps the edges invariant it is an automorphism. Note that it is possible to generate the automorphism group by fixing one discrete partition and letting the other run through the possibilities.

These possibilities can be narrowed down by using the refinement function $\mathcal{R}$. Let π and π' be partitions of the same graph. Suppose there exists an automorphism σ such that for every vertex v we have $\pi(v) = \pi'(\sigma(v))$, then because of the nature of the refinement function $\mathcal{R}(G, \pi, v) = \mathcal{R}(G, \pi', \sigma(v)$. In general it is not necessary to carry out the full refinement procedure. It suffices to show that the step in which the partitions are made finer runs parallel in the sense that the refinements of π and π' are compatible; if they are not, there cannot be an automorphism and we are finished.

For the children of the node, it is enough to look at vertices in different orbits. Suppose π is a partition in the tree and u_1 and u_2 are vertices to split and a is an automorphism such that $a(u_1) = u_2$, then for every vertex $v : a((\pi \perp u_1)(v)) = (\pi \perp u_2)(v)$. This means that the node $\pi \perp u_2$ has only leaves as descendants that are either not isomorphic or isomorphic with an automorphism already calculated from the descendants of $\pi \perp u_1$.

Each leaf, or discrete partition in the search tree, is compared with the fixed partition. If the resulting map is an automorphism it is added to the generators of the automorphism group.

McKay has written an implementation of his algorithm called nauty [12]. This implementation is in C [9]. Included in the implementation is an interactive program called

Algorithm 1 Refinement algorithm

```
Input: $G$ is a graph (used to calculate $d$), $\pi$ is the partition that needs to be refined and
    $\alpha = (W_1 \dots W_M)$ is a list of cells, with which the partition will be refined.
Returns: $\tilde{\pi}$, a partition finer than $\pi$        ▷ more can be said, but this is not needed to
    generate a proof
 1: function $\mathcal{R}(G, \pi, \alpha)$
 2:     var
 3:         $\tilde{\pi}$:partition                          ▷ a partition finer than $\pi$
 4:         $\pi'$:partition                                 ▷ a partition finer than $\pi$
 5:         $\tilde{\alpha}$:partition                       ▷ a partition finer than $\alpha$
 6:         $m$:integer                                      ▷ index of $\tilde{\alpha}$
 7:         $t$:integer                                      ▷ position in $\pi'$
 8:     end var
 9:     $\tilde{\pi} := \pi$                                 ▷ $\tilde{\pi}$ is finer than $\pi$
10:     $\tilde{\alpha} := \alpha$
11:     $m := 1$                    ▷ $M = \tilde{\alpha}$ only grows if $\tilde{\pi}$ becomes strictly finer.
12:     while $m \leq |\tilde{\alpha}|$ and $\tilde{\pi}$ is not discrete do
13:         $k := 1$            ▷ Let $|\tilde{\pi}| = K$. Then $K - k$ decreases and is nonnegative.
14:         while $k \leq |\tilde{\pi}|$ do
15:             calculate the partition $\pi' = (X_1, \dots, X_s)$ of $\tilde{\pi}[k]$ ordered by $adj_{\tilde{\alpha}[m]}$.
16:             let $t$ be the index of the first set in $\pi'$ with maximal size

17:             if $\tilde{\pi}[k] = \tilde{\alpha}[j]$, for any $j$ then
18:                 replace $\tilde{\alpha}[j]$ by $\pi'[t]$
19:             end if

20:             for $i := 1$ to $t - 1$ do
21:                 append $\pi'[i]$ to $\tilde{\alpha}$
22:             end for
23:             for $i := t + 1$ to $|\pi'[i]|$ do
24:                 append $\pi'[i]$ to $\tilde{\alpha}$
25:             end for

26:             update $\tilde{\pi}$ by splitting the cell $\tilde{\pi}[k]$ into the cells $X_1, \dots, X_s$ in that order.
                                                             ▷ $\tilde{\pi}$ becomes finer.
27:             $k := k + 1$
28:         end while
29:         $m := m + 1$
30:     end while
31:     return $\tilde{\pi}$
32: end function
```

dreadnaut. It has options to give more information. We have extended these options so that with new options turned on dreadnaut will produce output needed to construct a proof. In particular, we display which node of the search tree we are currently working on. We also display the partition computed in line 15 of Algorithm 1, whenever this partition is strictly finer than the existing one.

This modified dreadnaut program is called from GAP. The data from the calculation in dreadnaut is sent to standard output in XML form and parsed using the XML parser in the GAPDoc package. The resulting tree is then traversed recursively and transformed into a human readable proof. At the moment a lot of information is sent from dreadnaut to GAP in this way. It should be possible to reduce this to improve performance. Some of

Algorithm 2 Finding generating automorphisms

Input: G is a graph (used to check automorphism), p is the reference discrete partition, π is a partition finer than π_0, $\tilde{\pi}$ is the set used to refine

Returns: *result* is (the set of generators of) the group of automorphisms that of G that fix π.

```
 1: function FindAutomorphisms($G, p, \pi, \tilde{\pi}$)
 2:     var
 3:         $c$:cell                          ▷ $c$ is the first cell of $\pi$ of maximal length
 4:         $v$:vertex                                                      ▷ $v \in c$
 5:         $\pi'$:partition                                      ▷ $\pi'$ is finer than $\pi$
 6:     end var
 7:     $result := \emptyset$
 8:     $\pi' := \mathcal{R}(G, \pi, \tilde{\pi})$
 9:     if $\pi'$ is discrete then                           ▷ $p$ and $\pi'$ define a map
10:         if the map from $(p, \pi')$ denotes an automorphism then
11:             $result := \{$that automorphism$\}$
12:         else
13:             $result := \emptyset$
14:         end if
15:     else                 ▷ recursion; this terminates since the number of cells in
16:         $c$ :=the first cell of $\pi'$ of maximal length
17:         for $v \in c$ do
18:             if an automorphism $\sigma$ of $G$ *is known* such that $\sigma(\pi') = \pi'$ and such that
        there exists a marked $v' \in c$ with $\sigma(v') = v$ then
19:                 do nothing       ▷ no new generator will be found, $v$ does not have to be
        marked
20:             else
21:                 $result := result \cup$ FindAutomorphisms($G, p, \pi' \circ v, (v)$)
22:                 mark $v$
23:             end if
24:         end for
25:     end if
26:     return $result$
27: end function
```

the calculations in the refinement function turn out not to be necessary in the final proof; these calculations are omitted.

3.4. **Example.** We want to check whether the two graphs are isomorphic, but the part of nauty that we use gives the automorphism group of a colored graph. It is then possible to check whether a vertex can be mapped to a vertex of the other graph by creating a new graph by adding an edge between two vertices of the same color of different graphs, coloring these two vertices in a new color and running the algorithm on that graph and that edge. It is clear that if for all pairs of vertices there are no automorphisms that exchange the graphs, there is no graph isomorphism. It is sufficient to fix a vertex in one of the graphs and to only use one vertex in an orbit of the other graph.

Now look at the graphs in Figure 1. We use the upper left vertex of the right graph and look at the orbits of the left graph. All vertices are in the same orbit (under rotation). So it is sufficient to do the construction on the upper right vertex: see Figure 2.

The search tree in Figure 3 is formed by refining and case distinction. The root of the search tree is the starting partition [15|234678]. From looking ahead at the algorithm output, we get the reference partition p = [1|5|3|7|2|4|6|8]. The starting partition can be

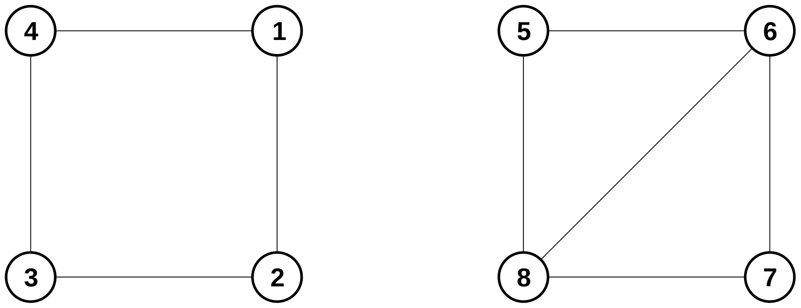


FIGURE 1. Two graphs

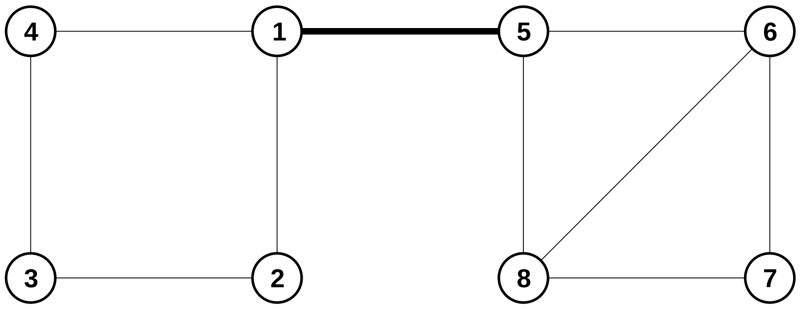


FIGURE 2. The two graphs connected by an edge

refined to [1|5|3|7|24|68] in a number of steps. Since there has not been case distinction yet, all discrete partitions p' finer than the starting partition can be refined in the same way and we will not prove this for each refining step.

If we look at how the cell 234678 is connected to 15 we see that 3 and 7 are the only two vertices that have no connection to 15 and we can therefore split the partition to [15|37|2468]. Now we look at how 2468 is connected to itself. The vertices 6 and 8 are connected to another vertex in 2468 but 2 and 4 are not. The partition can now be split further to [15|37|24|68]. Now we look at how 15 is connected to 24. 1 is connected to 24, but 5 is not. The partition can therefore be split to [15|37|24|68]. Finally we look at how 37 is connected to 24. 3 is connected to two vertices of 24 and 7 is connected to none. So we end get the partition [1|5|3|7|24|68].

Since this partition cannot be split further by refinement (it is not necessary to prove this, we would just be doing more work), the tree is split by case distinction of 24: we can color 24 so that $\gamma(2) < \gamma(4)$ or so that $\gamma(4) < \gamma(2)$ (where γ is the coloring).

In the left branch we have a case distinction again for the cell 68 and we get our first two end-nodes. The graphs represented by these end-nodes are isomorphic with isomorphism $(6, 8)$. The first leaf we get is [1|5|3|2|4|6|8] which is our reference partition p. If $p' = p$ we get the identity. The second leaf is p' = [1|5|3|7|2|4|8|6], which leads to the automorphism $(6, 8)$.

Now we return to the case $\gamma(4) < \gamma(2)$. Since we know that 6 and 8 are in the same orbit under permutations that stabilize 2 and 4 we can to assume $\gamma(6) < \gamma(8)$. This gives us another end-node p' = [1|5|3|7|4|2|6|8], which gives another isomorphism with the first end-node: $(2, 4)$.

The automorphism group now becomes $\langle(2, 4), (6, 8)\rangle$. There are no automorphisms that interchange 1 and 5, and therefore the graphs are not isomorphic.

4. A PROOF CONSTRUCTOR

We have developed a software package that automatically constructs a proof of (non)isomorphism of two given graphs. It is possible to ask for a specific proof by choosing

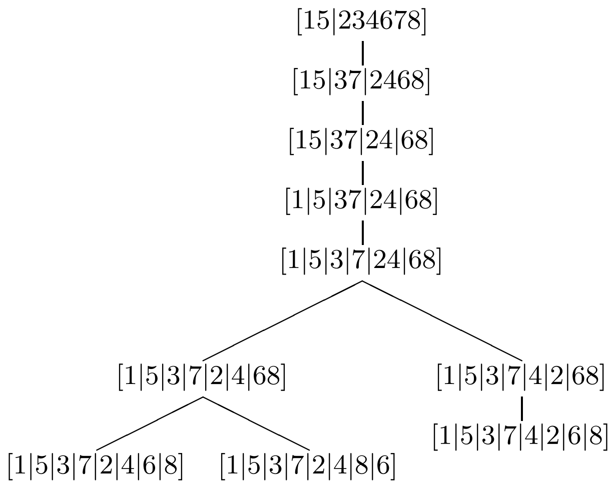


FIGURE 3. The search tree in McKay's algorithm

invariants, calling Luks' algorithm, or calling McKay's algorithm. It is conceivable that the package be made more interactive by, for example, using a vertex invariant as the coloring in the first step of MacKay's algorithm. For instance, in the example below, the case $v = 2$ could be ruled out by the observation that the valence of vertex 2 in H does not match the valence of vertex 1 in G.

The software can derive the automorphism group of a single graph by calling the algorithm from Section 3. It can further derive a proof of graph nonisomorphism by using the graph automorphism algorithm from Section 3 in the following way.

The proof constructor and most of the algorithms assume that the graphs are connected. For example Luks's algorithm fails if the graphs are not connected. However it is relatively easy to reduce graph (non-) isomorphism of unconnected graphs to graph (non-) isomorphism of the connected components.

Because of the recursive nature of our proof, it is possible to modify the output for the level of mathematical sophistication of the user by removing low-level lemmas. The following example involving McKay's algorithm on the graphs above, has been modified in this way.

```
Proposition: the graph G with vertices [ 1, 2, 3, 4 ] and edges [ [ 1, 2 ], [ 1, 4 ], [ 2, 3 ], [ 3, 4 ] ] and
the graph H with vertices [ 1, 2, 3, 4 ] and edges [ [ 1, 2 ], [ 1, 4 ], [ 2, 3 ], [ 2, 4 ], [ 3, 4 ] ] are not
isomorphic.
Proof:
  Suppose that p is an isomorphism that transforms G to H. Let v = 1^p. For all vertices v of H we show that
  there are no isomorphisms transforming 1 to v.
  To prove this we use information about the orbits of H under automorphisms on H. If a is an automorphism
  and v^a=v', then 1^p = v' if and only if 1^p^(a^-1) = v. In other words it is enough to verify for all v in
  different orbits.
  Let A be the group generated by (2,4) and (1,3). It is straightforward to verify that A is a group of
  automorphisms of H. Then we calculate the orbits.
  Proposition: The orbits of A are [ 1, 3 ] and [ 2, 4 ]. (** proof hidden **)
  It suffices to consider one vertex for each orbit i.e. the cases for v = 1 and v = 2.
    case v = 1
      From G and H we now construct a new graph F by relabelling G with (), relabelling H with
      (1,5)(2,6)(3,7)(4,8) and by joining the images of 1 of G and 1 of H with a new edge.
      The resulting graph F has vertices [ 1 .. 8 ], edges [ [ 1, 2 ], [ 1, 4 ], [ 1, 5 ], [ 2, 3 ], [ 3, 4 ], [
      5, 6 ], [ 5, 8 ], [ 6, 7 ], [ 6, 8 ], [ 7, 8 ] ] and new coloring [ 1 5 | 2:4 6:8 ].
      We now calculate the automorphism group of F and check whether there exists an automorphism that
      transforms 1 to 5.
      The automorphism group of the coloured graph G with vertices [ 1 .. 8 ] and edges [ [ 1, 2 ], [ 1, 4 ], [
      1, 5 ], [ 2, 3 ], [ 3, 4 ], [ 5, 6 ], [ 5, 8 ], [ 6, 7 ], [ 6, 8 ], [ 7, 8 ] ] and colored by the
      partition [ 1 5 | 2:4 6:8 ] is generated by the permutations [ (6,8), (2,4) ].
      Proof:
        Lemma: The permutations [ (6,8), (2,4) ] are automorphisms. (This is straightforward to verify.)
        Any automorphism can be written in the form p^-1 p', with p a fixed permutation and p' a variable
        permutation.
        Let p be [ 1 | 5 | 3 | 7 | 2 | 4 | 6 | 8 ] i.e. (2,5)(4,7,6) in cycle notation.
        If [ 1 2 | 3:8 ]^p' = [ 1 5 | 2:4 6:8 ] then [ 1 | 2 | 3 | 4 | 5 6 | 7 8 ]^p' =[ 1 | 5 | 3 | 7 | 2 4 | 6
        8 ].
        Proof:
          Lemma (refine part)
            If [ 1 2 | 3:8 ]^p' = [ 1 5 | 2:4 6:8 ] then [ 1 2 | 3 4 | 5:8 ]^p' = [ 1 5 | 3 7 | 2 4 6 8 ].
          Proof:
            Look at [ 1 2 ]^pi and how it is connected to [3:8]^pi, for pi=p,p'.
            First for p
              [ 1 2 ]^p = [ 1 5 ].
              [ 3:8 ]^p = [ 2:4 6:8 ].
              The vertices 3 7 are not connected to any vertices of [ 1 5 ].
              The vertices 2 4 6 8 are each connected to 1 vertex of [ 1 5 ].
            QED(p)
            Then for p'
              [ 1 2 ]^p' = [ 1 5 ].
              [ 3:8 ]^p' = [ 2:4 6:8 ].
              The vertices 3 7 are not connected to any vertices of [ 1 5 ].
              The vertices 2 4 6 8 are each connected to 1 vertex of [ 1 5 ].
            QED(p')
            If p^-1p' is an automorphism then it transfers [ 1 5 ] to [ 1 5 ] and [ 2:4 6:8 ] to [ 2:4 6:8 ] and
            must therefore transfer [ 3 7 ] to [ 3 7 ] and [ 2 4 6 8 ] to [ 2 4 6 8 ].

            Since [ 1 2 | 3 4 | 5:8 ]^p = [ 1 5 | 3 7 | 2 4 6 8 ], we now know that [ 1 2 | 3 4 | 5:8 ]^p' = [ 1
            5 | 3 7 | 2 4 6 8 ].
          QED(refine part)
          Lemma (refine part)
            If [ 1 2 | 3 4 | 5:8 ]^p' = [ 1 5 | 3 7 | 2 4 6 8 ] then [ 1 2 | 3 4 | 5 6 | 7 8 ]^p' = [ 1 5 | 3 7
            | 2 4 | 6 8 ].
          (** proof hidden **)
          Lemma (refine part)
            If [ 1 2 | 3 4 | 5 6 | 7 8 ]^p' = [ 1 5 | 3 7 | 2 4 | 6 8 ] then [ 1 | 2 | 3 4 | 5 6 | 7 8 ]^p' = [ 1
            | 5 | 3 7 | 2 4 | 6 8 ].
          (** proof hidden **)
          Lemma (refine part)
            If [ 1 | 2 | 3 4 | 5 6 | 7 8 ]^p' = [ 1 | 5 | 3 7 | 2 4 | 6 8 ] then [ 1 | 2 | 3 | 4 | 5 6 | 7 8 ]^p'
            = [ 1 | 5 | 3 | 7 | 2 4 | 6 8 ].
          (** proof hidden **)
        QED(refinement)
        Now we look at all the different possibilities for [ 1 | 2 | 3 | 4 | 5 6 | 7 8 ]^p' = [ 1 | 5 | 3 | 7 |
        2 4 | 6 8 ] by looking at different possibilities for 5^p'.
          Suppose that 5^p' = 2.
            Now [ 1 | 2 | 3 | 4 | 5 | 6 | 7 8 ]^p' = [ 1 | 5 | 3 | 7 | 2 | 4 | 6 8 ].
            Now we look at all the different possibilities for [ 1 | 2 | 3 | 4 | 5 | 6 | 7 8 ]^p' = [ 1 | 5 | 3
            | 7 | 2 | 4 | 6 8 ] by looking at different possibilities for 7^p'.
              Suppose that 7^p' = 6.
                Now [ 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 ]^p' = [ 1 | 5 | 3 | 7 | 2 | 4 | 6 | 8 ].
                So p' = [ 1 | 5 | 3 | 7 | 2 | 4 | 6 | 8 ] or (2,5)(4,7,6).
                Then p^-1p' = () is an automorphism.
                Further more it is included in H (it is the identity).
              QED(case 7^p' = 6)
              Suppose that 7^p' = 8.
                Now [ 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 ]^p' = [ 1 | 5 | 3 | 7 | 2 | 4 | 8 | 6 ].
                So p' = [ 1 | 5 | 3 | 7 | 2 | 4 | 8 | 6 ] or (2,5)(4,7,8,6).
                Then p^-1p' = (6,8) is an automorphism.
                Further more it is included in H (it is a generator of H).
              QED(case 7^p' = 8)
            QED(case distinction 7^p')
          QED(case 5^p' = 2)
          Suppose that 5^p' = 4.
            Now [ 1 | 2 | 3 | 4 | 5 | 6 | 7 8 ]^p' = [ 1 | 5 | 3 | 7 | 4 | 2 | 6 8 ].
            Now we look at all the different possibilities for [ 1 | 2 | 3 | 4 | 5 | 6 | 7 8 ]^p' = [ 1 | 5 | 3
            | 7 | 4 | 2 | 6 8 ] by looking at different possibilities for 7^p'.
              Suppose that 7^p' = 6.
                Now [ 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 ]^p' = [ 1 | 5 | 3 | 7 | 4 | 2 | 6 | 8 ].
                So p' = [ 1 | 5 | 3 | 7 | 4 | 2 | 6 | 8 ] or (2,5,4,7,6).
              QED(case 7^p' = 6)
            QED(case distinction 7^p')
          QED(case 5^p' = 4)
        QED(case distinction 5^p')
      QED(automorphismgroup)
    QED(case v = 1)
    case v = 2
    (** the proof is similar to the case v = 1 and hence omitted. **)
  QED(case distinction)
QED(graphisomorphism)
```

The graphical frontend of our proof constructor is written in Java. Most of the algorithms are written in GAP. From Java it is possible to call these through the RIACA GAP Service by the corresponding RIACA GAP Link [8]. From GAP a modified local copy of dreadnaut is called on demand. The information to dreadnaut is send in the format used by dreadnaut, the information sent back to GAP is sent in a simple XML format. For the link with GAP we use the OpenMath library [15] and GAP phrasebook from RIACA. They depend on the parsing library ANTLR [1].

References

[1] *ANTLR Reference Manual*, 2005. Available from World Wide Web: `http://www.antlr.org/doc/index.html`.

[2] László Babai. Moderately exponential bound for graph isomorphism. In *Fundamentals of computation theory (Szeged, 1981)*, volume 117 of *Lecture Notes in Comput. Sci.*, pages 34–50. Springer, Berlin, 1981.

[3] A. E. Brouwer, A. M. Cohen, and A. Neumaier. *Distance-regular graphs*, volume 18 of *Ergebnisse der Mathematik und ihrer Grenzgebiete (3) [Results in Mathematics and Related Areas (3)]*. Springer-Verlag, Berlin, 1989.

[4] Arjeh Cohen and Scott Murray. Certifying solutions to permutation group problems. Lecture notes for the Calculemus Autumn School, Pisa, 23 Sep-4 Oct 2002. Available from World Wide Web: `http://www.win.tue.nl/~amc/pub/permgp.pdf`.

[5] Arjeh Cohen, Scott Murray, Martin Pollet, and Volker Sorge. Certifying solutions to permutation group problems. In Franz Baader, editor, *19th International Conference on Automated Deduction*, pages 258–273. Springer-Verlag, Berlin, 2003.

[6] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to algorithms.* The MIT Electrical Engineering and Computer Science Series. MIT Press, Cambridge, MA, 1990.

[7] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.4*, 2004. Available from World Wide Web: `http://www.gap-system.org`.

[8] RIACA GAP phrasebook, 2004. Available from World Wide Web: `http://www.mathdox.org/phrasebook/gap/`.

[9] B. W. Kernighan and D. M. Ritchie. *The C Programming Language, Second Edition.* Prentice-Hall, Englewood Cliffs, New Jersey, 1988.

[10] Adam R. Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM J. Comput.*, 31(5):1501–1526 (electronic), 2002.

[11] Eugene M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *J. Comput. System Sci.*, 25(1):42–65, 1982.

[12] Brendan D. McKay. *nauty User's Guide.* Computer Science Department Australian National University, ACT 0200, Australia. Available from World Wide Web: `http://cs.anu.edu.au/~bdm/nauty/nug.pdf`. version 2.2.

[13] Brendan D. McKay. Computing automorphisms and canonical labellings of graphs. In *Combinatorial mathematics (Proc. Internat. Conf. Combinatorial Theory, Australian Nat. Univ., Canberra, 1977)*, volume 686 of *Lecture Notes in Math.*, pages 223–232. Springer, Berlin, 1978.

[14] Brendan D. McKay. Practical graph isomorphism. In *Proceedings of the Tenth Manitoba Conference on Numerical Mathematics and Computing, Vol. I (Winnipeg, Man., 1980)*, volume 30, pages 45–87, 1981.

[15] RIACA OpenMath library, 2004. Available from World Wide Web: `http://www.mathdox.org/omlib/`.

[16] Proof assistant for graph non-isomorphism, 2006. Available from World Wide Web: `http://www.mathdox.org/graphiso/`.

[17] Sun. *Java 2 Platform, Standard Edition, v 1.4.2 API Specification*, 2004. Available from World Wide Web: `http://java.sun.com/j2se/1.4.2/docs/api/`.

A Paradigm Shift in ATP:
Towards Model-based Reasoning Systems

(invited talk abstract)

Koen Claessen (koen@chalmers.se)

Introduction Traditionally and historically, ATP systems have been used in a rather specialized way. The problems they solved were relatively small, and almost always constructed by hand, very often by an expert on ATP systems (if not the actual creator of the ATP system at hand). In this way, if a problem could not be proved by a system, the behavior of the system could somehow be understood by the constructor of the problem, and the problem was tweaked, for example by changing an axiom, or adding new assumptions. When a problem was difficult, this usually was caused by the *proof* of the problem being complex.

A paradigm shift in the *use* of ATP systems has happened over the last number of years. Problems are not being created only by ATP experts anymore. Novel areas for applications of ATP systems for first-order logic keep popping up; areas whose contents have very little to do with ATP systems, and whose active members are not experts on ATP. People are automatically generating problems for ATP systems from other formalisms. The problem sizes are steadily increasing. Axiomatizations are not carefully crafted anymore. For many application areas, problems are not difficult anymore simply because their proofs are complex; new problems are often difficult because the proofs, which are relatively small, are hard to find — there is often a lot of "junk" irrelevant to the proof, but clogging up the proof search.

In this talk, I would like to argue for the necessity of a corresponding paradigm shift in ATP system design. The key feature that is lacking in almost all of today's most powerful ATP systems is **feedback in the case of a failed proof**. Most ATP systems simply never terminate when presented with a problem that they cannot solve (in some lucky cases termination does occur when a saturation or a finite model has been found).

SAT-solvers I would like to draw a parallel with the development in the area of SAT-solvers. Today, SAT-solvers are (literally) routinely used as black box procedures in many different application areas. There are two key features that make this possible. The first feature is *feedback*; when a SAT-solver terminates, it actually produces more than just a "yes" or a "no" answer. When a problem is deemed satisfiable, an actual model is produced as the result. For most SAT-based algorithms it is vital that this model is produced. Quite quickly after the developments in the SAT community that made SAT-solvers practical, users of SAT-solvers realized that they also wanted feedback when a problem was deemed unsatisfiable. In this case, the SAT-solver can produce a proof of unsatisfiability. The second feature is *incrementality.* After feedback has been examined by the caller of the SAT-solver, small changes can be made to the problem (usually automatically), and the SAT-solver can be run again, reusing as much information from the previous run as possible.

Some of the most industrially successful applications of SAT-solvers (for example interpolant-based model checking) critically depend on getting useful feedback from the SAT-solver in both the ”yes” and the ”no” case, and the use of incrementality.

First-order Logic In first-order logic, things do not look as bright. Although there are a number of areas where ATP systems have successfully been used as black boxes, very little feedback is provided from today’s ATP systems. When a proof is found, most systems are able to produce this. When a proof cannot be found, the result of the ATP system does not contain any kind of *reason* why not.

If we want our ATP systems to be used by non-experts, we need to provide feedback for failed proofs that can be understood by non-experts. Even more important, if we want our ATP systems to be used as a black box component inside a larger system, it is vital that the ATP system provides feedback about why it could not find a proof to its caller! Incremental use of a system is empowered by an order of magnitude if useful feedback is provided between calls.

The kind of feedback I propose to use is that of a **candidate counter model**. One reason why we cannot simply copy what happens in the world of SAT-solvers is that first-order logic is semi-decidable, which means that we cannot always (or rather: we can very seldomly) produce an actual counter model of the problem if there does not exist a proof. What we can do however, is to produce a candidate counter model. A candidate counter model can be seen as an *approximation* of a real counter model; the longer time the ATP system runs, the closer to a real counter model the candidate counter model becomes.

The Talk In the talk, I will describe what such a candidate counter model is, some theory behind it, how candidate counter models can be found, and how they can be used.

It turns out that it is very natural for an ATP system that produces candidate counter models to be *instance based.* We present some experimental evidence that instance based (and thus model-based) methods actually are a very good fit with the above mentioned paradigm shift in ATP usage. I will also discuss the impact that a model-based view has on the internal design of an ATP system; for example, it becomes very important to deal with *typed problems*, and even basic algorithms such as *clausification* should be revisited!

Solution to some right alternative ring problems

Jan Otop

Institute of Computer Science, University of Wrocław
ul. Joliot-Curie 15, PL-50-383, Wrocław, Poland.

Abstract. In this paper we show a computer construction of a counterexample to the TPTP problems RNG030 and RNG032. These problems have been open up to now. The used method shares some ideas with the Z-module method, but it is more semantic oriented.
We deal also with ensuring program correctness. This is harder than in case of proving theorem, because there is no succinct "proof".

1 Introduction

The TPTP is a library of test problems for evaluating theorem provers [3]. We tried to prove the open problems RNG30 and RNG32 which are problems in ring theory. Even after implementing sophisticated data structures for term indexing and different term orderings, the problems remained intractable.

Considering possibility that given hypothesis can be satisfiable, we wrote a dedicated program that disproves these conjectures.

This paper is organized as follows. First we introduce basic notation (section 2) which will be used in this paper. Our method will be introduced in section 3. The implementation of our method will be described in section 4, we will mention there how one can ensure program[1] correctness. In the conclusion we will discuss what can be straightforwardly generalized.

2 Preliminaries

A right alternative ring is an algebraic structure $\mathcal{R} = (R, +, -, 0, *)$ in which $-$ is an unary symbol, satisfying the following axioms:

1. Associativity and commutativity of addition $\forall X, Y, Z\ X + Y = Y + X \wedge X + (Y + Z) = (X + Y) + Z$
2. Neutral element of addition $\forall X\ X + 0 = X$
3. Additive inverse $\forall X\ X + (-X) = 0$
4. Additive inverse is involution $\forall X\ -(-X) = X$
5. Multiplication by additive identity $\forall X\ X0 = 0X = 0$
6. Left distributivity $\forall X, Y, Z\ X(Y + Z) = XY + XZ$
7. Right distributivity $\forall X, Y, Z\ (X + Y)Z = XZ + YZ$

[1] We know that the method is correct, but the question is, whether its implementation is correct.

8. Right alternative $\forall X, Y \; X(YY) = (XY)Y$

These are rings with weaker associativity. Every standard ring is a right alternative ring, but the converse is not true. Multiplication in right alternative rings is nonassociative. The right alternative rings are subset of nonassociative rings. Nonassociative rings are defined similar to the right alternative rings, but satisfy only the axioms 1-7.

The associator (X, Y, Z) is defined as $(X, Y, Z) = (XY)Z + (-X(YZ))$. In rings the associator is always equal to 0. Using distributivity one can show that the associator is additive with respect to any argument, for example $(X_1 + X_2, Y, Z) = (X_1, Y, Z) + (X_2, Y, Z)$. In this paper we will use the abbreviation $X - Y := X + (-Y)$.

Ring $\mathcal{R}$ is called a free right alternative ring with generating set $\mathcal{X}$ if $\mathcal{R}$ is a right alternative ring generated by $\mathcal{X}$ and for every right alternative ring $\mathcal{R}'$ every mapping $\varphi : \mathcal{X} \to \mathcal{R}'$ can be extended to homomorphism $\hat{\varphi} : \mathcal{R} \to \mathcal{R}'$.

For every signature Σ, and every set of elements $\mathcal{X}$ such that $\mathcal{X} \cap \Sigma = \emptyset$ we define set of terms $\mathcal{T}(\Sigma, \mathcal{X})$ as the smallest set such that:

- $\mathcal{X} \subseteq \mathcal{T}(\Sigma, \mathcal{X})$
- for every constant $c \in \Sigma$, we have $c \in \mathcal{T}(\Sigma, \mathcal{X})$
- for every function $f \in \Sigma$ of arity k, and $t_1, ..., t_k \in \mathcal{T}(\Sigma, \mathcal{X})$, we have $f(t_1, ..., t_k) \in \mathcal{T}(\Sigma, \mathcal{X})$

The set of monomials over $\mathcal{X}$ is defined as $\mathcal{T}(\{*\}, \mathcal{X})$. A polynomial is a sum of monomials and additive inverses of monomials.

From now on Σ will denote the ring signature, rather than some abstract signature, so $\Sigma = \{+, *, 0, -\}$. In signature Σ there are constants that are defined by the axioms. Constants in set $\mathcal{X}$ are Skolem symbols introduced by the hypothesis.

For a monomial m we define the degree $deg(m)$ recursively, $deg(c) = 1$ where $c \in \mathcal{X}$ and $deg(m_1 m_2) = deg(m_1) + deg(m_2)$. It can be extended to the polynomials in the usual way, consider polynomial $p = \sum_i m_i$ then $deg(p) = max_i(deg(m_i))$.

For a monomial m let $deg_a(m)$ denote the number of occurrences of the symbol $a \in \mathcal{X}$ in the term m. We extend this definition to polynomials, as follows: for $p = \sum_i m_i$ degree $deg_a(p) = max_i(deg_a(m_i))$ for the polynomial p.

The ring $\mathcal{R}$ is torsion free iff for every $a \in \mathcal{R} \setminus \{0\}$ and a natural number n, we have $na \neq 0$, where $na = a + ... + a$ is added n times.

3 Method description

The problem RNG30 can be written as "in right associative rings of characteristic greater than 2, $(X, X, Y)^3 = 0$ for all X, Y" and in RNG032 the question is essentially the same but assumption is stronger, "in rings with characteristic greater than 6, $(X, X, Y)^3 = 0$ for all X, Y". Although the multiplication in alternative rings in nonassociative, the X^3 is uniquely defined because from the right alternative axiom follows that $X(XX) = (XX)X$.

To falsify the above identities one has to show that there exists a structure in which they are false. How can we construct a right alternative ring that is not an associative ring? Consider a vector space over the rationals with multiplication of elements considered as an arbitrary linear operator[2]. Such a structure satisfies all the axioms except (possibly) the right alternative axiom. How can we define such a linear operator to obtain a right alternative ring?

There exists the free right alternative ring generated by $\{a, b\}$ and it is isomorphic to $\mathcal{T}(\Sigma, \mathcal{X})/\approx_A$ where $\approx_A = \{(s,t) \in \mathcal{T}(\Sigma, \{a,b\}) \times \mathcal{T}(\Sigma, \{a,b\}) : A \vdash s = t\}$ where A is a set of right alternative ring axioms. In other words $\approx_A$ is the equivalence relation of terms which have to be equal in every right alternative ring. Deciding whether two terms are equal in the free right alternative ring is so complicated as proving theorems. However if we have two terms then we can easily compute result of the multiplication.

We will construct a homomorphism from the free right alternative ring into a vector space over rationals. The multiplication in a vector space will be induced by this homomorphism. On the one hand, the multiplication will be well-defined. On the other hand, we will have structure in which we can easily decide if given two elements are equal. The constructed homomorphism will be constant on the equivalence classes of $\approx_A$ so it will induce a homomorphism form the free right alternative ring into the vector space.

3.1 Homomorphism construction

First we prove some general facts about the right alternative rings.

Claim. For every term $t \in \mathcal{T}(\Sigma, \mathcal{X})$ there exist a polynomial p, such that $t = p$ in every right alternative ring.

Proof. The claim above can be proved by a straightforward induction. Using distributivity it can be proved that a product of polynomials can be transformed into a polynomial.

The claim allows us to think about elements of the ring as polynomials. A polynomial can be seen as a finite sum $\sum_{i=0}^{N} n_i m_i$ where n_i are integer coefficients and m_i are different monomials.

Consider a linear mapping $\phi : \mathcal{T}(\Sigma, \{a,b\}) \to \mathcal{V}$ where $\mathcal{V}$ is a vector space over the rationals with countable basis.

Definition 1. *We call $lin(X)$ the linear closure of set X, i.e. $lin(X)$ is the smallest set such that:*

- $X \subseteq lin(X)$
- $q_1 x_1 + q_2 x_2 \in lin(X)$ *for each* $x_1, x_2 \in lin(X)$ *and* $q_1, q_2 \in \mathbb{Q}$

$lin(X)$ is the smallest vector space containing X.

[2] A linear function from vector pairs into vectors

From now on, we will treat $\phi(\mathcal{T}(\Sigma, \{a, b\}))$ as a structure $(U, +, *, -, 0)$ where $U = lin(\phi(\mathcal{T}(\Sigma, \{a, b\})))$, where 0 is the zero in vector space, $+$ coincides with the addition from vector space, $-$ is interpreted as the multiplication by -1 and the multiplication defined as follows $V_1V_2 = \phi(\phi^{-1}(V_1)\phi^{-1}(V_2))$. The multiplication is induced by ϕ from the multiplication in the free right alternative ring. One must only argue that the multiplication is well defined, because ϕ does not need to be injective.

Definition 2. *A function $\phi : \mathcal{T}(\Sigma, \mathcal{X}) \to \mathcal{V}$ defined on polynomials is said to be monotonic if for each term X, Y, Z if X is subterm of Z and $\phi(X) = \phi(Y)$ then $\phi(Z) = \phi(Z[X := Y])$, where $Z[X := Y]$ denotes usual term substitution.*

Notice that monotonicity on monomials implies monotonicity on polynomials. Every ring homomorphism is monotonic.

Claim. Monotonicity of ϕ implies that the multiplication in a vector space induced from the free right alternative ring is well defined.

Proof. Assume, that $v_1 = \phi(x_1) = \phi(x_2)$ and $v_2 = \phi(y_1) = \phi(y_2)$ then $\phi(x_1y_1) = \phi(x_2y_1) = \phi(x_2y_2)$. It shows that $\phi(\phi^{-1}(v_1)\phi^{-1}(v_2))$ is unique assuming monotonicity of ϕ.

ϕ is a ring homomorphism iff it is linear and monotonic, this follows directly from the definition of homomorphism. Taking all together, if ϕ is linear and monotonic then $\phi(\mathcal{T}(\Sigma, \{a, b\}))$ is a nonassociative ring. How can we achieve right alternativity?

Definition 3. *The* linearization of the right alternative axiom $(X, Y, Y) = 0$ *are two axioms $(X, Y, Y) = 0$ and $(X, Y, Z) + (X, Z, Y) = 0$.*

The following lemma justifies name *linearization.*

Lemma 1. *For every ring $\mathcal{R} = (R, +, -, *)$, if $A \subseteq R$ generates R with respect to addition, i.e. every $x \in R$ is a linear combination of elements from A, then $R \models \forall X, Y \in R\ (X, Y, Y) = 0$ iff $R \models \forall X, Y, Z \in A\ (X, Y, Y) = 0 \wedge (X, Y, Z) + (X, Z, Y) = 0$*

Proof.

$$0 = (X, Y + Z, Y + Z) = (X, Y, Y) + (X, Y, Z) + (X, Z, Y) + (X, Z, Z) =$$

$$= (X, Y, Z) + (X, Z, Y) = 0$$

This proves the implication from left to right. To prove the converse, we use the fact that A is a generating set, so X,Y can be written as a sum of elements of A, there exist $n_i, m_j \in A$, such that

$$X = \sum_{i=1}^{K} m_i \qquad Y = \sum_{j=1}^{N} n_j$$

$$(X, Y, Y) = (\sum_{i=1}^{K} m_i, Y, Y) = \sum_{i=1}^{K} (m_i, Y, Y)$$

it is enough to show that for every $1 \leq i \leq K$ holds $(m_i, Y, Y) = 0$.

$$(m_i, Y, Y) = (m_i, \sum_{j=1}^{N} n_j, \sum_{k=1}^{N} n_k) = \sum_{j=1}^{N} \sum_{k=1}^{N} (m_i, n_j, n_k) =$$

$$= \sum_{j=1}^{N} (m_i, n_j, n_j) + \sum_{j=1}^{N} \sum_{k=1}^{j-1} (m_i, n_j, n_k) + (m_i, n_k, n_j) = 0.$$

In the torsion free rings $(X, Y, Z) + (X, Z, Y) = 0$ implies $(X, Y, Y) = 0$ because if we consider $Y = Z$ in the first equation $(X, Y, Y) + (X, Y, Y) = 0$ but since the ring is torsion free $(X, Y, Y) = 0$. In the free right alternative ring it does not need to be true.

From above lemma it follows that, if homomorphism ϕ will be defined on monomials in such a way that it will satisfy linearization of the right alternative axiom, it implies that $\phi(\mathcal{T}(\Sigma, \mathcal{X}))$ satisfies the right alternative axiom. In other words ϕ will be constant on equivalent classes of relation $\approx_A$ so ϕ induce the homomorphism from the right alternative ring into vector space, so its image is a right alternative ring. We cannot define ϕ on the set of monomials, because it is infinite. At least we cannot define it explicitly. However we do not need to care about the "large" monomials, in this case we can define ϕ to be zero on the monomials with a degree larger than the hypothesis without affecting the truth value of hypothesis.

Lemma 2. *Suppose that a homomorphism ϕ is defined, such that $\phi(\mathcal{T}(\Sigma, \{a, b\}))$ is a right alternative ring and let the polynomial $p \in \mathcal{T}(\Sigma, \{a, b\})$ be such that $\phi(p) \neq 0$. Define ψ on monomials such that*

- *$\psi(m) = \phi(m)$ if $deg(m) \leq deg(p)$*
- *$\psi(m) = 0$ otherwise.*

Then $\psi(p) \neq 0$ and ψ $(\mathcal{T}(\Sigma, \{a, b\}))$ is a right alternative ring.

Proof. Since $\phi(m) = \psi(m)$ if $deg(m) \leq deg(p)$ it follows that $\phi(p) = \psi(p)$ so ψ $(\mathcal{T}(\Sigma, \{a, b\})) \models \psi(p) \neq 0$. It is enough to show that ψ satisfies the linearized right alternative axiom.

To show that $\psi((X, Y, Z) + (X, Z, Y)) = 0$ where X, Y, Z are monomials, notice that in $(X, Y, Z) + (X, Z, Y) = (XY)Z - X(YZ) + (XZ)Y - X(ZY)$ all this monomials have equal degree. Then either this degree is greater than $deg(p)$ and all monomials in $(X, Y, Z) + (X, Z, Y)$ are mapped to zero, so $\psi((X, Y, Z) + (X, Z, Y)) = 0$, either they have degree less or equal. In the second case ψ and ϕ coincides on this monomials and it follows that $\psi((X, Y, Z) + (X, Z, Y)) = \phi((X, Y, Z) + (X, Z, Y)) = 0$.

The structure carrier can be even more pruned without changing truth value of our conjecture.

Lemma 3. *Suppose that a homomorphism ϕ is defined, such that $\phi(\mathcal{T}(\Sigma, \{a, b\}))$ is a right alternative ring and let the polynomial $p \in \mathcal{T}(\Sigma, \{a, b\})$ will be such that $\phi(p) \neq 0$. Define ψ on monomials such that*

- $\psi(m) = \phi(m)$ *if* $deg_a(m) \leq deg_a(p)$ *and* $deg_b(m) \leq deg_b(p)$
- $\psi(m) = 0$ *otherwise.*

Then $\psi(p) \neq 0$ and ψ $(\mathcal{T}(\Sigma, \{a, b\}))$ is a right alternative ring.

Proof. Proof is essentially the same as in case of the degree in lemma 2.

4 Implementation

A run of the program can be divided into two stages. First it constructs a linear and monotonic mapping $\phi : \mathcal{T}(\Sigma, \{a, b\}) \to \mathcal{V}$ which was described in section 3. When ϕ is constructed, one can query for vectors which correspond to certain monomials. ϕ is linear and monotonic, so it has unique extension to the polynomials.

Since every term can be transformed to a polynomial, we can decide whether an identity $t_1 = t_2$ holds in our structure by calculating if $\phi(p_1 - p_2) = 0$ where $\mathcal{A} \vdash (t_1 = p_1 \wedge t_2 = p_2)$ and p_1, p_2 are polynomials.

4.1 Construction of linear mapping

The homomorphism ϕ is represented as an associative array, a data structure similar to `std::map`in C++, which associates monomials with vectors. This associative array will be denoted by `T`.

The algorithm can be seen as an inductive construction. Step 0 is the induction basis, no inferences can be done between monomials of length one, so they are simply associated with different vectors which are basic vectors. Next, steps 1, 2 in the algorithm can be considered as the inductive step. Assuming that we have constructed ϕ for every monomial of degree $d' < d$, we will use the definition of ϕ on the monomials of length less than d to construct ϕ for the monomials of length d. These steps are repeated for every $1 < d \leq h$ where h is the degree of the hypothesis.

The number $n(d)$ of monomials of a given degree d can be easily computed form recursive equation $n(1) = |\mathcal{X}|$ (in our case 2) and $n(k) = \sum_{i=1}^{k-1} n(i) * n(k - i - 1)$. There are 862118 monomials constructed from the symbols a, b of degree less than 9 which is the degree of our hypothesis. We are not able to store all this monomials with corresponding vectors in memory. The vectors associated with the monomials can be quite long, counting only nonzero coefficients. Fortunately, we need to keep only a relatively small subset of the whole set of monomials, that is the set of non-redundant monomials which will be defined later.

There are some linear dependencies between monomials, for example a(aa) = (aa)a, so the set of monomials is not a basis of the vector space. A monomial is called basic, if the corresponding vector was unchanged in step 2, while enforcing the linearized right alternative axiom. A resulting vector is called basic vector. A monomial is called non-redundant, if all its proper subterms are basic monomials. In fact, it is equivalent to say that its two immediate subterms are basic, i.e. t has form s_1s_2 and s_1, s_2 are basic monomials. Notice that due to the inductive construction, all proper subterms are constructed and fully resolved.

Lemma 1 was used to ensure that the linearized right alternative axiom on monomials yields right alternativity on all polynomials. But we can use linearization lemma on a smaller set of basic monomials and obtain that it is sufficient to enforce the linearized right alternativity on the basic monomials.

We will define ϕ on the non-redundant monomials, because we need to know what is the result of multiplication of two basic monomials. Multiplying two basic monomials one can obtain a nonbasic monomial, but it will be a non-redundant monomial. They need to be considered in order for ϕ to enjoy monotonicity.

```
0. T[a]:=(1.a), T[b]:=(1.b)
For d=2 to length_of_hypothesis:
   1.  Construct all non-redundant monomials of degree d and
         insert them to associative tab
   2   Enforce linearization of the right alternative axioms on
         monomials of degree d
```

During step 1, the program generates all non-redundant monomials of degree d, and associates them with the basic vectors as follows:

```
1. For k=1 to d-1
2.   For each basic monomial s1 of degree k
3.     For each basic monomial s2 of degree n - k - 1
4.        T[s1s2]:=(1.s1s2)
```

After this step there are no relations between the non-redundant monomials of degree d.

At step 2, the program enforces the linearized right alternative axiom on the basic monomials.

To enforce that $(X, Y, Z) + (X, Y, Z) = 0$ each monomial m with $deg(m) \geq 3$ is decomposed, like $m = a(bc)$ to $[a, b, c]$ or $m = (ab)c$ to $[a, b, c]$. It is also possible that a monomial can be decomposed in two ways, for example $\{(a(bb))(bb)\}$ can be decomposed into $[a, bb, bb]$ and $[a(bb), b, b]$. For each possible decomposition of a monomial m into $[a, b, c]$ such that $b = c$ and a, b, c are basic monomials, we calculate a vector $w = \phi((a, b, c) + (a, c, b)) = \phi((ab)c) - \phi(a(bc)) + \phi((ac)b) - \phi(a(cb))$. According to the right alternative ring axioms w should be equal to 0. If $w \neq 0$ then there is a *conflict* and the program *resolves* the conflict. Since w is nonzero vector, it contains basic vector e_j with nonzero coefficient a_j, therefore $e_j = \frac{1}{a_j}(w - a_je_j)$. Using the last equation, program resolves e_j from every vector. Changing the basic vector of the monomial which corresponds to e_j does not suffice, e_j can occur in many vectors because it could be on the right side of last equation used to resolving some earlier conflict. The choice of e_j is arbitrary, it can change the basis of linear space, but the set of polynomials which are equal to zero is invariant.

We need to convince ourselves that this definition of ϕ on non-redundant monomials yields unique linear and monotonic extension to polynomials. Uniqueness refers to the extension, because ϕ can be defined in many different ways, according to the choice of e_j while resolving conflicts. Step 2 can be seen in other way as enforcing linearity of ϕ because all linear conflicts were resolved. Moreover the images of non-redundant monomials generates $\phi(\mathcal{T}(\Sigma, \{a, b\}))$ from which uniqueness of extension follows. We need to argue that there are no conflicts with monotonicity.

Consider non-redundant monomials m_1, m_2 with subterms, respectively m_3, m_4. m_3, m_4 are basic monomials, so $\phi(m_3) = \phi(m_4)$ implies that $m_3 = m_4$ and $m_1[m_3/m_4] = m_1$. Indeed ϕ defined on the non-redundant monomials is monotonic.

Notice that in the steps 1,2 of the algorithm, according to the lemma 3, one can consider only monomials m such that $deg_a(m) \leq 6$ and $deg_b(m) \leq 3$ because $deg_a((a, a, b)^3) = 6$ and $deg_b((a, a, b)^3) = 3$.

4.2 Calculating vectors

ϕ is defined on the non-redundant monomials, since ϕ is monotonic and linear it is enough to calculate the value of every polynomial. Here is the function that calculates a vector that corresponds to the monomial m.

```
Calculate(m)
1.  If m is non-redundant return Phi(m)
2.  Decompose m = m1 m2
3.  v1 = calculate(m1)
4.  v2 = calculate(m2)
5.  result = 0
6.  For each basic vector w1 in v1
7.     For each basic vector w2 in v2
8.        n1 is basic monomial to which corresponds w1
9.        n2 is basic monomial to which corresponds w2
10.       result += coff(v1,w1)*coff(v2,w2)*Phi(n1n2)
11. Return result
```

`Phi(m)` returns a vector corresponding to the non-redundant monomial and `coff(v,n)` if coefficient with which the monomial `n` is taken in the vector `v`.

4.3 Program results

Our program has no input. Equation $(a, a, b)^3 = 0$ is built-in. Considering monomials with degree not greater than 9 program runs 3 hours on a 1.7 GHz Intel

Core Duo and uses approximately 400 MB of memory. As output it writes out the vector which corresponds to $(a, a, b)^3$. The computed vector is nonzero which means that $\phi((a, a, b)^3)) \neq 0$ so RNG030 and RNG032 are satisfiable. Considering only monomials m with $deg_a(m) \leq 6$ and $deg_b(m) \leq 3$ resulted in significant speedup to 15 minutes and memory consumption to 80 MB. We anticipated high memory consumption (the number of monomials of the degree less than the hypothesis can be easyli computed), so the program is implemented in C++.

Before writing out results, the program performs the consistency checking.

4.4 Program testing

How can we convince ourselves that the program implementation is correct? The generated structure is quite large, so it cannot be checked manually. Taking into account the program construction, we can write a testing module. After the first stage when the structure has been constructed one can query about the vectors which correspond to polynomials. Querying is quite cheap so we can query about all the axioms.

In our program we implemented tests if:

1. for monomials X, Y mapped by ψ on nonzero elements $\psi((X, Y, Y)) = 0$
2. for monomials X, Y, Z mapped by ψ on nonzero elements $\psi((X, Y, Z) + (X, Z, Y)) = 0$
3. for a monomial X with the degree less or equal than the hypothesis (or additionally with the number of occurrences a, b less or equal than in the hypothesis) whether $\phi(X) \neq 0$
4. for monomials X, Y, Z mapped by ψ on nonzero elements, if X is a subterm of Z and $\phi(X) = \phi(Y)$ then $\phi(Z) = \phi(Z[X/Y])$ (monotonicity)

Thinking about ϕ as a "black-box" and assuming only linearity, we need to test 1,2 and 4 to ensure that $\phi(\mathcal{T}(\Sigma, \mathcal{X}))$ is a right alternative ring. Although checking 1 and 2 takes notably less time than constructing ϕ, testing monotonicity is a very time consuming process.

Testing 3 is not needed because we have a model which is counterexample. But this is an additional consistency constraint. We know that every monomial is not equal to zero because there are torsion free models in which it is unequal to zero (for example standard associative $\mathbb{Z} \times \mathbb{Z}$ with $a = (2, 1)$ and $b = (1, 2)$)

5 Conclusions

The presented method allowed us to disprove two open problems from TPTP. The generated structure is infinite but it is enough to represent only a finite part. As far as we know there are no other systems that can produce such structure because of the description size. Although in this paper we have referred to specify equations, there are possible numerous generalizations.

5.1 Possible generalizations

Without any changes one can try to disprove any identity containing only two variables. The hypothesis after transformation to polynomial should have degree less or equal then 9. Due to the memory consumption (80 MB in case of proving the main hypothesis), the degree could be incremented to 10 or even 11, but not larger.

The method can be straightforwardly generalized to disprove identities with a greater number of variables. There are also possible generalizations to the alternative rings (the rings that are left and right alternative). More general, this method can be applied to any nonassociative ring which extra axioms after modification (like the right alternativity in case of the right alternative rings), which behave like right alternativity, that is

- the axioms transformed to the polynomials are homogeneous polynomials, i.e. all monomials in a given polynomial have equal degree
- the set of axioms has a linearization, i.e. it can be transformed to an equivalent set of axioms called linearization, such that if the linearization holds on any generating set then axioms holds on all polynomials.

5.2 Comparison with the Z-module method

This paper shares some important ideas with the Z-module method[2]. However in the Z-module method a structure is represented in a different way. In the Z-module method, in the first step there is generated a set of polynomials which are instances of the linearized right alternative axiom. Not all such instances are generated, only those which can infer with the hypothesis. All the polynomials in such a set are equal to zero, so in the second step the algorithm decides whether the difference between both sides in the hypothesis is a linear combination with integer coefficients of the polynomials generated in the first step. If it is then the difference between both sides of hypothesis is equal to zero in every right alternative ring, so the hypothesis is theorem.

Although there are some common ideas, the Z-module method is a rather syntactic method and it is oriented on proving theorems. In our method we are taking into account monotonicity of multiplication to introduce a notation of basic monomial, that is the knowledge about equality on subterms propagate to decrease the number of considered monomials. In the Z-module method there is no such analogon. The number of polynomials generated in step one is more than ten times larger than the number of the basic monomials in our method. The increase in memory consumption dramatic.

6 Acknowledgment

I would like to thank Hans de Nivelle from University of Wroclaw for his help in writing this paper. Also, I thank the anonymous referees for their helpful comments.

References

1. Baader,F. and Nipkow, T.: Term Rewriting and All That. Cambridge University Press, 1998
2. Stevens, R. L.: Some Experiments in Nonassociative Ring Theory with an Automated Theorem Prover. J. Autom. Reasoning 3(2): 211-221 (1987)
3. Sutcliffe, G. and Suttner, C.B.: The TPTP Problem Library: CNF Release v3.2.0 `http://www.cs.miami.edu/~tptp/`

Provability and Countermodels in Gödel-Dummett Logics

D. Galmiche[†] and D. Larchey-Wendling[♭] and Y. Salhi[†]

LORIA – UHP Nancy 1[†] – CNRS[♭]
Campus Scientifique, BP 239
54 506 Vandœuvre-lès-Nancy, France

Abstract. Hypersequent calculi, that are a generalization of sequent calculi, have been studied for Gödel-Dummett logics LC and LC_n. In this paper we propose a new characterization of validity in these logics from the construction of particular bi-colored graphs associated to hypersequents and the search of specific chains in such graphs. It leads to other contributions that are a new hypersequent calculus and a related tableau system for LC_n. We mainly study the class of so-called basic hypersequents and then we generalize our approach to hypersequents.

1 Introduction

Gödel-Dummett logic LC and its finitary versions $(\mathsf{LC}_n)_{n>0}$ were introduced by Gödel and later axiomatized by Dummett in [8]. They are *intermediate* logics (between classical and intuitionistic logics) with semantics based on linear Kripke models. It has a Hilbert axiomatic system composed of axioms of intuitionistic logic and the axiom $A \to B \vee B \to A$. One of its interests lies in its relationship with fuzzy logics [12] and recently LC_n logics have been characterized as resource use bounding logics for some particular process calculus [14].
There exist various calculi dedicated to proof-search in LC like sequent calculi [9], sequent of relations calculi [6], tableau calculi [1], goal-directed calculi [17], decomposition systems [4] and also based-on bi-colored graphs calculi [16]. Hypersequent calculi, that generalize sequent calculi, have been also studied [3,5,10]. In order to propose decision procedures from sequent or hypersequent calculi an interesting approach consists in defining local and invertible proof rules, in reducing a (hyper)sequent into a set of irreducible (hyper)sequents and in defining an algorithm to decide such (hyper)sequents [13]. For instance [3] presents a decision procedure from a hypersequent calculus by using particular hypersequents, called *basic hypersequents* [3].
In this work we focus on hypersequent calculi mainly with countermodel search that is not developed in the above-mentioned works. Our alternative approach for deciding hypersequents is based on two main steps: the construction of a semantic graph called *bi-colored graph* and a characterization of validity based on detection of particular chains in this graph. The idea to characterize validity in a given logic from a semantic graph and its analysis has been already studied in non-classical logics, for instance in BI (logic of Bunched Implications) with resource graphs [11]. The notion of bi-colored graph has been recently defined to deal with formulae in LC and LC_n [13], but its possible use for

hypersequents is a non-trivial question to solve. In fact a hypersequent calculus incorporates a sequent calculus as a sub-calculus but adds an additional layer of information by considering a sequent to live in the context of finite multisets of sequents (called hypersequents) [5]. It includes rules for exchanging information between sequents that make it powerful but proof-search methods for sequents are not well-adapted to such structures.
A first contribution concerns the *basic hypersequents* [3], for which we define new characterizations of validity for LC and LC_n. They are based on the construction of a specific bi-colored graph associated to a basic hypersequent and on the search of particular chains in such a graph. The detection of such chains and the generation of countermodels can be realized in linear time [13]. Using the result that for every hypersequent G one can find a set $\mathcal{B}$ of basic hypersequents such that G is valid if and only if every element of $\mathcal{B}$ is valid [3] we also provide new decision procedures for hypersequents in LC and LC_n with a focus on countermodel construction. The study of bi-colored graphs associated to hypersequents leads to other important contributions that complete the results of [3]: a new hypersequent calculus and a related tableau system for LC_n. In the above mentioned results we start with a particular class of hypersequents but it seems important to study if we can directly deal with (general) hypersequents and define associated bi-colored graphs in order to characterize provability. From this perspective we can apply to a given hypersequent an indexing process defined in [13] and then associate a so-called indexed flat sequent from which a bi-colored graph can be built. Then we can define a procedure that decides validity from the detection of particular chains in all instances of such a graph. The key point is that, if one of these instances contains no particular chains, then we can extract a countermodel from this instance. The results are obtained first for mono-conclusioned hypersequents but also hold for (multi-conclusioned) hypersequents. As a sequent is a specific hypersequent, our procedure also provides by specialization a new procedure to decide sequents and generate countermodels in Gödel-Dummett Logics.

2 Gödel-Dummett logics

In this section, we consider the family of propositional Gödel-Dummett logics LC_n. The value n belongs to the set $\overline{\mathbb{N}}^* = \{1,2,\ldots\} \cup \{\infty\}$ of strictly positive natural numbers with its natural order $\leqslant$, augmented with a greatest element ∞. In the case $n = \infty$, the logic LC_∞ is also denoted by LC: this is the usual Gödel-Dummett logic.
Let us start by reminding the key points about semantics and proof theory. The set of propositional *formulae*, denoted Form is defined inductively, starting from a set of propositional *variables* denoted by Var with an additional bottom constant $\perp$ denoting *absurdity* and using the connectives $\wedge$, $\vee$ and $\rightarrow$. IL denotes the set of formulae that are provable in any intuitionistic propositional calculus and CL denotes the classically valid formulae. As usual an *intermediate propositional logic* [1] is a set of formulae $\mathcal{L}$ satisfying $\mathsf{IL} \subseteq \mathcal{L} \subseteq \mathsf{CL}$ and closed under the rule of modus ponens and under arbitrary substitution. In the case of LC, the logic has a simple Hilbert axiomatic system: $(A \rightarrow B) \vee (B \rightarrow A)$ added to the axioms of IL but also a based-on sequent formulation.

From the semantic point of view Gödel-Dummett logic is characterized by the linear Kripke models of size n (see [8].) The following strictly increasing sequence holds: $\mathsf{IL} \subset \mathsf{LC} = \mathsf{LC}_\infty \subset \cdots \subset \mathsf{LC}_n \subset \cdots \subset \mathsf{LC}_1 = \mathsf{CL}$ Moreover there exists an algebraic semantics characterization of LC_n [3]. Let $n \in \overline{\mathbb{N}}^*$, the algebraic model is the set $\overline{[0,n)} = [0,\dots,n[\cup\{\infty\}$ composed of $n+1$ elements. An interpretation of propositional variables $[\![\cdot]\!] : \mathsf{Var} \to \overline{[0,n)}$ is inductively extended to formulae: $\bot$ interpreted by 0, the conjunction $\wedge$ is interpreted by the *minimum* function denoted $\wedge$, the disjunction $\vee$ by the *maximum* function $\vee$ and the implication $\to$ by the operator $\rightarrowtriangle$ defined by $a \rightarrowtriangle b =$ if $a \leqslant b$ then ∞ else b. Then we have $[\![A \wedge B]\!] = [\![A]\!] \wedge [\![B]\!]$, $[\![\bot]\!] = 0$, $[\![A \vee B]\!] = [\![A]\!] \vee [\![B]\!]$, $[\![A \to B]\!] = [\![A]\!] \rightarrowtriangle [\![B]\!]$. A formula D is *valid* for the interpretation $[\![\cdot]\!]$ if the equality $[\![D]\!] = \infty$ holds. This interpretation is complete for LC. A countermodel of a formula D is an interpretation $[\![\cdot]\!]$ such that $[\![D]\!] < \infty$.
For a sequent $\Gamma \vdash \Delta$, with Γ, Δ multisets of formulae, and a given interpretation $[\![\cdot]\!]$ we interpret $\Gamma \equiv A_1, \dots, A_n$ and $\Delta \equiv B_1, \dots, B_p$ by: $\lfloor\!\lfloor \Gamma \rfloor\!\rfloor = [\![A_1]\!] \wedge \cdots \wedge [\![A_p]\!]$, $\lfloor\!\lfloor \emptyset \rfloor\!\rfloor = \infty$ and $\lceil\!\lceil \Delta \rceil\!\rceil = [\![B_1]\!] \vee \cdots \vee [\![B_q]\!]$, $\lceil\!\lceil \emptyset \rceil\!\rceil = 0$. Then a sequent is *valid* for the interpretation $[\![\cdot]\!]$ if $\lfloor\!\lfloor \Gamma \rfloor\!\rfloor \leq \lceil\!\lceil \Delta \rceil\!\rceil$. Moreover $[\![\cdot]\!]$ is a countermodel of $\Gamma \vdash \Delta$ if $\lceil\!\lceil \Delta \rceil\!\rceil < \lfloor\!\lfloor \Gamma \rfloor\!\rfloor$.

From the proof-theoretic point of view there exist various calculi in LC mainly based on sequent calculi [6,9] but we consider here hypersequent calculi introduced as a natural generalization of Gentzen's sequent calculi [3,17]. A hypersequent calculus is defined by incorporating a sequent calculus as a sub-calculus and adding an additional layer of information. It allows to define rules for information exchange between sequents [5].
A *hypersequent* is a structure $\Gamma_1 \vdash \Delta_1 \mid \Gamma_2 \vdash \Delta_2 \mid \dots \mid \Gamma_n \vdash \Delta_n$ in which $\Gamma_i \vdash \Delta_i$ is a sequent, called a *component* of the hypersequent. Let us remark that sequents (resp. hypersequents) are multisets of formulae (resp. sequents). A hypersequent is mono-conclusioned if the Δ_i's consist of at most one formula. The symbol $\mid$ denotes a disjunction at the meta-level. Hypersequent calculi consists of axioms, structural and logical rules like in sequent calculi but structural rules are divided into internal and external rules. The first ones deal with formulae within components and the other manipulate whole components of a hypersequent.
The hypersequent calculus HG (Figure 1) for LC is an extension of the hypersequent calculus for intuitionistic logic HIL [9] with the *communication* rule $[com]$. A hypersequent can be seen as a multi-processor [2] and from this perspective the (com) rule fixes the way of exchanging information between processes. As an illustration we prove the axiom $(A \to B) \vee (B \to A)$ in HG that is not valid in intuitionistic logic.

$$\dfrac{\dfrac{\dfrac{A \vdash A \quad B \vdash B}{A \vdash B \mid B \vdash A}\,[com]}{\vdash A \to B \mid \vdash B \to A}\,[\to_R]}{\vdash (A \to B) \vee (B \to A)}\,[\vee_R]$$

We observe that it is not possible to derive $(A \to B) \vee (B \to A)$ in HG without using the $[com]$ rule. Let $\mathcal{H} = \Gamma_1 \vdash \Delta_1 \mid \Gamma_2 \vdash \Delta_2 \mid \dots \mid \Gamma_m \vdash \Delta_m$ be a hypersequent and $[\![\cdot]\!] : \overline{[0,n)}$ be an interpretation. $\mathcal{H}$ is *valid* for the interpretation $[\![\cdot]\!]$ iff there exists $i \in [1,m]$ such that $\lfloor\!\lfloor \Gamma_i \rfloor\!\rfloor \leq \lceil\!\lceil \Delta_i \rceil\!\rceil$. Then $[\![\cdot]\!]$ is a *countermodel* of $\mathcal{H}$ iff $\forall i \in [1,m]$, $\lceil\!\lceil \Delta_i \rceil\!\rceil < \lfloor\!\lfloor \Gamma_i \rfloor\!\rfloor$.

Axioms

$$A \vdash A$$

Cut rule

$$\frac{G \mid \Gamma' \vdash A \qquad G' \mid A, \Gamma \vdash C}{G \mid G' \mid \Gamma, \Gamma' \vdash C}\,[cut]$$

External structural rules

$$\frac{G}{G \mid \Gamma \vdash A}\,[ew] \qquad \frac{G \mid \Gamma \vdash A \mid \Gamma \vdash A}{G \mid \Gamma \vdash A}\,[ec]$$

Internal structural rules

$$\frac{G \mid \Gamma \vdash C}{G \mid \Gamma, A \vdash C}\,[w,l] \qquad \frac{G \mid \Gamma, A, A \vdash C}{G \mid \Gamma, A \vdash C}\,[c,l]$$

Logical rules

$$\frac{G \mid \Gamma, A, B \vdash C}{G \mid \Gamma, A \wedge B \vdash C}\,[\wedge_L] \qquad \frac{G \mid \Gamma \vdash A \qquad G \mid \Gamma \vdash B}{G \mid \Gamma \vdash A \wedge B}\,[\wedge_R]$$

$$\frac{G \mid \Gamma, A \vdash C \qquad G \mid \Gamma, B \vdash C}{G \mid \Gamma, A \vee B \vdash C}\,[\vee_L] \qquad \frac{G \mid \Gamma \vdash A \mid \Gamma \vdash B}{G \mid \Gamma \vdash A \vee B}\,[\vee_R]$$

$$\frac{G \mid \Gamma \vdash A \qquad G' \mid \Gamma, B \vdash C}{G \mid G' \mid \Gamma, A \rightarrow B \vdash C}\,[\rightarrow_L] \qquad \frac{G \mid \Gamma, A \vdash B}{G \mid \Gamma \vdash A \rightarrow B}\,[\rightarrow_R]$$

Special structural rule

$$\frac{G \mid \Gamma, \Gamma' \vdash A \qquad G' \mid \Gamma_1, \Gamma'_1 \vdash A'}{G \mid G' \mid \Gamma, \Gamma'_1 \vdash A \mid \Gamma', \Gamma_1 \vdash A'}\,[com]$$

Fig. 1. The Hypersequent Calculus *HG* for LC

Proof-search in LC and in some intermediate logics is based on different calculi: a contraction-free calculus derived from intuitionistic logic [1,9], sequent or hypersequent of relations calculi in LC [6,7] and more generally in many-valued logics and hypersequent calculi [3,18]. Some refinements, based on local and invertible rules, have been proposed for sequents or hypersequents with semantic criteria to decide irreducible sequents or hypersequents [3]. Here we aim at studying validity and proof-search in LC and LC_n with hypersequent calculi in a new perspective based on countermodel construction from so-called bi-colored graph introduced in [13,15].

3 A new procedure for basic hypersequents

Before starting to study a particular class of hypersequents, namely the basic hypersequents [3], let us remind what is a *bi-colored graph* in this context.

Definition 3.1. *A (conditional)* bi-colored graph *is a finite oriented graph with two kinds of arrows, the green ones represented by* $\rightarrow$ *and the red ones represented by* $\Rightarrow$*,* *that are indexed by boolean formulae. The boolean variables of these formulae can be instantiated by* $\{0,1\}$ *through a valuation. Moreover an* instance *of the graph is the bi-colored graph with only the arrows indexed by an expression* e *with* $v(e) = 1$.

We use the symbols $\rightarrow$ and $\Rightarrow$ to denote the corresponding relation in the graph. For example $\rightarrow\Rightarrow$ represents the composition of two relations and $u\rightarrow\Rightarrow w$ means that there exists a path $u\rightarrow v\Rightarrow w$ in the graph. The relation $\rightarrow^\star$ is the reflexive and transitive closure of $\rightarrow$, i.e, the accessibility of the relation $\rightarrow$. Moreover $\rightarrow + \Rightarrow$ is the union of both relations and $\overline{x}$ denotes the negation of the boolean expression x.

Definition 3.2. *Let* $\mathcal{G}$ *be a bi-colored graph, a* $\Rightarrow$-cycle *of* $\mathcal{G}$ *is a chain of the form* $u(\rightarrow + \Rightarrow)^\star \Rightarrow u$ *and a* k-alternating chain *of* $\mathcal{G}$ *is a chain of the form* $(\rightarrow^\star\Rightarrow)^k$.

Therefore the key point of our approach consists in associating a bi-colored graph to a given hypersequent and in relating validity in the given logic with the existence of $\Rightarrow$-cycle or k-alternating chain. Let us start this study with particular hypersequents.

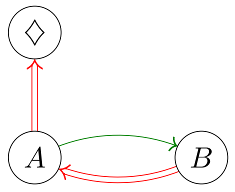
Definition 3.3. *A* basic hypersequent *is a hypersequent such that any component is either* $\Gamma \vdash p$ *where* p *and any element of* Γ *are atoms, or* $p\rightarrow q \vdash p$ *where* p *and* q *are atoms and* $p \neq q$, $p \neq \bot$.

Let $\mathcal{H} = S_1 \mid \ldots \mid S_k$ be a basic hypersequent, the bi-colored graph $\mathcal{G}_\mathcal{H}$ associated to $\mathcal{H}$ is built as follows:
- the *nodes* are: the variables of $\mathcal{H}$, a node denoted $\diamondsuit$ and a node $\bot$ if $\mathcal{H}$ contains $\bot$.
- the *arrows* are the union of the set B and arrows $A_{i\in[1,k]}$ where

$$B = \begin{cases} \emptyset & \textit{if } \mathcal{H} \textit{ does not contain } \bot \\ \{\bot \rightarrow p, \text{ for any } p \in \mathsf{Var}\} & \textit{otherwise} \end{cases}$$

and $A_{i\in[1,k]}$ associated to the components $S_{i\in[1,k]}$ of $\mathcal{H}$ defined as follows: if $S_i = p\rightarrow q \vdash p$ then $A_i = \{p\rightarrow q, p \Rightarrow \diamondsuit\}$, else if $S_i = q_1,\ldots,q_m \vdash p$ then $A_i = \{p \Rightarrow q_1, \ldots, p \Rightarrow q_m\}$.

Let us illustrate this construction with the basic hypersequent $\mathcal{H}_1 \equiv A\rightarrow B \vdash A \mid A \vdash B$. The bi-colored graph associated to $\mathcal{H}_1$ is the following:



Proposition 3.1. *Let* $\mathcal{H}$ *be a basic hypersequent and* $\mathcal{G}_\mathcal{H}$ *be its associated bi-colored graph. Let* $[\![\cdot]\!]$ *be a countermodel of* $\mathcal{H}$ *in* LC_n *(extended with* $[\![\diamondsuit]\!] = \infty$*) and* $X_1 \rightarrow \ldots \rightarrow X_k \Rightarrow Y$ *be a chain in* $\mathcal{G}_\mathcal{H}$*. Then we have* $[\![X_1]\!] \leqslant \ldots \leqslant [\![X_k]\!] < [\![Y]\!]$.

Proof. Let $\mathcal{H} = S_1 \mid \ldots \mid S_k$ be a basic hypersequent. As $[\![\cdot]\!]$ is a countermodel of $\mathcal{H}$ then $[\![\cdot]\!]$ is a countermodel of all the components $S_{i\in[1,k]}$ of $\mathcal{H}$. If $Y \neq \diamondsuit$ then there exists a component S_i such that X_k is the conclusion of S_i and Y belongs to the multiset of S_i hypotheses. Thus we have $[\![X_k]\!] < [\![Y]\!]$. If $Y = \diamondsuit$ then there exists a component S_i having X_k as conclusion. Therefore $X_k < [\![\diamondsuit]\!] = \infty$. Moreover for all $j \in [2,k]$ we have $X_{j-1} \rightarrow X_j \in \mathcal{G}_{\mathcal{H}}$ and there exists a component $S_i = X_{j-1} \rightarrow X_j \vdash X_{j-1}$ or $X_{j-1} = \perp$. Then we deduce that $[\![X_{j-1}]\!] \leqslant [\![X_j]\!]$ and $[\![X_1]\!] \leqslant \ldots \leqslant [\![X_k]\!] < [\![Y]\!]$.

Moreover we can define, from a bi-colored graph $\mathcal{G}$, the notion of *bi-height* that is a function $h : \mathcal{G} \rightarrow \mathbb{N}$ such that for any $u, v \in \mathcal{G}$, if $u \rightarrow v \in \mathcal{G}$ then $h(u) \leq h(v)$ and if $u \Rightarrow v \in \mathcal{G}$ then $h(u) < h(v)$ [13]. Then a countermodel can be generated from $\mathcal{G}$ by using the following results: if a bi-colored graph $\mathcal{G}$ does not contain a $\Rightarrow$-cycle (resp. a n-alternating chain) then there exists a bi-height h (resp. that satisfies $h(v) < n$ for any $v \in \mathcal{G}$) [13]. Moreover it is known that we can decide if a graph instance contains or not a $\Rightarrow$-cycle and also compute the bi-height both in a linear time [15].

Proposition 3.2. *Let $\mathcal{H}$ be a basic hypersequent containing $\perp$. If there exists a bi-height h for $\mathcal{G}_{\mathcal{H}}$ then the function h' defined by: $h'(X) = 0$ if $h(X) = h(\perp)$ and $h'(X) = h(X)$ if $h(X) \neq h(\perp)$, is a bi-height for $\mathcal{G}_{\mathcal{H}}$.*

Proof. From the set of arrows B defined in the construction of the bi-colored graphs associated to the basic hypersequents.

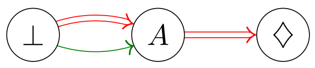
Theorem 3.1 **($n < \infty$).** *A basic hypersequent $\mathcal{H}$ has a countermodel in* LC_n *if and only if its bi-colored graph $\mathcal{G}_{\mathcal{H}}$ does not contain a $(n+1)$-alternating chain.*

Proof. First we prove the *if* part. Let $\mathcal{H} = S_1 \mid \ldots \mid S_k$ be a basic hypersequent. We suppose that $\mathcal{G}_{\mathcal{H}}$ does not contain a chain of the form $(\rightarrow^{\star}\Rightarrow)^{n+1}$. Then there exists a bi-height $h : \mathcal{G}_{\mathcal{H}} \rightarrow [0,n]$. By Proposition 3.2, we define from h a new bi-height $h' : \mathcal{G}_{\mathcal{H}} \rightarrow [0,n]$ by: $h'(X) = 0$ if $h(X) = h(\perp)$ and $h'(X) = h(X)$ if $h(X) \neq h(\perp)$ and then we define the semantic function $[\![\cdot]\!] : \mathsf{Var} \rightarrow \overline{[0,n)}$ by: $[\![X]\!] = h'(X)$ if $h'(X) < n$ and $[\![X]\!] = \infty$ if $h'(X) = n$. We prove that $[\![\cdot]\!]$ is a countermodel of $\mathcal{H}$, i.e., a countermodel of any component $S_{i \in [1,k]}$.
(i) if $S_i = p \rightarrow q \vdash p$ then $p \rightarrow q \in \mathcal{G}_{\mathcal{H}}$ and $p \Rightarrow \diamondsuit \in \mathcal{G}_{\mathcal{H}}$. Thus, we have $h'(p) \leqslant h'(q)$ and $h'(p) < n$. We deduce that $[\![p]\!] \leqslant [\![q]\!]$ and $[\![p]\!] < \infty$. Thus we have $[\![p \rightarrow q]\!] = \infty$ and $[\![p]\!] < \infty = [\![p \rightarrow q]\!]$. Consequently $[\![\cdot]\!]$ is a countermodel of S_i.
(ii) if $S_i = q_1, \ldots, q_m \vdash p$ then $\forall i \in [1,m]$, $p \Rightarrow q_i \in \mathcal{G}_{\mathcal{H}}$ and $\forall i \in [1,m]$, $h'(p) < h'(q_i)$. We deduce that $\forall i \in [1,m]$, $[\![p]\!] < [\![q_i]\!]$ and $[\![\cdot]\!]$ is a countermodel of S_i.
From (i) and (ii) we deduce that $[\![\cdot]\!]$ is a countermodel of $\mathcal{H}$.
We now prove the *only if* part. Let $[\![\cdot]\!]$ be a countermodel of $\mathcal{H}$, we define a new interpretation $[\![\cdot]\!]'$ by: $[\![V]\!]' = [\![V]\!]$ for any variable V of $\mathcal{H}$ and $[\![\diamondsuit]\!]' = \infty$. As $[\![\cdot]\!]'$ and $[\![\cdot]\!]$ have the same values for $\mathcal{H}$'s atoms, we deduce that $[\![\cdot]\!]'$ is a countermodel of $\mathcal{H}$. We suppose that there exists a chain of the form $(\rightarrow^{\star}\Rightarrow)^{n+1}$ in $\mathcal{G}_{\mathcal{H}}$: $X_0 \rightarrow^{\star} \Rightarrow X_1 \rightarrow^{\star} \Rightarrow X_2 \rightarrow^{\star} \Rightarrow \ldots \rightarrow^{\star} \Rightarrow X_n \rightarrow^{\star} \Rightarrow X_{n+1}$. Thus, by Proposition 3.1, the sequence $[\![X_0]\!] < [\![X_1]\!] < [\![X_2]\!] < \ldots < [\![X_n]\!] < [\![X_{n+1}]\!]$ is a strictly increasing sequence of $n+2$ elements in $\overline{[0,n)}$. As this set does contain only $n+1$ elements, that is contradictory.

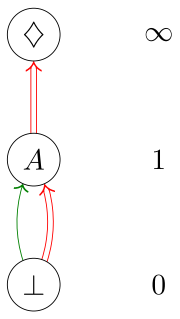
Theorem 3.2 **($n = \infty$).** *A basic hypersequent $\mathcal{H}$ has a countermodel in* LC *if and only if its bi-colored graph $\mathcal{G}_{\mathcal{H}}$ does not contain a $\Rightarrow$-cycle.*

Proof. For the *if* part: if $\mathcal{G}_{\mathcal{H}}$ does not contain a $\Rightarrow$-cycle, we know that there exists a bi-height $h : \mathcal{G}_{\mathcal{H}} \to \mathbb{N}$. By Proposition 3.2, we define from h a new bi-height $h' : \mathcal{G}_{\mathcal{H}} \to \mathbb{N}$ by: $h'(X) = 0$ if $h(X) = h(\bot)$ and $h'(X) = h(X)$ if $h(X) \neq h(\bot)$. By defining $[\![X]\!] \in \mathbb{N} \cup \{\infty\}$ by and $[\![X]\!] = h'(X)$ we obtain a countermodel of $\mathcal{H}$ in LC. For the *only if* part: the existence of a chain $X \to^{\star} \Rightarrow \to^{\star} \ldots \to^{\star} \Rightarrow \to^{\star} \Rightarrow X$ implies $[\![X]\!] < [\![X]\!]$ and then we have a contradiction.

Coming back to our example we observe that the bi-colored graph associated to $\mathcal{H}_1$ contains a $\Rightarrow$-cycle: $A \to B \Rightarrow A$. Then we conclude that $\mathcal{H}_1$ has no countermodel in LC. Let us give another example with the basic hypersequent $\mathcal{H}_2 \equiv \vdash A \mid A \vdash \bot$. The bi-colored graph associated to $\mathcal{H}_2$ is the following:



This graph has one instance and does not contain a $\Rightarrow$-cycle. In order to extract a countermodel we modify the previous graph in such a way that red arrows always go up and greens arrows never go down.



Then $[\![\cdot]\!] : \mathsf{Var} \to \overline{[0,n)}$ such that $[\![A]\!] = 1$ is a countermodel of $\mathcal{H}_2$ in LC_n for $n \geqslant 2$.

A decision procedure for basic hypersequents has been already provided but only for LC [3]. It is based on the generation of constraints and some criteria for solving them. Here we define new criteria, based on bi-colored graphs, to decide the basic hypersequents in LC but also in $(\mathsf{LC}_n)_{n>0}$. Moreover, the extraction of countermodels can be realized from the bi-colored graphs associated to hypersequents.

4 New results for LC and LC_n

In order to propose new results for (general) hypersequents from the above results on basic hypersequents, we can relate them to the system GLC^* and some results of [3].

4.1 Decision procedures for hypersequents

The main one is that for every hypersequent G one can effectively find a set $\mathcal{B}$ of basic hypersequents, so that G is valid if and only if $\mathcal{H}$ is valid for every $\mathcal{H} \in \mathcal{B}$.

$$\frac{G \mid \Gamma, A, B \vdash C}{G \mid \Gamma, A \wedge B \vdash C}\,[\wedge_L] \qquad \frac{G \mid \Gamma \vdash A \qquad G \mid \Gamma \vdash B}{G \mid \Gamma \vdash A \wedge B}\,[\wedge_R]$$

$$\frac{G \mid \Gamma, A \vdash C \qquad G \mid \Gamma, B \vdash C}{G \mid \Gamma, A \vee B \vdash C}\,[\vee_L] \qquad \frac{G \mid \Gamma \vdash A \mid \Gamma \vdash B}{G \mid \Gamma \vdash A \vee B}\,[\vee_R]$$

$$\frac{G \mid \Gamma, A \to B, A \to C \vdash D}{G \mid \Gamma, A \to (B \wedge C) \vdash D}\,[\to\wedge_L] \qquad \frac{G \mid \Gamma, A \to B \vdash D \qquad G \mid \Gamma, A \to C \vdash D}{G \mid \Gamma, A \to (B \vee C) \vdash D}\,[\to\vee_L]$$

$$\frac{G \mid \Gamma, A \to C \vdash D \qquad G \mid \Gamma, B \to C \vdash D}{G \mid \Gamma, (A \wedge B) \to C \vdash D}\,[\wedge\to_L] \qquad \frac{G \mid \Gamma, A \to C, B \to C \vdash D}{G \mid \Gamma, (A \vee B) \to C \vdash D}\,[\vee\to_L]$$

$$\frac{G \mid \Gamma, A \to C \vdash D \qquad G \mid \Gamma, B \to C \vdash D}{G \mid \Gamma, A \to (B \to C) \vdash D}\,[\to(\to)_L] \qquad \frac{G \mid A \vdash B \mid \Gamma, B \to C \vdash D \qquad G \mid \Gamma, C \vdash D}{G \mid \Gamma, (A \to B) \to C \vdash D}\,[(\to)\to_L]$$

$$\frac{G \mid \Gamma \vdash r \mid p \to q \vdash p \qquad G \mid \Gamma, q \vdash r}{G \mid \Gamma, p \to q \vdash r}\,[\to_L] \qquad \frac{G \mid \Gamma, A \vdash B}{G \mid \Gamma \vdash A \to B}\,[\to_R]$$

Fig. 2. The Rules of GLC^* for LC

Proposition 4.1. *Let* $[\![\cdot]\!] : \mathsf{Var} \to \overline{[0,n)}$ *be an interpretation,*
$[\![\cdot]\!]$ *is a countermodel of* $G \mid \Gamma, \bot \to A \vdash B$ *in* LC_n *(resp.* LC*) iff* $[\![\cdot]\!]$ *is countermodel of* $G \mid \Gamma \vdash B$ *in* LC_n *(resp.* LC*);*
$[\![\cdot]\!]$ *is countermodel of* $G \mid \Gamma, A \to A \vdash B$ *in* LC_n *(resp.* LC*) iff* $[\![\cdot]\!]$ *is countermodel of* $G \mid \Gamma \vdash B$ *in* LC_n *(resp.* LC*);*
$[\![\cdot]\!]$ *is countermodel of* $G \mid \Gamma, p \to q \vdash p$ *in* LC_n *(resp.*LC*) iff* $[\![\cdot]\!]$ *is countermodel of* $G \mid \Gamma \vdash p \mid p \to q \vdash p$ *in* LC_n *(resp.* LC*).*

From these results we now consider the GLC^* system the rules of which are given in Figure 2. The axioms of GLC^* are the generalized axioms defined as follows [3]: A generalized axiom is a basic hypersequent of one of the following forms: a) $p_1 \prec p_2 \mid p_2 \prec p_3 \mid \ldots \mid p_{n-1} \prec p_n \mid p_n \vdash p_1$ where $n \geqslant 1$, $p_1, \ldots, p_n$ are n distinct propositional variables, and for all $1 \leq i \leq n-1$, $p_i \prec p_{i+1}$ is either $p_i \vdash p_{i+1}$ or $(p_{i+1} \to p_i) \vdash p_{i+1}$; b) $(p_1 \to \bot) \vdash p_1 \mid (p_2 \to p_1) \vdash p_2 \mid \ldots \mid (p_{n-1} \to p_{n-2}) \vdash p_{n-1} \mid p_{n-1} \vdash p_n$ where $n \geqslant 1$, $p_1, \ldots, p_n$ are n distinct propositional variables (in the case $n = 1$ we take p_0 to be $\bot$).

Let us recall some useful definitions. Knowing that a proof rule is composed of premises H_i with a conclusion C, it is *strongly sound* if, for any instance of the rule and any interpretation $[\![\cdot]\!]$, if $[\![\cdot]\!]$ is a model of all the H_i then it is a model of C. Moreover it is *strongly invertible* if, for any instance of the rule and any interpretation $[\![\cdot]\!]$, if $[\![\cdot]\!]$ is a countermodel of at least one H_i then it is a countermodel of C.

Theorem 4.1. *The rules of* GLC^* *are strongly sound for* LC_n *(resp.* LC*).*

Proof. We consider the $(\to)\to$ rule. The other cases are similar. Let $[\![\cdot]\!]$ be an interpretation which is a model of both premises. Thus, $[\![\cdot]\!]$ is a model of G or both

$\llbracket\Gamma\rrbracket \wedge \llbracket C\rrbracket \leq \llbracket D\rrbracket$ and either $\llbracket A\rrbracket \leq \llbracket B\rrbracket$ or $\llbracket\Gamma\rrbracket \wedge \llbracket B \to C\rrbracket \vdash \llbracket D\rrbracket$ hold:
- if $\llbracket\cdot\rrbracket$ is a model of G then $\llbracket\cdot\rrbracket$ is a model of $G \mid \Gamma, (A \to B) \to C \vdash D$, conclusion of the $(\to)\!\to$ rule;
- if $\llbracket\Gamma\rrbracket \wedge \llbracket C\rrbracket \leq \llbracket D\rrbracket$ and $\llbracket A\rrbracket \leq \llbracket B\rrbracket$. Since $\llbracket A\rrbracket \leq \llbracket B\rrbracket$, we have $\llbracket(A \to B) \to C\rrbracket = \llbracket C\rrbracket$ and we conclude that $\llbracket\Gamma\rrbracket \wedge \llbracket(A \to B) \to C\rrbracket \leq \llbracket D\rrbracket$; - if $\llbracket\Gamma\rrbracket \leq D$ then $\llbracket\Gamma\rrbracket \wedge \llbracket(A \to B) \to C\rrbracket \leq \llbracket D\rrbracket$ holds;
- if $\llbracket C\rrbracket \leq \llbracket D\rrbracket$ and $\llbracket B \to C\rrbracket \leq \llbracket D\rrbracket$ then if $\llbracket A\rrbracket > \llbracket B\rrbracket$ then $\llbracket(A \to B) \to C\rrbracket = \llbracket B \to C\rrbracket$ and $\llbracket\Gamma\rrbracket \wedge \llbracket(A \to B) \to C\rrbracket \leq \llbracket D\rrbracket$ holds. Else, $\llbracket(A \to B) \to C\rrbracket = \llbracket C\rrbracket$ and we deduce that $\llbracket\Gamma\rrbracket \wedge \llbracket(A \to B) \to C\rrbracket \leq \llbracket D\rrbracket$.

Theorem 4.2. *The rules of GLC* are strongly invertible for* LC_n *(resp.* LC*).*

Proof. We consider the $(\to)\!\to$ rule. The other cases are similar. Let $\llbracket\cdot\rrbracket$ be a countermodel of $G \mid \Gamma, C \vdash D$ (the left premise). Then both $\llbracket\cdot\rrbracket$ is a countermodel of G and $\llbracket\Gamma\rrbracket \wedge \llbracket C\rrbracket > \llbracket D\rrbracket$ hold. Since $\llbracket C\rrbracket \leq \llbracket(A \to B) \to C\rrbracket$, we deduce that $\llbracket\cdot\rrbracket$ is a countermodel of G and $\llbracket\Gamma\rrbracket \wedge \llbracket(A \to B) \to C\rrbracket > \llbracket D\rrbracket$. Therefore, $\llbracket\cdot\rrbracket$ is a countermodel of the conclusion of the rule $(\to)\!\to$. Let $\llbracket\cdot\rrbracket$ be a countermodel of $G \mid \Gamma, B \to C \vdash D$ (the right premise). We have $\llbracket\cdot\rrbracket$ is a countermodel of G and both $\llbracket A\rrbracket > \llbracket B\rrbracket$ and $\llbracket\Gamma\rrbracket \wedge \llbracket B \to C\rrbracket > \llbracket D\rrbracket$ hold. Since $\llbracket A\rrbracket > \llbracket B\rrbracket$, we have $\llbracket(A \to B) \to C\rrbracket = \llbracket B \to C\rrbracket$. Thus $\llbracket\Gamma\rrbracket \wedge \llbracket(A \to B) \to C\rrbracket > \llbracket D\rrbracket$ holds and we conclude that $\llbracket\cdot\rrbracket$ is a countermodel of the $(\to)\!\to$ rule conclusion.

Since all *GLC** rules are strongly invertible, we obtain, for any $\mathcal{H} \in \mathcal{B}$, if $\llbracket\cdot\rrbracket : \mathsf{Var} \to \overline{[0,n)}$ is countermodel of $\mathcal{H}$ in LC_n (resp. LC) then $\llbracket\cdot\rrbracket$ is countermodel of G in LC_n (resp. LC) because $\mathcal{B}$ is obtained from the *GLC** rules and Proposition 4.1. Thus we get a decision procedure for hypersequents in LC_n (resp. LC) which builds countermodels, by using the previous decision procedure in order to decide which basic hypersequents are valid in LC_n (resp. LC) and eventually to build a countermodel. Moreover we can characterize the axioms of *GLC** as the basic hypersequents with associated bi-colored graphs that contain a $\Rightarrow$-cycle.
Therefore we have provided new decision procedures for LC but also LC_n with construction of countermodels and decision of irreducible hypersequents that can be realized in linear time. In comparison sequent of relations calculi provide a nice framework for proof search in LC [6,7] but cannot deal with the finitary versions LC_n.

4.2 A new hypersequent calculus and a tableau system for LC_n

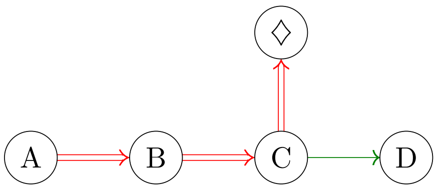
Having defined a new procedure for hypersequents in LC but mainly for LC_n by defining bi-colored graphs associated to hypersequents. In a dual approach we show how we can deduce, from our study of bi-colored grpahs, a new hypersequent calculus for LC_n similar to system *GLC**, by providing a new class of axioms called n-generalized axioms.

Definition 4.1 (n-generalized axiom). *A* n-generalized axiom *is either a generalized axiom or a basic hypersequent of the form:*
$p^1_{m_1} \vdash p^1_{m_1-1} \mid (p^2_1 \to p^2_2) \vdash p^2_1 \mid (p^2_2 \to p^2_3) \vdash p^2_2 \mid \ldots \mid (p^2_{m_2-2} \to p^2_{m_2-1}) \vdash p^2_{m_2-2} \mid p^2_{m_2} \vdash p^2_{m_2-1} \mid (p^3_1 \to p^3_2) \vdash p^3_1 \mid (p^3_2 \to p^3_3) \vdash p^3_2 \mid \ldots \mid (p^3_{m_3-2} \to p^3_{m_3-1}) \vdash p^3_{m_3-2} \mid p^3_{m_3} \vdash p^3_{m_3-1}$

$\ldots$
$\mid (p_1^n \to p_2^n) \vdash p_1^n \mid (p_2^n \to p_3^n) \vdash p_2^n \mid \ldots \mid (p_{m_n-2}^n \to p_{m_n-1}^n) \vdash p_{m_n-2}^n \mid p_{m_n}^n \vdash p_{m_n-1}^n \mid p_{m_n}^n \vdash' p_f$
where for all $1 \leq k \leq n$, $m_k \geqslant 2$ and $p_{m_k}^i = p_1^{i+1}$. Moreover, for all $2 \leq i \leq n$ and $1 \leq j \leq m_i$, $p_j^i, p_{m_1}^1, p_f$ are $2+m_2+\ldots+m_n$ distinct propositional variable, and $p_{m_1-1}^1$ is either a distinct propositional variable or $\bot$ and $p \vdash' q$ is either $q \prec p$ or $q \to p \vdash q$.

From the n-generalized axioms, we can derive all the basic hypersequents the bi-colored graphs of which contain a $(n+1)$-alternating chain, by using (internal and external) weakenings and permutations.

As an example $\mathcal{H} \equiv B \vdash A \mid C \vdash B \mid C \to D \vdash C$ is a 2-generalized axiom with the following bi-colored graph:



Theorem 4.3. *A basic hypersequent is valid in* LC_n *iff it is a basic hypersequent derived from some n-generalized axiom using weakenings and permutations.*

Proof. First we prove the if part. Let $\mathcal{H}$ be a basic hypersequent. We suppose that $\mathcal{H}$ is derived from a n-generalized axiom using weakenings and permutations. Then the bi-colored $\mathcal{G}_{\mathcal{H}}$ contain a $n+1$-alternating chain. By Theorem 3.1, we have $\mathcal{H}$ valid in LC_n. We now prove the only if part. Let $\mathcal{H}$ be a basic hypersequent valid in LC_n. We suppose that $\mathcal{H}$ is not derived from a n-generalized axiom using weakenings and permutations. Then the bi-colored $\mathcal{G}_{\mathcal{H}}$ does not contain a $(n+1)$-alternating chain. By Theorem 3.1, we deduce that $\mathcal{H}$ is not valid in LC_n and then we have a contradiction.

Definition 4.2. *We define the GLC_n^* system as the hypersequent calculus having*
- basic hypersequents derived from the n-generalized axioms using (internal and external) weakenings and permutations, as axioms;
- rules of GLC^* as rules.

The GLC_n^* axioms are the basic hypersequents whose the bi-colored graphs contain $(n+1)$-alternating chains. Therefore, they are the basic hypersequents valid in LC_n. Since for every hypersequent $\mathcal{G}$, one can find a set of basic hypersequents $\mathcal{B}$, so that $\mathcal{G}$ is valid in LC_n if and only if $\mathcal{H}$ is valid in LC_n for every $\mathcal{H} \in \mathcal{B}$, we conclude that a hypersequent $\mathcal{G}$ is valid in LC_n if and only if $\mathcal{G}$ has a proof in GLC_n^*.

Theorem 4.4. *A formula $\mathcal{F}$ is valid in* LC_n *iff the sequent $\vdash \mathcal{F}$ has a proof in GLC_n^*.*

A consequence is that a tableau system for finitary versions of Gödel-Dummett logic $(\mathsf{LC}_n)_{n>0}$ based on the hypersequent calculus GLC_n^* can be obtained from the Avron's tableau system for LC based on GLC^* [3]. We only have to change the definition of closed branchs by using the axioms of GLC_n^* instead of the ones of GLC^*. This direct extension to LC_n is the result of the use of bi-colored graphs to decide the basic hypersequents. In order to check if a branch is closed, it seems simpler to verify the existence of a particular chain or cycle in the graph than to verify if a set of signed formulas (and links) represents or not an instance of an axiom (in GLC_n^* or GLC^*).

5 Bi-colored graphs and hypersequents in LC and LC_n

In this section we consider (general) hypersequents and aim at studying if the approach used in the case of a particular class of hypersequents can be generalized in the general case by defining adequate bi-colored graphs to to characterize provability.
Our approach consists in applying to a given hypersequent $\mathcal{H}$ an indexing process [13] and then to reduce it to a flat sequent $\mathcal{S}$ such that $\mathcal{H}$ is valid if and only if $\mathcal{S}$ is valid. Let us precise that $\mathcal{H}$ cannot include occurrences of special variables $\Box$ and $\Diamond$ but can include occurrences of $\bot$. Such occurrences are eliminated during the flattening process. We remind that a formula is flat if it is implicational, of the form $X \to (Y \star Z)$ or $(X \star Y) \to Z$ with $X,Y,Z \in \mathsf{Var}$ and $\star \in \{\wedge,\vee,\to\}$. A $\Diamond$-context $\Delta_\Diamond$ is a non-empty multiset of implicational formulae such that if $A \to B \in \Delta_\Diamond$ then $\Diamond \to B \in \Delta_\Diamond$. Moreover $\Gamma \vdash \Delta_\Diamond$ is a flat sequent if the context Γ contains only flat formulae and $\Delta_\Diamond$ is a $\Diamond$-context. A flat hypersequent is such that all its components are flat.
The indexing process is based on the two linear functions δ^+ and δ^-, that map occurrences of subformulae of a given formula D to multisets. They are defined as follows:

$$\begin{array}{ll}
\delta^+(\bot) & = \mathcal{X}_\bot \to \Box \\
\delta^+(V) & = \mathcal{X}_V \to V, \Box \to V \text{ with } V \text{ is a variable} \\
\delta^+(A * B) & = \delta^+(A), \delta^+(B), \mathcal{X}_{A*B} \to (\mathcal{X}_A * \mathcal{X}_B) \text{ with } * \in \{\wedge,\vee\} \\
\delta^+(A \to B) & = \delta^-(A), \delta^+(B), \mathcal{X}_{A\to B} \to (\mathcal{X}_A \to \mathcal{X}_B)
\end{array}$$

$$\begin{array}{ll}
\delta^-(\bot) & = \Box \to \mathcal{X}_\bot \\
\delta^-(V) & = V \to \mathcal{X}_V, \Box \to V \text{ with } V \text{ is a variable} \\
\delta^-(A * B) & = \delta^-(A), \delta^-(B), (\mathcal{X}_A * \mathcal{X}_B) \to \mathcal{X}_{A*B} \text{ with } * \in \{\wedge,\vee\} \\
\delta^-(A \to B) & = \delta^+(A), \delta^-(B), (\mathcal{X}_A \to \mathcal{X}_B) \to \mathcal{X}_{A\to B}
\end{array}$$

The size of a formula is the number of occurrences of its subformulae that is the number of nodes in its decomposition tree. Let D be a formula of size n, it has been proved that the cardinals of $\delta^+(D)$ and $\delta^-(D)$ are smaller than $2n$. Moreover the elements of these multisets are only flat formulae of size less than 5. Then the size of $\delta^-(D)$ and $\delta^+(D)$ are bounded by $5n$.

Proposition 5.1. *Let D be a formula, if $[\![\cdot]\!]$ is an interpretation such that $[\![\Box]\!] = 0$ and $[\![\mathcal{X}_K]\!] = [\![K]\!]$ for any occurrence of subformula K of D then $\delta^+(D) = \delta^-(D) = \infty$.*

This proposition has been proved in [15]. The next step consists, using this indexing process, in transforming a given hypersequent $\mathcal{H}$ into an indexed flat sequent $\mathcal{S}$ and in building a bi-colored graph from this sequent. Before to give this construction we study the preservation of validity and countermodels through such a transformation.

5.1 Hypersequents and flat sequents

For the presentation we consider mono-conclusioned hypersequents but we finally show how and why results are valid for (multi-conclusioned) hypersequents.
Let $\mathcal{H} = A^1_1,\ldots,A^1_{n_1} \vdash B_1 \mid \ldots \mid A^p_1,\ldots,A^p_{n_p} \vdash B_p$ be a hypersequent of LC. We associate to $\mathcal{H}$ a particular flat sequent $\mathcal{S} = FS(\mathcal{H}) = \delta^+(A^1_1),\ldots,\delta^+(A^p_{n_p}),\ \delta^-(B_1),\ldots,\delta^-(B_p)$

$\vdash \mathcal{X}_{A^1_1} \to \mathcal{X}_{B_1}, \ldots, \mathcal{X}_{A^1_{n_1}} \to \mathcal{X}_{B_1}, \mathcal{X}_{A^2_1} \to \mathcal{X}_{B_2}, \ldots, \mathcal{X}_{A^2_{n_2}} \to \mathcal{X}_{B_2}, \ldots, \mathcal{X}_{A^p_1} \to \mathcal{X}_{B_p}, \ldots, \mathcal{X}_{A^p_{np}} \to \mathcal{X}_{B_p}, \Diamond \to \mathcal{X}_{B_1}, \ldots, \Diamond \to \mathcal{X}_{B_p}$.

Theorem 5.1. *Let $\mathcal{H}$=$A^1_1,\ldots,A^1_{n_1} \vdash B_1 \mid \ldots \mid A^p_1,\ldots,A^p_{n_p} \vdash B_p$. If the sequent $FS(\mathcal{H})$ is valid in LC_n then the hypersequent $\mathcal{H}$ is valid in LC_n.*

Proof. Let $[\![\cdot]\!] : \mathsf{Var} \to \overline{[0,n)}$ be an interpretation. We define a new interpretation $[\![\cdot]\!]'$ by $[\![V]\!]' = [\![V]\!]$ for any variable V of $\mathcal{H}$, $[\![\mathcal{X}_K]\!]' = [\![K]\!]$ for any K subformula of $\mathcal{H}$ formulae, $[\![\Diamond]\!]' = \infty$ and $[\![\Box]\!]' = 0$. As $[\![\cdot]\!]'$ and $[\![\cdot]\!]$ have the same values for $\mathcal{H}$'s atoms , for any subformula K of $\mathcal{H}$ formulae, $[\![K]\!]' = [\![K]\!]$. Therefore $[\![\mathcal{X}_K]\!]' = [\![K]\!]'$ and $[\![\Box]\!]' = 0$. By Proposition 5.1, we obtain $\forall i \in [1,p]\ \forall j \in [1,n_i]\ [\![\delta^+(A^i_j)]\!] = [\![\delta^-(B_i)]\!] = \infty$. As $\mathcal{S}$ is valid in LC_n, $[\![\cdot]\!]'$ is a model of $\mathcal{S}$ and consequently $\exists i \in [1,p]\ \exists j \in [1,n_i]\ [\![\mathcal{X}_{A^i_j} \to \mathcal{X}_{B_i}]\!]' = \infty$ or $[\![\Diamond \to \mathcal{X}_{B_i}]\!]' = \infty$. But $[\![\Diamond]\!] = \infty$ and then $\exists i \in [1,p]\ \exists j \in [1,n_i]\ [\![\mathcal{X}_{A^i_j}]\!]' \leqslant [\![\mathcal{X}_{B_i}]\!]'$ or $[\![\mathcal{X}_{B_i}]\!]' = \infty$. Thus $\exists i \in [1,p]\ \exists j \in [1,n_i]\ [\![A^i_j]\!] \leqslant [\![B_i]\!]$ or $[\![B_i]\!] = \infty$. As we have proved that $\exists i \in [1,p]\ \exists j \in [1,n_i]\ [\![A^i_j]\!] \leqslant [\![B_i]\!]$ or $[\![B_i]\!] = \infty$, for any interpretation $[\![\cdot]\!] : \mathsf{Var} \to \overline{[0,n)}$, we deduce that $\mathcal{H}$ is valid in LC_n.

Let $[\![\cdot]\!] : \mathsf{Var} \to \overline{[0,n)}$ be an interpretation and $\alpha \in \overline{[0,n)}$, we define the translated interpretation by:

$$[\![X]\!]_{-\alpha} = \begin{cases} \infty & \text{if } [\![X]\!] = \infty \\ [\![X]\!] - \alpha & \text{if } [\![X]\!] \geq \alpha \\ 0 & \text{if } [\![X]\!] < \alpha \end{cases}$$

Theorem 5.2. *Let $\mathcal{H}$=$A^1_1,\ldots,A^1_{n_1} \vdash B_1 \mid \ldots \mid A^p_1,\ldots,A^p_{n_p} \vdash B_p$, if $[\![\cdot]\!] : \mathsf{Var} \to \overline{[0,n)}$ is a countermodel of the sequent $FS(\mathcal{H})$ in LC_n then $[\![\Box]\!] < \infty$ and for $\alpha = [\![\Box]\!]$, the translated interpretation $[\![\cdot]\!]_{-\alpha}$ is a countermodel of the hypersequent $\mathcal{H}$ in LC_n.*

Proof. Given in appendix A.

5.2 Bi-colored graphs

Let $\mathcal{H}$ be a given hypersequent of LC. As shown in the previous section we can associate to $\mathcal{H}$ an equivalent flat sequent $FS(\mathcal{H}) = \mathcal{S} = \delta^+(A^1_1), \ldots, \delta^+(A^p_{n_p}), \delta^-(B_1), \ldots, \delta^-(B_p) \vdash \mathcal{X}_{A^1_1} \to \mathcal{X}_{B_1}, \ldots, \mathcal{X}_{A^1_{n_1}} \to \mathcal{X}_{B_1}, \mathcal{X}_{A^2_1} \to \mathcal{X}_{B_2}, \ldots, \mathcal{X}_{A^2_{n_2}} \to \mathcal{X}_{B_2}, \ldots, \mathcal{X}_{A^p_1} \to \mathcal{X}_{B_p}, \ldots, \mathcal{X}_{A^p_{np}} \to \mathcal{X}_{B_p}$, $\Diamond \to \mathcal{X}_{B_1}, \ldots, \Diamond \to \mathcal{X}_{B_p}$. Now we define a procedure that builds, from $\mathcal{H}$ and the flat sequent $FS(\mathcal{H})$, a particular bi-colored graph $\mathcal{G}_{\mathcal{H}}$ that is associated to $\mathcal{H}$.

The **nodes** of $\mathcal{G}_{\mathcal{H}}$ are defined from the set of the nodes of the decomposition tree of all the formulae of $\mathcal{H}$ (set of the subformulae occurrences).
Moreover we introduce a new variable $\mathcal{X}_F$ for every occurrence F of subformula of D in $\mathcal{H}$. It is the corresponding node of F. The nodes are signed as follows: we have $-$ at the root D^- if D is a hypothesis else we have $+$ and we propagate the signs as usual.[1]

[1] The connectors $\wedge$ and $\vee$ preserve the signs and $\to$ preserves the sign on the righthand side and inverses the sign on the lefthand side.

We can write $\mathcal{X}_F^+$ or $\mathcal{X}_F^-$ in order to emphasize the signs.
We also add the node denoted V for all propositional variables of $\mathcal{H}$. Thus several occurrences of V generate only one node V and several nodes $\mathcal{X}_V^+$ or $\mathcal{X}_V^-$. Moreover we add two new nodes denoted $\diamondsuit$ et $\square$.

The **arrows** of $\mathcal{G}_{\mathcal{H}}$ are defined as follows: we describe the green and red arrows between the nodes together with the boolean expressions indexing them.
First we start with the non-indexed arrows introduced independently of the internal structure of $\mathcal{H}$'s formulae. We add
- a red arrow $\mathcal{X}_D^- \Rightarrow \diamondsuit$ for any formula D of the multi-set of conclusions of $\mathcal{H}$.
- a red arrow $\mathcal{X}_B^- \rightarrow \mathcal{X}_A^+$ for any formula A and B of $\mathcal{H}$ that belong to the same component.
- a green arrow $V \rightarrow \mathcal{X}_V^-$ for any negative occurrence of variable V and a green arrow $\mathcal{X}_V^+ \rightarrow V$ for any positive occurrence of variable V.
- a green arrow $\square \rightarrow V$ for any variable V, a green arrow $\square \rightarrow \mathcal{X}_\perp^-$ for any negative occurrence of $\perp$ and a green arrow $\mathcal{X}_\perp^+ \rightarrow \square$ for any positive occurrence of $\perp$.
Secondly we consider the introduction of arrows for the internal nodes. Let us start with the non-indexed arrows. We add
- two green arrows $\mathcal{X}_C^+ \rightarrow \mathcal{X}_A^+$ and $\mathcal{X}_C^+ \rightarrow \mathcal{X}_B^+$ for any positive subformula occurrence $C \equiv A \wedge B$.
- two green arrows $\mathcal{X}_A^- \rightarrow \mathcal{X}_C^-$ and $\mathcal{X}_B^- \rightarrow \mathcal{X}_C^-$ for any negative subformula occurrence $C \equiv A \vee B$.
We now complete with the indexed arrows, i.e., arrows indexed by boolean expressions of the form x or $\overline{x}$ with x propositional variable. Then we introduce a new boolean variable for any subformula occurrence. We add
- a new boolean variable x and two green arrows $\mathcal{X}_A^- \rightarrow_x \mathcal{X}_C^-$ and $\mathcal{X}_B^- \rightarrow_{\overline{x}} \mathcal{X}_C^-$ for any negative subformula occurrence $C \equiv A \wedge B$.
- a new boolean variable x and two green arrows $\mathcal{X}_C^+ \rightarrow_x \mathcal{X}_A^+$ and $\mathcal{X}_C^+ \rightarrow_{\overline{x}} \mathcal{X}_B^+$ for any positive subformula occurrence $C \equiv A \vee B$.
- a new boolean variable x, two green arrows $\mathcal{X}_B^- \rightarrow_x \mathcal{X}_C^-$ and $\diamondsuit \rightarrow_{\overline{x}} \mathcal{X}_C^-$ and two red arrows $\mathcal{X}_B^- \Rightarrow_x \mathcal{X}_A^+$ and $\mathcal{X}_B^- \Rightarrow_x \diamondsuit$ for any negative subformula occurrence $C \equiv A \rightarrow B$.
- a new boolean variable x and two green arrows $\mathcal{X}_C^+ \rightarrow_x \mathcal{X}_B^+$ and $\mathcal{X}_A^- \rightarrow_{\overline{x}} \mathcal{X}_B^+$ for any positive subformula occurrence $C \equiv A \rightarrow B$.

We illustrate this construction with the hypersequent $\mathcal{H}_3 = A \vdash B \mid A \rightarrow B \vdash B$. The bi-colored graph $\mathcal{G}_{\mathcal{H}_3}$ associated to $\mathcal{H}_3$ is given in Figure 3.
It is clear that the construction of the graph $\mathcal{G}_{\mathcal{H}}$ for a given hypersequent $\mathcal{H}$ is in linear time because we add at most four arrows for each subformula instance. Now we have to define a characterization of the validity of $\mathcal{H}$ from the associated bi-colored graph.

5.3 A procedure for countermodel search

We have proved, in the previous section, that deciding if a hypersequent $\mathcal{H}$ is valid or has a countermodel in LC_n can be reduced to deciding if the flat sequent associated to $\mathcal{H}$ (see Theorem 5.1 and 5.2) is valid or has a countermodel in LC_n. Then we focus

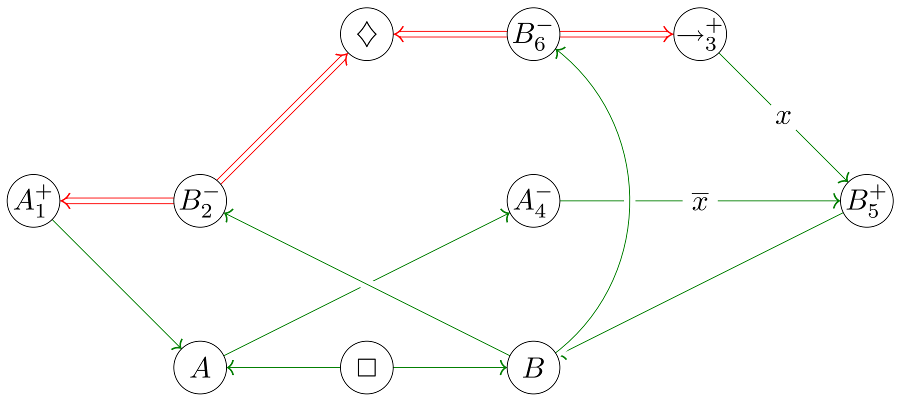


Fig. 3. A Bi-colored Graph of a Hypersequent

now on the graph $\mathcal{G}_{\mathcal{H}}$ and analyze validity of $\mathcal{H}$ from it.
First any flat sequent can be reduced or transformed into a set of implicational sequents, i.e., sequents of the form $\mathcal{S} = X_1 \to Y_1, \ldots, X_k \to Y_k \vdash A_1 \to B_1, \ldots, A_l \to B_l$. Secondly we can associate a bi-colored graph $\mathcal{G}_{\mathcal{S}}$ to such a sequent $\mathcal{S}$ as follows: the set of nodes is the set of the variables of $\mathcal{S}$, namely $\{X_i\} \cup \{Y_i\} \cup \{A_i\} \cup \{B_i\}$ and the set of arrows is $\{X_1 \to Y_1, \ldots, X_k \to Y_k\} \cup \{B_1 \Rightarrow A_1, \ldots, B_l \Rightarrow A_l\}$. Then an implicational sequent $\mathcal{S}$ has a countermodel in LC_n (resp. in LC) if and only if its associated bi-colored graph does not contain a $(n+1)$-alternating chain (resp. a $\Rightarrow$-cycle) [16]. Thus we now study if we can relate the search of particular chains in an instance of the associated graph $\mathcal{G}_{\mathcal{H}}$ to the existence of countermodels.

Theorem 5.3 **($n = \infty$).** *Let $\mathcal{H}$ be a hypersequent and $\mathcal{G}_{\mathcal{H}}$ its bi-colored graph, $\mathcal{H}$ has a countermodel in* LC *if and only if there exists an instance of $\mathcal{G}_{\mathcal{H}}$ that does not contain a $\Rightarrow$-cycle.*

Proof. Let $\mathcal{H}$ be a hypersequent and $\mathcal{S}$ be the associated flat sequent. An interpretation $[\![\cdot]\!]$ is a countermodel of $\mathcal{S}$ if and only if at least it is a countermodel of one of the implicational sequents issued of the transformation of $\mathcal{S}$. The bi-colored graphs associated to these implicational sequents exactly correspond to the instances of $\mathcal{G}_{\mathcal{H}}$. By the above-mentioned results $\mathcal{S}$ has a countermodel if and only if one of the instances of $\mathcal{G}_{\mathcal{H}}$ does not contain a $\Rightarrow$-cycle. Thus, $\mathcal{H}$ has a countermodel if and only if one of the instances of $\mathcal{G}_{\mathcal{H}}$ does not contain a $\Rightarrow$-cycle.

Theorem 5.4 **($n < \infty$).** *Let $\mathcal{H}$ be a hypersequent and $\mathcal{G}_{\mathcal{H}}$ its associated bi-colored graph, $\mathcal{H}$ has a countermodel in* LC_n *if and only if there exists an instance of $\mathcal{G}_{\mathcal{H}}$ that does not contain a $(n+1)$-alternating chain.*

Proof. For LC_n with $n \neq \infty$, the proof is similar by replacing the notion of $\Rightarrow$-cycle by the one of $(n+1)$-alternating chain.

Let us come back to our example with the hypersequent $\mathcal{H}_3 = A \vdash B \mid A \to B \vdash B$. If we consider its associated graph $\mathcal{G}_{\mathcal{H}_3}$ (see previous subsection) we observe that it has two instances ($x = 0$ and $x = 1$). The first one ($x = 0$) contains the following $\Rightarrow$-cycle: $B_2^- \Rightarrow A_1^+ \rightarrow A \rightarrow A_4^- \rightarrow B_5^+ \rightarrow B \rightarrow B_2^-$. The second one ($x = 1$) contains the following $\Rightarrow$-cycle: $B_6^- \Rightarrow \rightarrow_3^+ \rightarrow B_5^+ \rightarrow B \rightarrow B_6^-$. Then we deduce that $\mathcal{H}_3$ does not contain countermodels in LC. An example with countermodel generation is given in appendix B.

These results on mono-conclusioned hypersequents can be easily extended to (multi-conclusioned) hypersequents.
Let $\mathcal{H}$=$A_1^1,\ldots,A_{n_1}^1 \vdash B_1^1,\ldots,B_{m_1}^1 \mid \ldots \mid A_1^p,\ldots,A_{n_p}^p \vdash B_1^p,\ldots,B_{m_p}^p$ be a given hypersequent, we build the flat sequent $FS(\mathcal{H}) = \delta^+(A_1^1),\ldots,\delta^+(A_{n_p}^p),\delta^-(B_1^1),\ldots,\delta^-(B_{m_p}^p)$, $(\mathcal{X}_{\vdash_1}) \rightarrow \mathcal{X}_{A_1^1},\ldots,(\mathcal{X}_{\vdash_1}) \rightarrow \mathcal{X}_{A_{n_1}^1},\ldots,(\mathcal{X}_{\vdash_p}) \rightarrow \mathcal{X}_{A_{n_p}^p} \vdash (\mathcal{X}_{\vdash_1}) \rightarrow \mathcal{X}_{B_1^1},\ldots,(\mathcal{X}_{\vdash_1}) \rightarrow \mathcal{X}_{B_{m_1}^1},\ldots,$ $(\mathcal{X}_{\vdash_p}) \rightarrow \mathcal{X}_{B_1^p},\ldots,(\mathcal{X}_{\vdash_p}) \rightarrow \mathcal{X}_{B_{m_p}^p},\Diamond \rightarrow \mathcal{X}_{B_1^1},\ldots,\Diamond \rightarrow \mathcal{X}_{B_{m_p}^p}$. Then we can prove theorems similar to Theorem 5.1 and Theorem 5.2 and then directly use the procedure defined for bi-colored graph construction. In addition our new procedure can be applied to a specific case of hypersequents that are sequents. Therefore we also provide a new procedure for deciding provability of LC sequents through bi-colored graphs with generation of countermodels.

6 Conclusion and perspectives

In this paper we propose new characterizations of validity in LC and LC_n based on the construction, from a hypersequent, of a specific bi-colored graph on which the search of particular chains corresponds to countermodel search. It leads to new decision procedures for hypersequents in Gödel-Dummett logics that is well adapted to countermodel generation. These results present an alternative approach to works on proof-search with analytic calculi. Thus we aim at developing it for other logics including substructural or intermediate logics [18].
Recent works have studied the relationships between parallel dialogue games and hypersequents for some intermediate logics including LC [10]. We also aim at relating bi-colored graphs and such games in such a way that we could generate directly winning strategies from bi-colored graphs associated to sequents or hypersequents. From preliminary results for graphs associated to implicational sequents we expect to study how to deal with general sequents or hypersequents.

References

1. A. Avellone, M. Ferrari, and P. Miglioli. Duplication-free tableau calculi and related cut-free sequent calculi for the interpolable propositional intermediate logics. *Logic Journal of the IGPL*, 7(4):447–480, 1999.
2. A. Avron. Hypersequents, logical consequence and intermediate logics for concurrency. *Annals of Mathematics and Artifi cial Intelligence*, 4:225–248, 1991.
3. A. Avron. A Tableau System for Gödel-Dummett Logic based on a Hypersequent Calculus. In *Int. Conference on Analytic Tableaux and Related Methods, TABLEAUX 2000, LNAI 1847*, pages 98–111, St Andrews, Scotland, 2000.
4. A. Avron and B. Konikowska. Decomposition Proof Systems for Gödel-Dummett Logics. *Studia Logica*, 69(2):197–219, 2001.

5. M. Baaz, A. Ciabattoni, and C. Fermüller. Hypersequent calculi for Gödel logics - a survey. *Journal of Logic and Computation*, 13(6):835–861, 2004.
6. M. Baaz and C. Fermüller. Analytic calculi for projective logics. In *Int. Conference on Analytic Tableaux and Related Methods, TABLEAUX'99, LNAI 1617*, pages 36–51, Saratoga Spring, USA, 1999.
7. A. Ciabattoni, C. Fermüller, and G. Metcalfe. Uniform rules and dialogue games for fuzzy logics. In *Int. Conference on Logic for Programming, Artifi cial Intelligence, and Reasoning, LPAR 2004, LNAI 3452*, pages 496–510, Montevideo, Uruguay, 2004.
8. M. Dummet. A propositional calculus with a denumerable matrix. *JSL*, 24:96–107, 1959.
9. R. Dyckhoff. A deterministic terminating sequent calculus for Gödel-Dummett logic. *Logic Journal of the IGPL*, 7(3):319–326, 1999.
10. C. Fermüller. Parallel dialogue games and hypersequents for intermediate logics. In *Int. Conference on Analytic Tableaux and Related Methods, TABLEAUX 2003, LNAI 2796*, pages 48–64, Rome, Italy, 2003.
11. D. Galmiche and D. Méry. Resource graphs and countermodels in resource logics. *Electronic Notes in Theoretical Computer Science*, 125(3):117–135, 2005.
12. P. Hájek. *Metamathematics of Fuzzy Logic*. Kluwer Academic Publishers, 1998.
13. D. Larchey-Wendling. Counter-model search in Gödel-Dummett logics. In *2nd Int. Joint Conference IJCAR 2004, LNAI 3097*, pages 274–288, Cork, Ireland, July 2004.
14. D. Larchey-Wendling. Bounding resource consumption with Gödel-Dummett logics. In *Int. Conference on Logic for Programming, Artifi cial Intelligence, and Reasoning, LPAR 2005, LNAI 3835*, pages 682–696, December 2005.
15. D. Larchey-Wendling. Gödel-Dummett counter-models through matrix computations. *Electronic Notes in Theoretical Computer Science*, 125(3):137–148, 2005.
16. D. Larchey-Wendling. Graph-based decision for Gödel-Dummett logics. *Journal of Automated Reasoning*, 38:201–225, 2007.
17. G. Metcalfe, N. Olivetti, and D. Gabbay. Goal-directed calculi for Gödel-Dummett logics. In *17th Int. Workshop on Computer Science Logic, CSL 2003, LNCS 2803*, pages 413–426, September 2003. Vienna, Austria.
18. G. Metcalfe, N. Olivetti, and D. Gabbay. Sequent and hypersequent calculi for Abelian and Lukasiewicz logics. *ACM Transactions on Computational Logic*, 6(3), 2005.

A Proof of Theorem 5.2

To prove the result we need the three following propositions, proved in [16].

Proposition A.1. *Let A be a formula without $\bot$, α be a value in $\overline{[0,n)}$ and $[\![\cdot]\!] : \mathsf{Var} \to \overline{[0,n)}$ be an interpretation such that, for any variable X of A, $[\![X]\!] \geqslant \alpha$. We have $[\![A]\!] \geqslant \alpha$ and if $\alpha < \infty$ then $[\![A]\!]_{-\alpha} = [\![A]\!] - \alpha$.*

Proposition A.2. *Let D be a formula, we have the following properties:*
1. For any subformula K of D, the formulae of $\delta^+(K)$ and $\delta^-(K)$ are flat and do not contain the constant $\bot$;
2. For any variable V of D, the atomic implication $\square \to V$ is in $\delta^+(K)$ and in $\delta^-(K)$;
3. The size of $\delta^+(K)$ and $\delta^-(K)$ is linear in the size of D.

Proposition A.3. *Let D be a formula, for any subformula K of D, the two sequents $\delta^+(K), \mathcal{X}_K \vdash K_\square$ and $\delta^-(K), K_\square \vdash \mathcal{X}_K$ are valid in LC_n.*

Theorem 5.2
Let $\mathcal{H}=A^1_1,\dots,A^1_{n_1}\vdash B_1 \mid \dots \mid A^p_1,\dots,A^p_{n_p}\vdash B_p$, if $[\![\cdot]\!] : \mathsf{Var}\to\overline{[0,n)}$ is a countermodel of the sequent $FS(\mathcal{H})$ in LC_n then $[\![\Box]\!] < \infty$ and for $\alpha = [\![\Box]\!]$, the translated interpretation $[\![\cdot]\!]_{-\alpha}$ is a countermodel of the hypersequent $\mathcal{H}$ in LC_n.

Proof. Let $[\![\cdot]\!] : \mathsf{Var}\to\overline{[0,n)}$ be a countermodel of the sequent $\mathcal{S}$. The following properties are satisfied:

1. $\forall i,j\in[1,p]$ and $\forall k\in[1,n_i]$, $\llbracket\delta^+(A^i_k)\rrbracket > [\![\Diamond\to\mathcal{X}_{B_j}]\!]$;
2. $\forall i,j\in[1,p]$, $\forall k\in[1,n_i]$ and $\forall l\in[1,n_j]$, $\llbracket\delta^+(A^i_k)\rrbracket > [\![\mathcal{X}_{A^j_l}\to\mathcal{X}_{B_j}]\!]$;
3. $\forall i,j\in[1,p]$, $\llbracket\delta^-(B_i)\rrbracket > [\![\Diamond\to\mathcal{X}_{B_j}]\!]$;
4. $\forall i,j\in[1,p]$ and $\forall k\in[1,n_j]$, $\llbracket\delta^-(B_i)\rrbracket > [\![\mathcal{X}_{A^j_k}\to\mathcal{X}_{B_j}]\!]$.

By property 4, we have $\forall i\in[1,p]$, $\llbracket\delta^-(B_i)\rrbracket > [\![\Diamond\to\mathcal{X}_{B_i}]\!]$ and we deduce that for any $i\in[1,p]$, $[\![\Diamond\to\mathcal{X}_{B_i}]\!]<\infty$ and $[\![\mathcal{X}_{B_i}]\!] = [\![\Diamond]\!]\twoheadrightarrow[\![\mathcal{X}_{B_i}]\!]<\infty$. Then, for any $i\in[1,p]$, $[\![\mathcal{X}_{B_i}]\!] < \llbracket\delta^-(B_i)\rrbracket$. By Proposition A.3, $\forall i\in[1,p]$, $\delta^-(B_i),(B_i)_\Box\vdash\mathcal{X}_{B_i}$ is a valid sequent and then $\forall i\in[1,p]$, $\llbracket\delta^-(B_i)\rrbracket\wedge[\![(B_i)_\Box]\!]\leqslant[\![\mathcal{X}_{B_i}]\!]<\llbracket\delta^-(B_i)\rrbracket$. Therefore $\forall i\in[1,p]$, $\llbracket\delta^-(B_i)\rrbracket\wedge[\![(B_i)_\Box]\!]<\llbracket\delta^-(B_i)\rrbracket$ and we obtain $[\![(B_i)_\Box]\!]<\llbracket\delta^-(B_i)\rrbracket$ for any $i\in[1,p]$.
- We now prove that, for any variable V of $\mathcal{H}$, $[\![V]\!]\geqslant[\![\Box]\!]$. First, if $\mathcal{H}$ does not contain variables, i.e., all its atoms are occurrences of $\perp$, then the previous property is trivially verified. Else, let B_i be one of its conclusion formulae and V_0 be a variable of $(B_i)_\Box$ that realizes the minimal value γ of the non-empty set $\{[\![\Box\to V]\!],\ V$ variable of $B_{i\Box}\}$. Thus $\gamma=[\![\Box\to V_0]\!]$ and for any variable V of $(B_i)_\Box$, $[\![\Box\to V]\!]>\gamma$. If V_0 is in the variable set of B_i then, by Proposition A.2, $\Box\to V_0\in\delta^-(B_i)$ and then $\llbracket\delta^-(B_i)\rrbracket\leqslant[\![\Box\to V_0]\!]=\gamma$. Else $V_0=\Box$ and therefore $\llbracket\delta^-(B_i)\rrbracket\leqslant[\![\Box\to V_0]\!]=[\![\Box\to\Box]\!]=\infty$. In both cases $\llbracket\delta^-(B_i)\rrbracket\leqslant[\![\Box\to V_0]\!]=\gamma$.
- We now prove that $[\![\Box]\!]\leqslant\gamma$. Let us suppose $[\![\Box]\!]>\gamma=[\![\Box\to V_0]\!]$ and let V be a variable de $(B_i)_\Box$: either $V=\Box$ and $[\![V]\!]=[\![\Box]\!]>\gamma$ or V is a variable of B_i and $[\![\Box\to V]\!]\geqslant\gamma$, and then $[\![V]\!]\geqslant\gamma$. By Proposition A.1, as $(B_i)_\Box$ does not contain $\perp$, $[\![(B_i)_\Box]\!]\geqslant\gamma$. We deduce $\gamma\leqslant[\![(B_i)_\Box]\!]<\llbracket\delta^-(B_i)\rrbracket\leqslant[\![\Box\to V_0]\!]=\gamma$, that is contradictory and then $\Box\leqslant\gamma$. For any variable V of $(B_i)_\Box$, $[\![\Box]\!]\leqslant\gamma\leqslant[\![\Box\to V]\!]=[\![\Box]\!]\twoheadrightarrow[\![V]\!]$. Then , for any variable V of $(B_i)_\Box$, $[\![\Box]\!]\leqslant[\![V]\!]$ and by Proposition A.1 $[\![(B_i)_\Box]\!]\geqslant[\![\Box]\!]$. Thus $\forall i\in[1,p]$ $[\![(B_i)_\Box]\!]\geqslant[\![\Box]\!]$ because we have no hypothesis on B_i.
- We now prove that, for any variable V of $\mathcal{H}$, $[\![V]\!]\geqslant[\![\Box]\!]$. We suppose that there exists a variable V_1 such that $[\![V_1]\!]<[\![\Box]\!]$. By Proposition A.2, $\Box\to V_1$ belongs to the multiset of hypotheses of $\mathcal{S}$. Then $\forall i\in[1,p]$ $[\![\Box\to V_1]\!]=[\![V_1]\!]>\mathcal{X}_{B_i}$. We have $\forall i\in[1,p]$ $\llbracket\delta^-(B_i)\rrbracket>[\![\mathcal{X}_{B_i}]\!]$ and by Proposition A.3, $\delta^-(B_i),(B_i)_\Box\vdash\mathcal{X}_{B_i}$ is a valid sequent and then $\forall i\in[1,p]$ $[\![(B_i)_\Box]\!]\leqslant[\![\mathcal{X}_{B_i}]\!]$. Thus we have $\forall i\in[1,p]$ $[\![\Box]\!]\leqslant[\![(B_i)_\Box]\!]\leqslant[\![\mathcal{X}_{B_i}]\!]<[\![V_1]\!]$ that is contradictory.
By property 3, $\forall i,j\in[1,p]$ $\forall k\in[1,n_i]$ $\forall l\in[1,n_j]$ $\llbracket\delta^+(A^i_k)\rrbracket>[\![\mathcal{X}_{A^j_l}\to\mathcal{X}_{B_j}]\!]$ and we deduce that $\forall j\in[1,p]$ $\forall l\in[1,n_j]$ $[\![\mathcal{X}_{A^j_l}\to\mathcal{X}_{B_j}]\!]<\infty$. Then $\forall i\in[1,p]$ $\forall j\in[1,n_i]$ $[\![\mathcal{X}_{A^i_j}]\!]>[\![\mathcal{X}_{B_i}]\!]$ and finally $\forall i\in[1,p]$ $\forall j\in[1,n_i]$ $\llbracket\delta^+(A^i_j)\rrbracket>[\![\mathcal{X}_{B_i}]\!]$ and then $\llbracket\delta^+(A^i_j),\mathcal{X}_{A^i_j}\rrbracket>[\![\mathcal{X}_{B_i}]\!]$. By Proposition A.3, $\forall i\in[1,p]$ $\forall j\in[1,n_i]$ $\delta^+(A^i_j),(A^i_j)_\Box\vdash\mathcal{X}_{A^i_j}$ is a valid sequent. Thus $\forall i\in[1,p]$ $\forall j\in[1,n_i]$ $[\![(A^i_j)_\Box]\!]>[\![\mathcal{X}_{B_i}]\!]$. Moreover $\forall i\in[1,p]$ $\forall j\in[1,n_i]$

$[\![(A^i_j)_\square]\!] > [\![(B_i)_\square]\!]$ because $\forall k \in [1,p]$, $[\![B_{k\square}]\!] \leqslant [\![\mathcal{X}_{B_k}]\!]$.
We have proved that $\forall i \in [1,p]$ $[\![(B_i)_\square]\!] < [\![\delta^-(B_i)]\!]$. But $\forall i \in [1,p]$ $[\![(B_i)_\square]\!] \geqslant [\![\square]\!]$ and then $[\![\square]\!] < \infty$. Let $\alpha = [\![\square]\!]$, as $[\![\square]\!]_{-\alpha} = [\![\square]\!] - \alpha = 0 = [\![\bot]\!]_{-\alpha}$. We obtain $[\![D]\!]_{-\alpha} = [\![D_\square]\!]_{-\alpha}$, for any formula D of $\mathcal{H}$, and by Proposition A.1, $[\![D_\square]\!]_{-\alpha} = [\![D_\square]\!] - \alpha$. We have $\forall i \in [1,p]$ $\forall j \in [1,n_i]$ $[\![A^i_{j\square}]\!] > [\![(B_i)_\square]\!]$ and $\alpha < \infty$. Thus $\forall i \in [1,p]$ $\forall j \in [1,n_i]$ $[\![(A^i_j)_\square]\!] - \alpha > [\![(B_i)_\square]\!] - \alpha$ and then $[\![A^i_j]\!]_{-\alpha} > [\![B_i]\!]_{-\alpha}$. Then $[\![\cdot]\!]_{-\alpha}$ is a countermodel.

B An example with countermodel generation

The bi-colored graph of the hypersequent $\mathcal{H}_4 \equiv \vdash A \mid A \vdash \bot$ is

$$\begin{tikzcd}
A_1^- \arrow[rr, Rightarrow] & & \Diamond \\
A \arrow[u] & A_2^+ \arrow[l] & \bot_3^- \arrow[l, Rightarrow] \arrow[u, Rightarrow] \\
& \square \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

This graph has only one instance that does not contain $\Rightarrow$-cycle. Then we deduce that $\mathcal{H}_4$ has a countermodel. In order to extract it we modify the graph as follows: the red arrows always go up and the green arrows never go down.

$$\begin{tikzcd}
& & \Diamond & \infty \\
A_1^- & A \arrow[l] & A_2^+ \arrow[l] \arrow[u, Rightarrow] & 1 \\
& \square \arrow[u] \arrow[r] & \bot_3^- \arrow[u, Rightarrow] \arrow[uu, Rightarrow, bend right] & 0
\end{tikzcd}$$

Then $[\![\cdot]\!] : \mathsf{Var} \to \overline{[0,n)}$ such that $[\![A]\!] = 1$ and $n > 2$, is a countermodel of $\mathcal{H}_4$ in LC_n.

Redundancy for Geometric Resolution

Hans de Nivelle

Institute of Computer Science,
University of Wroclaw, Poland,
`http://www.ii.uni.wroc.pl/∼nivelle`

Abstract. We study lemma simplification for geometric resolution, mainly from a theoretical viewpoint. For this purpose we develope a framework of proof permutations, which is somewhat similar to the permutions used in proofs of cut elimination. A side effect of this framework is, that one of the rules of the original geometric resolution calculus, could be simplified into simpler rules, which may have advantage for proof presentation.
Using the framework of proof permutations, we are able to prove theoretical results on proof length for three simplification rules that have been empirically successful in our implementation **geo**. These rules are subsumption, functional reduction, and equality splitting.
This work is work-in-progress, because there exist more simplification principles, for which at this moment we have neither theoretical results, nor practical experience.

1 Introduction

Geometric resolution is a proof search strategy, which was initiated in [3]. It works on a normal form called *g*eometric formula, which it tries to refute by enumerating candidate models.

The variant of geometric resolution studied in this paper was initiated in [6], and differs from the one in [3] in the following ways:

1. The structure of geometric formulas is more restricted, but it is allowed to contain equality.
2. Witnesses for quantifiers are enumerated in the same way as in [4], (and different from [3]), which makes it possible to obtain completeness for first-order logic with equality.
3. From every failed attempt to construct a model, a *lemma* is learnt, which ensures that no similar models will explored later during proof search.

We now define (our variant) of a geometric formula, then we outline the proof search algorithm for geometric resolution. It makes use of a resolution-like calculus (the geometric resolution calculus) with which it derives a closing lemma from every failed attempt to find a model. (Very similar to the way lemmas are learnt in modern approaches to DPLL, see [7]). The fact that this is always possible, was proven in [6].

After that we discuss what effects one can expect from simplification in geometric resolution, and we compare with simplification for saturation-based calculi. ([1]) The main difference is that in geometric resolution, inferences are controlled by the model search algorithm, where in saturation-based calculi, they are made blindly. Because of this, we expect the effect of simplification in geometric resolution to be more predictable.

Definition 1. *We assume an infinite set of* variables $\mathcal{V}$. *A* variable atom *is defined by one of the following two forms:*

- $x_1 \not\approx x_2$, *with* $x_1, x_2 \in \mathcal{V}$ *and* $x_1 \neq x_2$.
- $p(x_1, \ldots, x_n)$ *with* $n \geq 0$ *and the* $x_i \in \mathcal{V}$.

There are no constants and no function symbols in variable atoms. There are also no positive equalities. Negative of equalities of form $v \not\approx v$ are disallowed, because they are trivially false. Geometric formulas are built from variable atoms as follows:

Definition 2. *A* geometric formula *has form*

$$\forall \overline{x}\ A_1(\overline{x}) \wedge \cdots \wedge A_p(\overline{x}) \wedge x_1 \not\approx x_1' \wedge \cdots \wedge x_q \not\approx x_q' \rightarrow Z(\overline{x}),$$

in which $p \geq 0,\ q \geq 0$, *and the* $x_1, x_1', \ldots, x_q, x_q' \in \overline{x} \subseteq \mathcal{V}$.
The right hand side $Z(\overline{x})$ *must have one of the following three forms:*

1. *The false constant* $\bot$.
2. *A non-empty disjunction of non-disequality atoms* $B_1(\overline{x}) \vee \cdots \vee B_r(\overline{x})$ *with* $r > 0$.
3. *An existential formula of form* $\exists y\ B(\overline{x}, y)$ *with* $y \in \mathcal{V}$ *but* $y \notin \overline{x}$. *The variable* y *must occur in* $B(\overline{x}, y)$.

A formula of the first type is called lemma. *A formula of the second type is called* disjunctive. *A formula of the third type is called* existential.

The notations can be clarified as follows:

- In $\forall \overline{x}$, $\overline{x}$ denotes an enumeration of $\overline{x}$, in arbitrary order, mentioning each variable of $\overline{x}$ exactly once. The scope of $\forall \overline{x}$ is the complete geometric formula.
- In $A(\overline{x})$, $\overline{x}$ denotes a sequence of variables from $\overline{x}$. Variables may be repeated, and not all variables need to occur.
- Later in this paper, an expression of form $\Phi(\overline{x})$, $\Psi(\overline{x})$ or $X(\overline{x})$ will denote a conjunction of variable atoms, possibly containing disequality and non-disequality atoms.

It was shown in [6] that every first-order formula can be translated into an equi-satisfiable set of geometric formulas. The translation is related to the translation in of [2], and also somewhat related to the translation in [5]. The main difference with [5] is that we do not introduce functionality axioms. (although we will see them back as simplification rules later)

An *interpretation* can be viewed as a set of ground atoms I. An interpretation does not contain disequality atoms. Let $\rho = \forall \overline{x}\ \Phi(\overline{x}) \rightarrow Z(\overline{x})$ be a geometric rule. Let Θ be a substitution that assigns constants occurring in I to the variables in $\overline{x}$. We call the rule ρ *applicable* in I *with substitution* Θ if

1. for each disequality atom $x_1 \not\approx x_2 \in \Phi(\overline{x})$, we have $x_1\Theta \neq x_2\Theta$,
2. for each usual atom $A(\overline{x}) \in \Phi(\overline{x})$, the atom $A(\overline{x})\Theta$ occurs in I, and
3. $Z(\overline{x})\Theta$ is false in I under substitution Θ.
   (The constant $\perp$ is always false. The disjunction $B_1(\overline{x}) \vee \cdots \vee B_r(\overline{x})$ is false if none of the $B_j(\overline{x})\Theta$ occurs in I. A formula of form $\exists y\ B(\overline{x}, y)$ is false if I contains no constant c, s.t. $B(\overline{x}\Theta, c) \in I$)

As an example, $\forall x\ A(x) \rightarrow B(x)$ is not applicable in $\{A(0), B(0), A(1), B(1)\}$.
The rule $\forall x\ A(x) \rightarrow B(x) \vee C(x)$ is not applicable in $\{A(0), B(0), A(1), C(1)\}$.
It is applicable in $\{A(0), C(0), A(1)\}$ with substitution $\{x := 1\}$.
The rule $\forall xy\ A(x) \wedge B(y) \wedge x \not\approx y \rightarrow \exists z\ C(x, y, z)$ is applicable in $\{A(0), A(1), B(0), B(1), C(0, 1, 0)\}$ with substitution $\{x := 1, y := 0\}$. It is not applicable with any other substitution.
In geometric resolution, proof search proceeds by a combination of model search and lemma generation. The algorithm recursively tries to extend an interpretation I into a model. (i.e. an interpretation in which no rule is applicable) At each recursive level, the input consists of an interpretation I, and a set of geometric formulas G. When I cannot be extended into a model, the algorithm returns a pair (ρ, Θ) s.t. ρ is a lemma which applicable on I with substitution Θ. In case the lemma ρ is not already present in G, either I is a model, or there is an applicable rule ρ' which is not a lemma. In that case, the algorithm uses ρ' to extend I, by which it possibly has to backtrack. When backtracking is complete, it uses the geometric resolution rules to derive a ρ. It was proven in [6] that this is always possible, using geometric resolution. We describe the algorithm:

1. Select a rule ρ and a substitution Θ, s.t. ρ is applicable in I with Θ.
2. If no (ρ, Θ) was found, then I is a model. Report I.
3. If ρ is of type 1, then return (ρ, Θ).
4. If ρ is of type 2, then write $\rho = \forall \overline{x}\ \Phi(\overline{x}) \rightarrow B_1(\overline{x}) \vee \cdots \vee B_q(\overline{x})$. Recursively call the algorithm on each $I \cup \{B_j(\overline{x}\Theta)\}$. If one of the recursive calls results in a model, then report this model. Otherwise, the recursive calls will collect a sequence of pairs $(\rho_1, \Theta_1), \ldots, (\rho_q, \Theta_q)$, s.t. each ρ_j is a lemma applicable in the interpretation $I \cup \{B_j(\overline{x}\Theta)\}$ with Θ_j. Using disjunction resolution, it is possible to derive a pair (ρ', Θ'), s.t. ρ' is a lemma applicable in I with substitution Θ'.
5. If ρ is of type 3, then write $\rho = \forall \overline{x}\ \Phi(\overline{x}) \rightarrow \exists y\ B(\overline{x}, y)$. Assume that the constants occurring in I are called $c_0, \ldots, c_{n-1}$. Let c_n be the next constant which is not in I. For each i with $1 \leq i < n$, recursively call the model search algorithm on the interpretation $I \cup \{B(\overline{x}\Theta, c_i)\}$. Also call the model search algorithm on $I \cup \{B(\overline{x}\Theta, c_n)\}$. If one of the recursive calls constructs a model, then report this model. Otherwse, the recursive calls will collect a sequence of pairs $(\rho_1, \Theta_1), \ldots, (\rho_n, \Theta_n)$, s.t. each ρ_i is a lemma which is applicable in

$I \cup \{B(\overline{x}\Theta, c_i)\}$ with substitution Θ_i. Using existential resolution, one can derive a pair (ρ', Θ'), s.t. ρ' is a lemma applicable in I with substitution Θ'.

The most important heuristic of the algorithm is the choice which application (ρ, Θ) should be expanded. In general, it seems sensible to prefer lemmas over rules of other types. Between lemmas, **geo** currently decides by selecting the smaller lemma. Between rules $\forall \overline{x}\ \Phi(\overline{x}) \rightarrow Z(\overline{x})$ of type 2 or type 3, it decides by selecting the application for which $\Phi(\overline{x})\Theta$ has the smallest set of premisses, viewing this set as a multiset, and considering older atoms smaller than new atoms. In this way, fairness is guaranteed. However, there is a much variation possible and the effect of the heuristic on the performance of **geo** is largely unexplored.
The main distinction between geometric resolution and saturation-based theorem proving, (e.g. superposition) apart from the different normal form, is the fact that in geometric resolution, proof search is controlled by the model search algorithm. The model search algorithm decides which resolution inferences are made. When it needs a closing lemma, it calls the resolution module with detailed instructions about which inferences should be made. In saturation-based theorem, inferences are made essentially 'in a blind way'. Clauses are selected, and all possible inferences are made. This difference has some important consequences for the use of redundancy.
First recall that in saturation-based theorem proving a clause d is called *redundant* when it is implied by a set of clauses $c_1, \ldots, c_n$, such that (somewhat informally) $c_1, \ldots, c_n$ come before d in the multiset order. This notion was introduced in [1], and it is able to prove the completeness of most of the existing simplification rules for superposition theorem proving.
In this paper, we study only a relatively weak version of redundancy in the geometric setting: A lemma λ is redundant when it is implied by a set of lemmas $\lambda_1, \ldots, \lambda_n$, s.t. each of the lemmas $\lambda_1, \ldots, \lambda_n$ would be preferred over λ by the heuristic. This notion would cover λ-subsumption, but also the following example:
$\forall xyz\ S(x,y) \wedge S(x,z) \wedge y \not\approx z \rightarrow \bot$ and $\forall xy\ A(x) \wedge B(y) \wedge x \not\approx y \rightarrow \bot$ make $\forall xyzt\ A(x) \wedge S(x,y) \wedge B(y) \wedge S(y,z) \wedge y \not\approx z$ redundant. If one wants to obtain stronger notions of redundancy, where the implying clauses do not need to be lemmas, then one very probably needs to modify the heuristic. This will be a subject for future study.
Now that we have defined redundancy in our setting, we can discuss the differences with redundancy in saturation-based theorem proving.

1. Redundancy is much more important for saturation-based theorem proving it is than for geometric resolution. In saturation-based theorem proving, redundant clauses can become selected, they will create consequences, which again may be selected, etc. Therefore, high priority should be given to the deletion of redundant clauses.
   In geometric resolution, redundant lemmas will not be selected by the heuristic. Therefore, they will not be used in the derivation of new lemmas.

2. For saturation-based theorem proving, forward redundancy checking is more important than backward redundancy checking. For geometric resolution, forward redundancy checking is wasted effort: The algorithm will never create a redundant lemma. If $\lambda_1, \ldots, \lambda_n$ make λ redundant, then one of $\lambda_1, \ldots, \lambda_n$ would have been a closing lemma, and the algorithm would have reused it.

We have now seen, that whereas the price for tolerating redundant clauses in saturation-based theorem proving can be exponential, it causes only a small overhead in geometric resolution. So we could stop here, and make this is pleasant, short paper.
Unfortunately there is something more to tell, namely about simplification. Simplification is when one derives a consequence λ' of a lemma λ, s.t. λ' (possibly with some other lemmas) makes λ redundant. Although it is not possible that the model search algorithm derives a redundant lemma, it is possible that it derives a lemma that can be simplified. If one does not simplify a lemma that could have been simplified, it will possibly resolve with other lemmas that could have been simplified, and the effect will add up. For this reason, simplification is important for geometric resolution. As an example, consider the following simplification rule, which is an instance of functional reduction. Suppose that the rulesystem contains only one positive occurrence of A, which has form $\exists y\ A(y)$. Then in every interpretation constructed by the model search algorithm, there will be at most one constant c such that $A(c)$ occurs in the model. As a consequence, in any lemma containing more than one occurrence of A, all these occurrences can be unified (Because they will be unified anyway in every application of the lemma) If one does not unify all occurrences of A in a lemma, it may resolve with another lemma which also contains multiple occurrences of A. In that case, the resulting lemma will inherit the repeated occurrences of A from both its parents.
In the rest of this paper, we analyze redundancy-based simplification refinements of geometric resolution using proof theoretic methods. The reason for this is that we want to obtain results about proof length.
In saturation-based theorem proving, in general one cannot prove anything about proof length when redundant clauses are removed. The completeness proof implies that the new proof is smaller under the multiset order, but the length of the new proof can actually be bigger. One notable exception to this situation is subsumption. For subsumption, it can be proven that the new proof using the subsuming clause, is not longer than the proof using the subsumed clause.
Our intuition is that the chances of obtaining results about proof length with geometric resolution are better. The reason for this is the fact that the derivations are in some sense more deterministic, because they are governed by the model search algorithm. At this moment, we only have results for a few equality-based refinements, but we think that more results are possible. In order to prepare for proving the results about proof length, we first introduce a modification of the calculus of [6]. The reason for this is that the calculus of [6] is too much tuned towards the model search algorithm. (In particular the $\exists$-resolution rule of [6] is

too complicated to analyze) We show that the new calculus is as strong as the old calculus, and that proofs constructed by the model search algorithm are in a certain normal form, which we call $\approx\exists$-normal form. Note that this change of the calculus has no influence at all on the model search algorithm, because the lemmas derived remain the same.
We then study the effect of proof replacements. Suppose that one has a proof π obtained by a run of the model search algorithm. Let λ be a lemma occurring in π that was redundant at the moment it was used. If λ is made redundant by a set of lemmas $\lambda_1, \ldots, \lambda_n$, then it is possible to construct a proof π' which proves λ from $\lambda_1, \ldots, \lambda_n$. We can remove λ from π and replace it by the new proof π'. In all probability this new proof will not correspond to a run of the model search algorithm anymore, because of two possible reasons:

1. The new proof is not in $\approx\exists$-normal form.
2. The new proof is in $\approx\exists$-normal form, but is not consistent with the application selection heuristic.

We will introduce a set of proof permutations, with which every proof can be permuted back into $\approx\exists$-normal form. In case a clause was deleted due to redundancy, the proof $\pi[\pi']$ is almost in $\approx\exists$-normal form, except for the path leading to π' and π' itself. We will give examples (functional reduction, nested subsumption) where it can be shown that the permutation back to $\approx\exists$-normal form does not increase the size of the proof. This means that these refinements can be applied without restriction.

2 A Modified Calculus for Geometric Resolution

We present the modified calculus that we used for analyzing proof lengths. The main difference with the calculus of [6] is that we simplified the existential resolution rule and introduced a new rule called equality resolution. Apart from that, the only difference is that we made instantiation explicit. In practice the instantiations are determined by unification, but for analysis it is more convenient to have a calculus with explicit instantiation.

Definition 3. *The new calculus consists of the following rules:*

instantiation: *Let*

$$\rho = \forall \overline{x} \;\; \Phi(\overline{x}) \to Z(\overline{x})$$

be a geometric rule. Let Σ be a substitution of form $x_1 := x_2$, where $x_1 \in \overline{x}$. Then

$$\forall (\overline{x}\Sigma) \;\; \Phi(\overline{x}\Sigma) \to Z(\overline{x}\Sigma)$$

is an instance *of ρ. In case both x_1 and x_2 occur in $\Phi(\overline{x})$ or $Z(\overline{x})$, and $x_1 \neq x_2$, we call the instantiation a* proper *instantiation. In case x_2 does not occur in $\Phi(\overline{x})$ or $Z(\overline{x})$, we call the result a* renaming *of ρ.*

merging: *Let λ be a lemma of form*

$$\forall \overline{x}\ \Phi(\overline{x}) \wedge A(\overline{x}) \wedge A(\overline{x}) \rightarrow \bot.$$

Then the lemma

$$\forall \overline{x}\ \Phi(\overline{x}) \wedge A(\overline{x}) \rightarrow \bot$$

is a merging *of λ. $A(\overline{x})$ can be either a disequality atom, or a usual atom.*

disjunction resolution: *Let*

$$\rho = \ \forall \overline{x}\ \Phi(\overline{x}) \rightarrow B_1(\overline{x}) \vee \cdots \vee B_q(\overline{x})$$

be a disjunctive formula. For $1 \leq j \leq q$, let each

$$\lambda_j = \forall \overline{x}\ \Psi_j(\overline{x}) \wedge B_j(\overline{x}) \rightarrow \bot$$

be a lemma. Then

$$\forall \overline{x}\ \Phi(\overline{x}) \wedge \Psi_1(\overline{x}) \wedge \cdots \wedge \Psi_q(\overline{x}) \rightarrow \bot$$

is a disjunction resolvent of ρ with $\lambda_1, \ldots, \lambda_q$.

existential resolution: *Let $\rho = \forall \overline{x}\ \Phi(\overline{x}) \rightarrow \exists y\ B(\overline{x}, y)$ be an existential formula. Let λ have form*

$$\forall \overline{x}\ \forall y\ \Psi(\overline{x}) \wedge B(\overline{x}, y) \rightarrow \bot.$$

We have $y \notin \overline{x}$. Then

$$\forall \overline{x}\ \Phi(\overline{x}) \wedge \Psi(\overline{z}) \rightarrow \bot$$

is an existential resolvent of ρ with λ.

(degenerated) existential resolution: *Let $\rho = \forall \overline{x}\ \Phi(\overline{x}) \rightarrow \exists y\ B(\overline{x}, y)$ be an existential formula. Let λ have form*

$$\forall \overline{x}\ \forall y\ \Psi(\overline{x}) \rightarrow \bot.$$

We have $y \notin \overline{x}$. Then

$$\forall \overline{x}\ \Phi(\overline{x}) \wedge \Psi(\overline{x}) \rightarrow \bot$$

is a (degenerated) existential resolvent of ρ with λ.

equality resolution: *Let $\rho = \forall \overline{x}\ \Phi(\overline{x}) \wedge x_1 \not\approx x_2 \rightarrow \bot$ be a lemma. Let Σ be the substitution $x_1 := x_2$. Let λ be a lemma that can be written in the form*

$$\lambda = \forall (\overline{x}\Sigma)\ \Psi(\overline{x}\Sigma) \rightarrow \bot.$$

Then the lemma

$$\forall \overline{x}\ \Phi(\overline{x}) \wedge \Psi(\overline{x}) \rightarrow \bot$$

is an equality resolvent of ρ with λ.

Disjunction resolution is the same as hyperresolution. Equality resolution can be explained as follows: If Σ is the substitution $x_1 := x_2$, then $\forall(\overline{x}\Sigma)\Psi(\overline{x}\Sigma) \rightarrow \bot$is equivalent to $\forall\overline{x}\ x_1 \approx x_2 \wedge \Psi(\overline{x}) \rightarrow \bot$. In this formula, the equality can resolve with the disequality in ρ.
Most cases of degenerated existential resolution can be simulated by instantiationg y to one of the variables of $\overline{x}$. In that case, one would obtain $\forall\overline{x}\ \Psi(\overline{x}) \rightarrow \bot$ which subsumes the result. We keep the degenerated existential resolution rule because it is still needed for the case where $\overline{x}$ is empty, and in the future we may want to add types to geometric resolution. In that case it may happen that the type of y is not among the types of $\overline{x}$.

Theorem 1. *The calculus of Definition 3 is complete. If $\rho_1, \dots, \rho_n$ are geometric rules, and λ is a lemma, then if $\rho_1, \dots, \rho_n \models \lambda$, then λ is provable from $\rho_1, \dots, \rho_n$.*

For $\lambda = \bot$, completeness follows from the fact that the new calculus can simulate the old calculus. (This will be proven in the next section) For $\lambda \neq \bot$, completeness can be proven in a fairly standard way, by enumerating the models of $\rho_1, \dots, \rho_n$. Full completeness, (for $\lambda \neq \bot$) is not used in this paper, but it may be important in future studies of other redundancy rules.

3 $\approx\exists$-Normal Derivations

In this section we show that the calculus of Definition 3 can be used in the same way as the calculus of [6] for the generation of closing lemmas during model search. This change of calculus will have no impact on the model search algorithm and on the lemmas that it generates. [1] The reason for introducing the new calculus is that its permutations can be understood more easily. There could also exist an advantage in proof output for external verification, because the new calculus is more standard.
The difference between the old calculus and the calculus of Definition 3, is the replacement of the stronger existential resolution rule of [6] by the combination of a weaker existential resolution rule and equality resolution.
It is not possible to derive the stronger existential resolution rule in the new calculus, but it can be shown that every lemma obtained by an application of strong existential resolution can be obtained by a combination of weak existential resolution and equality resolution.
In [6], existential resolution is always used in the way shown in Figure 1. Figure 2 shows a proof of the same result using (non-generalized) existential resolution and equality resolution. First, the disequalities in $\forall\overline{x}\ \Phi(\overline{x}) \wedge B(\overline{x}, y) \wedge y \not\approx x_1 \wedge \cdots \wedge y \not\approx x_n \rightarrow \bot$ are resolved away one-by-one using equality resolution. On the result, (non-generalized) existential resolution is applied, and the same result is obtained.

[1] actually, they can sometimes be slightly stronger

It can be seen from Figure 2 that equality resolution is never applied 'stand alone' in proofs that are constructed by the model search procedure. Equality resolution is only used for resolving away the disequalities in a lemma of form $\forall \overline{x} y\ \Psi(\overline{x}) \wedge B(\overline{x}, y) \wedge y \not\approx x_1 \wedge \cdots \wedge y \not\approx x_n \rightarrow \bot$ in order 'to prepare it' for an existential resolution step in which $B(\overline{x}, y)$ is resolved away. We call proofs satisfying this condition $\approx\exists$-*normal*. Proofs constructed by the model search algorithm will be always $\approx\exists$-normal.

Fig. 1. Application of General $\exists$-Resolution

Let π be an existential resolution step

$$\frac{\forall \overline{x}\ \Phi(\overline{x}) \rightarrow \exists y\ B(\overline{x}, y) \qquad \forall \overline{x}\ \Psi(\overline{x}) \wedge B(\overline{x}, y) \wedge y \not\approx x_1 \wedge \cdots \wedge y \not\approx x_n \rightarrow \bot}{\forall \overline{x}\ \Phi(\overline{x}) \wedge \Psi(\overline{x}) \rightarrow B_1(\overline{x}, x_1) \vee \cdots \vee B_q(\overline{x}, x_n),}$$

It is used in a proof of form

$$\frac{\pi \qquad \forall \overline{x}\ X_1(\overline{x}) \wedge B_1(\overline{x}, x_1) \rightarrow \bot \qquad \cdots \qquad \forall \overline{x}\ X_n(\overline{x}) \wedge B_1(\overline{x}, x_n) \rightarrow \bot}{\forall \overline{x}\ \Phi(\overline{x}) \wedge \Psi(\overline{x}) \wedge X_1(\overline{x}) \wedge \cdots \wedge X_n(\overline{x}) \rightarrow \bot}\ (\vee\text{-res})$$

4 Redundancy through Proof Permutations

The final goal of the research reported in this paper is to study the effect of redundancy on proof length, using proof transformations. At present, we have only hard results for a restricted form of simplifications but we expect that more results are possible.

We outline the general technique: In case a lemma λ is made redundant by formulas $\rho_1, \ldots, \rho_n$, we take out every application of λ from the proof, and replace it by a proof of $\rho_1, \ldots, \rho_n \models \lambda$. The resulting proof is still a valid proof, but very probably it is not in $\approx\exists$-normal form anymore. The proof can be permuted back into $\approx \exists$-normal form, using proof permutations. If the new proof is not too long, in comparison to the old proof, then efficiency improves when λ is replaced by $\rho_1, \ldots, \rho_n$.
Since our calculus is similar to resolution, all transformations have essentially one of the forms that follow below. (Equality resolution is similar to standard resolution, if one keeps in mind that $\forall(\overline{x}\Sigma)\ \Phi(\overline{x}\Sigma) \rightarrow \bot$ with $\Sigma = \{x_1 := x_2\}$ is equivalent to $\forall \overline{x}\ x_1 \approx x_2 \wedge \Phi(\overline{x}) \rightarrow \bot$)
In both of the permutations, application of the first rule is postponed until after the second rule. The two possibilities depend on where the premiss of the second rule originates from. If it originates from only one of the parents of the first rule,

Fig. 2. Reconstruction of General ∃-Resolution

For each i with $1 \leq i \leq n$, let Σ_i be the substitution $\{y := x_i\}$.
Then each of the lemmas $\forall \overline{x}\ X_i(\overline{x}) \wedge B(\overline{x}, x_i) \rightarrow \perp$ can be written in the form

$$\forall((\overline{x}y)\Sigma_i)\ X_i((\overline{x}y)\Sigma_i) \wedge B((\overline{x}y)\Sigma_i, y\Sigma_i) \rightarrow \perp,$$

because no variable in $\overline{x}$ is modified by Σ_i, and $y\Sigma_i = x_i$. Let π_1 be the proof

$$\frac{\forall((\overline{x}y)\Sigma_1)\ X_1((\overline{x}y)\Sigma_1) \wedge B((\overline{x}y)\Sigma_1, y\Sigma_1) \rightarrow \perp \qquad \forall \overline{x}y\ \Psi(\overline{x}) \wedge B(\overline{x}, y) \wedge y \not\approx x_1 \wedge \cdots \wedge y \not\approx x_n \rightarrow \perp}{\forall \overline{x}y\ X_1(\overline{x}) \wedge \Psi(\overline{x}) \wedge B(\overline{x}, y) \wedge B(\overline{x}, y) \wedge y \not\approx x_2 \wedge \cdots \wedge y \not\approx x_n \rightarrow \perp}\ (\approx\text{-res})$$

Similarly, let π_2 be the proof

$$\frac{\forall((\overline{x}y)\Sigma_2)\ X_2((\overline{x}y)\Sigma_2) \wedge B((\overline{x}y)\Sigma_2, y\Sigma_2) \rightarrow \perp \qquad \pi_1}{\forall \overline{x}y\ X_1(\overline{x}) \wedge X_2(\overline{x}) \wedge \Psi(\overline{x}) \wedge B(\overline{x}, y) \wedge B(\overline{x}, y) \wedge B(\overline{x}, y) \wedge y \not\approx x_3 \wedge \cdots \wedge y \not\approx x_n \rightarrow \perp}\ (\approx\text{-res})$$

Continuing, one eventually reaches π_n, which has form

$$\frac{\forall((\overline{x}y)\Sigma_n)\ X_n((\overline{x}y)\Sigma_n) \wedge B((\overline{x}y)\Sigma_n, y\Sigma_n) \rightarrow \perp \qquad \pi_{n-1}}{\forall \overline{x}y\ X_1(\overline{x}) \wedge X_2(\overline{x}) \wedge \cdots \wedge X_n(\overline{x}) \wedge \Psi(\overline{x}) \wedge B(\overline{x}, y) \wedge \cdots \wedge B(\overline{x}, y) \rightarrow \perp}\ (\approx\text{-res})$$

At this point, one can apply ∃-resolution:

$$\frac{\forall \overline{x}\ \Phi(\overline{x}) \rightarrow \exists y\ B(\overline{x}, y) \qquad \dfrac{\pi_n}{\forall \overline{x}y\ X_1(\overline{x}) \wedge \cdots \wedge X_n(\overline{x}) \wedge \Psi(\overline{x}) \wedge B(\overline{x}, y) \rightarrow \perp}\ (\text{merging})}{\forall \overline{x}\ \Phi(\overline{x}) \wedge X_1(\overline{x}) \wedge \cdots \wedge X_n(\overline{x}) \wedge \Psi(\overline{x}) \rightarrow \perp}\ (\exists\text{-res})$$

then the transformation is unproblematic, because the size of the proof does not increase. If the premiss of the second rule originates from both parents of the first rule, then the size of the proof does increase. We give examples of both situations:
unproblematic:

$$\dfrac{\dfrac{A \lor B \lor R_1 \qquad \neg A \lor R_2}{B \lor R_1 \lor R_2}\,(\text{res}) \qquad \neg B \lor R_3}{R_1 \lor R_2 \lor R_3}\,(\text{res})$$

permutes into

$$\dfrac{\dfrac{A \lor B \lor R_1 \qquad \neg B \lor R_3}{A \lor R_1 \lor R_3}\,(\text{res}) \qquad \neg A \lor R_2}{R_1 \lor R_2 \lor R_3}\,(\text{res})$$

problematic:

$$\dfrac{\dfrac{A \lor B \lor R_1 \qquad \neg A \lor B \lor R_2}{B \lor R_1 \lor R_2}\,(\text{res+merging}) \qquad \neg B \lor R_3}{R_1 \lor R_2 \lor R_3}\,(\text{res})$$

permutes into

$$\dfrac{\dfrac{A \lor B \lor R_1 \qquad \neg B \lor R_3}{A \lor R_1 \lor R_3}\,(\text{res}) \qquad \dfrac{\neg A \lor B \lor R_2 \qquad \neg B \lor R_3}{\neg A \lor R_2 \lor R_3}\,(\text{res})}{R_1 \lor R_2 \lor R_3}\,(\text{res+merging})$$

As mentioned above, proof permutations can be used to bring a proof back into $\approx\exists$-normal form, after some lemma λ has been replaced by some formulas $\rho_1, \ldots, \rho_n$ that make it redundant. They also can be used to make a proof consistent with the selection heurstic of the search algorithm.
Unfortunately, each time a rule application from the redundancy proof is permuted down, it may double the part of π that it permutes through, due to the problematic permutations.
We conclude that, when designing redundancy strategies, one should look for strategies that do not cause too much doubling. One of the possible ways to

do this, is by showing that there exists a low upperbound on one of the copies. Since the other copy is not bigger than the original proof, the increase in proof size can be kept small in this way. We end the paper with a few examples, and explain for each of the examples how this can be done.

– We first study functional reduction. Functional reduction exploits the fact that some predicate can be shown to be functional in one or more of its arguments during proof search. Let F be a predicate whose only positive occurrences are in formulas of form $\forall \overline{x}\ \Phi(\overline{x}) \rightarrow \exists y\ F(\overline{x}, y)$. The model search algorithm will create an atom $F(\overline{c}, d_1)$ only in case there exists no other atom of form $F(\overline{c}, d_2)$ in the interpretation. Therefore, in every interpretation I the last argument of F is a function of the other arguments. This fact can be used in simplifications. Whenever a lemma contains two atoms of form $F(\overline{x}, y_1)$ and $F(\overline{x}, y_2)$, the variables y_1 and y_2 can be unified.
  In order to ensure that the simplified clause implies the original clause, we add so called *inductive axioms.* The axiom for F is

$$\forall \overline{x} y_1 y_2\ F(\overline{x}, y_1) \wedge F(\overline{x}, y_2) \wedge y_1 \not\approx y_2 \rightarrow \bot.$$

  Since F is functional, the inductive axiom will never be applicable. However, it triggers functional reduction. Consider the formula

$$\forall \overline{x} y z\ F(\overline{x}, y) \wedge F(\overline{x}, z) \wedge B(y, z) \rightarrow \bot,$$

  which can be functionally reduced into

$$\forall \overline{x} y\ F(\overline{x}, y) \wedge B(y, y) \rightarrow \bot.$$

  Using the inductive axiom for F, one can can construct the following proof:

$$\dfrac{\forall \overline{x} y z\ F(\overline{x}, y) \wedge F(\overline{x}, z) \wedge y \not\approx z \rightarrow \bot \qquad \forall \overline{x} z\ F(\overline{x}, z) \wedge B(z, z) \rightarrow \bot}{\forall \overline{x} y z\ F(\overline{x}, y) \wedge F(\overline{x}, z) \wedge B(y, z) \rightarrow \bot}\ (\approx\text{-res})$$

  The $\approx$-resolution has to be permuted down in order to restore $\approx\exists$-normality. We show that during these permutations, the $\approx$-resolution disappears. First the $\approx$-resolution permutes down to the point where either $F(\overline{x}, y)$ or $F(\overline{x}, z)$ is used in disjunction resolution or exists resolution. At this point, since functionality holds for F, $F(\overline{x}, y)$ and $F(\overline{x}, z)$ have to be merged before they are resolved away. But then the $\approx$-resolution will become an instantiation when it permutes with the merging.
  Using the same argument, it can be shown that the $\approx$-resolution also disappears in case the $\approx$-resolution is already in $\approx\exists$-normal form.
  Note the peculiar way in which the inductive axiom $\forall \overline{x} y z\ F(\overline{x}, y) \wedge F(\overline{x}, z) \wedge y \not\approx z \rightarrow \bot$ was used in the simplification. If F is indeed functional, the axiom will never be applicable, and therefore never occur in a proof. The axiom caused the functional reduction step, but the correctness of the step does not rely on it. Soundness follows from the fact that functional reduction is a form of instantiation.

– Next we consider nested subsumption. Suppose we want to use $a \approx b$ to delete $s(a) \approx s(b)$:

$$\frac{\forall xyt\ S(x,y) \wedge S(x,t) \wedge y \not\approx t \to \bot \qquad\qquad \forall xz\ A(x) \wedge B(z) \wedge x \not\approx z \to \bot}{\forall xyzt\ A(x) \wedge S(x,y) \wedge B(z) \wedge S(z,t) \wedge y \not\approx t \to \bot}\ (\approx\text{-res})$$

The equality resolution step uses the substitution $\{x := z\}$. By the same argument as in the previous case, it can be seen that the $\approx$-resolution will disappear when it is permuted downward.

– The following simplification has no counterpart in resolution, because it can be expressed only with relations. In the presence of

$$\lambda_1 = \forall xyzt\ A(x) \wedge S(x,y) \wedge B(z) \wedge S(z,t) \wedge y \not\approx t \to \bot,$$

the formula

$$\lambda_2 = \forall xyzt\ \alpha\beta\gamma\delta\ A(x) \wedge S(x,y) \wedge C(\alpha) \wedge F(y,\alpha,\beta) \wedge$$
$$B(z) \wedge S(z,t) \wedge D(\gamma) \wedge F(t,\gamma,\delta) \wedge \beta \not\approx \delta \to \bot$$

can be simplified into

$$\lambda_3 = \forall xzt\alpha\beta\gamma\delta A(x) \wedge S(x,t) \wedge C(\alpha) \wedge F(t,\alpha,\beta) \wedge$$
$$B(z) \wedge S(z,t) \wedge D(\gamma) \wedge F(t,\gamma,\delta) \wedge \beta \not\approx \delta \to \bot.$$

λ_2 is an equality resolvent of λ_1 and λ_3. In case all positive occurrences of the predicates S, A, B are in existential formulas, the only way in which formula λ_1 can be used is in an equality resolution, followed by an $\exists$-resolution. It follows that the path from λ_1 towards the $\approx$-resolution must have length 1. Therefore the increase in proof length is at most 1.

5 Conclusions and Future Work

First, we have modified the calculus of [6], in such a way that that the resulting calculus is close to standard resolution. The most notable difference, which is the equality resolution rule, can be explained from the equivalence

$$\forall(\overline{x}\{x_1 := x_2\})\ \Phi(\overline{x}\{x_1 := x_2\}) \to \bot \quad \Leftrightarrow \forall \overline{x}\ x_1 \approx x_2 \wedge \Phi(\overline{x}) \to \bot.$$

We intend to change **geo** to use the new calculus, because we expect that it will make proof verification easier.

We have introduced a proof theoretical method with which it is possible to justify some of the successful forms of redundancy in geometric resolution. With this method, we can rigorously prove that functional reduction and nested subsumption (the first two cases in the previous section) do not increase proof length.

At this moment, we do not have sufficient empirical evidence to be able to tell whether a more sophisticated form of analysis will be necessary. In particular, it may be necessary to take reuse of proofs into account. A concrete example where this could be the case is the last case of previous section, (the one with $\lambda_1, \lambda_2, \lambda_3$) It seems likely that also in the case when S occurs positively in disjunctive formulas, the replacement $\lambda_2 \Rightarrow \lambda_3$ would be an improvement. In order to justify such replacements, one could argue that it is very likely (perhaps provable) that the system will encounter situations in which λ_1 alone is applicable, as well as situations where λ_2 is applicable. The effect of the simplification can be viewed as replacing λ_2 by $\lambda_2 \backslash \lambda_1$. If, whenever $\lambda_2 \backslash \lambda_1$ is applied, λ_1 has already been applied before, then the simplification has caused no loss in logical strength. At this moment, we first need to collect some more experience with ad hoc implemented redundancy criteria.

References

1. Leo Bachmair and Harald Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *Journal of Logic and Computation*, 4(3):217–247, 1994.
2. Peter Baumgartner, Alexander Fuchs, Hans de Nivelle, and Cesare Tinelli. Computing finite models by reduction to function-free clause logic. *Journal of Applied Logic*, 2007.
3. Marc Bezem and Thierry Coquand. Automating coherent logic. In Geoff Sutcliffe and Andrei Voronkov, editors, *LPAR*, volume 3835 of *LNCS*, pages 246–260. Springer, 2005.
4. François Bry and Sunna Torge. A deduction method complete for refutation and finite satisfiability. In Jürgen Dix, Luis Fariñas del Cerro, and Ulrich Furbach, editors, *JELIA*, volume 1489 of *LNCS*, pages 122–138. Springer Verlag, 1998.
5. K. Claessen and N. Sörensson. New techniques that improve MACE-style finite model finding. In *CADE-19, Workshop W4. Model Computation — Principles, Algorithms, Applications*, 2003.
6. Hans de Nivelle and Jia Meng. Geometric resolution: A proof procedure based on finite model search. In John Harrison, Ulrich Furbach, and Natarajan Shankar, editors, *International Joint Conference on Automated Reasoning 2006*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 303–317, Seattle, USA, August 2006. Springer.
7. Niklas Eén and Niklas Sörensson. An extensiable sat solver. `http://www.cs.chalmers.se/~nik/` , 2003.