LOGIC JOURNAL of the IGPL

Volume 11 Number 6 November 2003

Interest Group in Pure and Applied Logics

OXFORD UNIVERSITY PRESS

ISSN 1367-0751
www.oup.co.uk/igpl
Printed in Great Britain by
Butler & Tanner

Subscription Information

Volume 11, 2003 (bimonthly) Full: Europe pounds sterling 355; Rest of World US$ 575. Personal: pounds sterling 180 (US$ 288). Please note that personal rates apply only when copies are sent to a private address and payment is made by a personal cheque or credit card.

Order Information

Subscriptions can be accepted for complete volumes only. Prices include air-speeded delivery to Australia, Canada, India, Japan, New Zealand, and the USA. Delivery elsewhere is by surface post. Payment is required with all orders and may be made in the following ways:

Cheque (made payable to Oxford University Press)
National Girobank (account 500 1056)
Credit card (Access, Visa, American Express)
UNESCO Coupons
Bankers: Barclays Bank plc, PO Box 333, Oxford, UK. Code 20-65-18, Account 00715654.

Requests for sample copies, subscription enquiries, orders and changes of address should be sent to the Journals Subscriptions Department, Oxford University Press, Great Clarendon Street, Oxford OX2 6DP, UK. Tel: +44 (0) 1865 267907. Fax: +44 (0) 1865 267485. Email: jnl.orders@oup.co.uk

Advertisements

Advertising enquiries should be addressed to Peter Carpenter, PRC Associates, The Annexe, Fitznells Manor, Chessington Road, Ewell Village, Surrey KT17 1TF, UK. Tel: +44 (0) 181 786 7376. Fax: +44 (0) 181 786 7262.

Copyright

Logic Journal of the IGPL (ISSN 1367-0751) is published bimonthly in January, March, May, July, September and November by Oxford University Press, Oxford, UK. Annual subscription price is US$ 575.00. *Logic Journal of the IGPL* is distributed by M.A.I.L. America, 2323 Randolph Avenue, Avenel, NJ 07001. Periodical postage paid at Rahway, New Jersey, USA and at additional entry points.
US Postmasters: Send address changes to *Logic Journal of the IGPL*, c/o Mercury International, 365 Blair Road, Avenel, NJ 07001, USA.

Back Issues

The current plus two back volumes are available from Oxford University Press. Previous volumes can be obtained from the Periodicals Service Company, 11 Main Street, Germantown, NY 12526 USA. Tel: +1 (518) 537 4700, Fax: +1 (518) 537 5899.

Logic Journal of the IGPL

Volume 11, Number 5, November 2003

Contents

Original Articles

Please visit the journal's World Wide Web site at
http://www.oup.co.uk/igpl

Logic Journal of the Interest Group in Pure and Applied Logics

Scope of the Journal

The *Journal* is the official publication of the International Interest Group in Pure and Applied Logics (IGPL), which is sponsored by The European Foundation for Logic, Language and Information (FoLLI), and currently has a membership of over a thousand researchers in various aspects of logic (symbolic, computational, mathematical, philosophical, etc.) from all over the world.

The *Journal* is published in hardcopy and in electronic form six times per year. Publication is fully electronic: submission, refereeing, revising, typesetting, checking proofs, and publishing, all is done via electronic mailing and electronic publishing.

Papers are invited in all areas of pure and applied logic, including: pure logical systems, proof theory, model theory, recursion theory, type theory, nonclassical logics, nonmonotonic logic, numerical and uncertainty reasoning, logic and AI, foundations of logic programming, logic and computation, logic and language, and logic engineering.

The *Journal* is an attempt to solve a problem in the logic (in particular, IGPL) community:

- Long delays and large backlogs in publication of papers in current journals.
- Very tight time and page number limits on submission.

Papers in the final form should be in LaTeX. The review process is quick, and is made mainly by other IGPL members.

Submissions

Submissions are made by sending a submission letter to the e-mail address: jigpl@dcs.kcl.ac.uk, giving the title and the abstract of the paper, and informing: of how to obtain the file electronically or, by sending 5 (five) hardcopies of the paper to the Editor-in-Chief.

As from Volume 10 (2002) the Journal is indexed in:
- Science Citation Index Expanded (SCIE)
- Research Alert
- CompuMath Citation Index (CMCI)

URL: www.oup.co.uk/igpl

Normative Models of Rational Agency: The Theoretical Disutility of Certain Approaches

DOV M. GABBAY, *Department of Computer Science, King's College London, Strand, London WC2R 2LS, UK.*
email: dg@dcs.kcl.ac.uk

JOHN WOODS, *The Abductive Systems Group, University of British Columbia, 1866 Main Mall, Vancouver, BC, V6T 1K1, Canada and Department of Computer Science, King's College London, Strand, London WC2R 2LS, UK.*
email: jhwoods@interchange.ubc.ca and woodsj@dcs.kcl.ac.uk

Abstract

Much of cognitive science seeks to provide principled descriptions of various kinds and aspects of rational behaviour, especially in beings like us or AI simulacra of beings like us. For the most part, these investigators presuppose an unarticulated common sense appreciation of the rationality that such behaviour consists in. On those occasions when they undertake to bring the relevant norms to the surface and to give an account of that to which they owe their legitimacy, these investigators tend to favour one or other of three approaches to the normativity question. They are (1) the *analyticity* or *truth-in-a-model* approach; (2) the *pluralism* approach; and (3) the *reflective equilibrium* approach.

All three of these approaches to the normativity question are seriously flawed, never mind that the first two have some substantial (and often tacit) provenance among logicians and the third has enjoyed a flourishing philosophical career.

Against these views, we propose a strong version of what might be called *normatively immanent descriptivism*. We attempt to elucidate its virtues and to deal with what appears to be its most central vulnerability, embodied in the plain fact that actual human behaviour is sometimes irrational.

> A great deal of modern economics is based on the accommodations of the discipline to the demand of mathematics. Models based on dynamic disequilibrium or nonlinearity ... are intractable to mathematical formulation. Without the postulate of general equilibrium there is no solution to the system of simultaneous equations which economics needs to prove that markets allocate resources efficiently. That is why economics has been uncomfortable with attempts to model economies as sequences of events occurring in historical time — which is what they are. There is a nice irony here. The more "formal" economics becomes, the more it has to treat reality as a purely logical construction. When it looks on the market system and finds it good, its admiring gaze is actually directed at its own handiwork.
>
> Robert Skidelsky, *New York Review*, 8 March, 2001.

1 A Brief Word on Modelling

Given its rapid absorption by mathematics, it is hardly surprising that modern mainstream logic has turned to mathematics for its working notion of model. Indeed it could be said that logic's model of the model was the mathematical model. Such in turn was the choice of the natural sciences, certainly of those of them for the expression of whose laws mathematics is indispensable. The more complex of the natural sciences fared less well in capturing its essential insights in mathematical formalisms. This, too, is not surprising, given the comparative messiness and lack of generality of, say, the life sciences. Biology is an interesting test case for the would-be practical logician. There is a use of the word 'theory' in which a scientific accounts theoretical component is that which falls beyond the ambit of observation. In many cases, a theory is little more than a mechanical device that computes or predicts output from a system's inputs. In biology, perhaps the classic example is theoretical populations and evolutionary genetics. Here all the basic processes are quite well known. These include the operations of inheritance, the facts of mutation, migration, and the mechanisms of natural selection under varying conditions of survival and fertility. Thus

> Theoretical evolutionary genetics assembles all these phenomena into a formal mathematical structure that predicts changes in the genetic composition of populations and species over time as a function of the numerical values of these elementary processes. [21, p. 40]

Here the formalisation works. It works because the underlying mechanisms are known. There are lots of cases in which this is not so. The formal modelist has, apart from desistance, no option but to fly high. For here, in its pure form, the mechanical formalities are posited without any direct connection to underlying material data. This makes the theorist's formal model an empirically unsupported place-holder for the actual dynamical details once they become known. An especially extreme, and failed, example of this theoretical high-flying was the Rashevsky school of mathematical biophysics, which operated in the late 1930s at the University of Chicago. Within three decades the movement was dead, made so by the extreme over-idealisation of his physical models, so radical as to make them empirically useless. The Rashevsky collapse teaches an important lesson. It is that certain biological processes may not admit of accurate mathematical expression. There are still other cases in which postulated mathematical expressions of biological processes turn out to be right, but a good deal less than optimal even so. This we see in the case of Turing's conjecture that early embryonic development could be understood as the result of different concentrations of (observationally undetermined) molecules, distributed differentially within the embryo. This is right as far as it goes. But developmental genetics owes nothing to Turing's model. What it achieves in accuracy it pays for in over-simplification.

These are lessons for the practical logician to take to heart. For one thing, the behaviour of human animals exceeds in complexity any grasp we have of the fruit fly, no matter how exhaustive. Apart from that, there is the vexing problem of own below, which is where much of a practical agent's cognitive agenda is transacted. The republic of down below is ringed by unwelcoming borders. Not only is much of what goes on there inaccessible to introspection, but experimental probes are heavily constrained by the ethical requirement to do no harm.[1]

[1] Even so, there are clear experimentally derived intuitions about how things go down below, not only in the case of reasoning, but also in the case of vision; concerning which, see [30].

2 Logic and Mistakes of Reasoning

Everyone makes mistakes, of course. Not every instance of a mistake, even of an avoidable mistake, is a failure of rationality, but we may take it that when rationality does fail it does so because mistakes have been made of a nature and on a scale and with a frequency of requisite gravity.

There are all kinds of mistakes, needless to say. You might mistake Hortense for Sarah or an utterance of "the cross I'd bear" for an utterance of "the cross-eyed bear". You might misremember Carol's birthday as the 16th of March. Or you might detach a proposition Φ on the strength of a pair of propositions $\ulcorner$*If* Φ *then* $\Psi\urcorner$ and Ψ. It is widely believed that this latter is a mistake of reasoning and that rationality fails when mistakes of this kind are made. We have it, then, that a theory of rationality or rational agency must contain a theory of mistakes of such a kind.

For a very long time it has been taken for granted that, for reasoning of a suitably rigorous nature, a theory of mistakes of reasoning is furnished by a logic of an appropriate design. Thus, for reasoning that might answer to the standards of strict proof, there is deductive logic, and for reasoning that is characteristic of science, there is inductive and abductive logic. Approached in this way, a mistake of reasoning is reasoning which disconforms to a logical rule or principle or norm. It would follow from this that one of the tasks of the logician is to privilege those propositions, standards or rules disconformity to which *constitutes* a mistake. Our task in this note is to press the following question:

> How does the logician know what proposition, standard or rule to privilege in this way?

A common answer is that the logician constructs normative models of reasoning.

3 Normative Models

Here is a case in point, having to do with the rationality of argumentative practice.

> In order to clarify what is involved in viewing argumentative discourse as aimed at resolving a difference of opinion, the theoretical notion of a critical discussion is given shape in an ideal model specifying the various stages, in the resolution process and the verbal moves that are instrumental in each of these stages. The principles *authorizing* these moves are accounted for in a set of rules for the performance of speech acts. Then together, these rules constitute a *theoretical definition* of a critical discussion [36, p. 280 emphasis added]. Moreover, [a]nalyzing argumentative discourse amounts to interpreting it from a theoretical perspective. This means that the interpretation is guided by a *theoretically motivated model* that provides a point of reference for the analysis, emphasis added. As a consequence of the adoption of such a theoretical perspective, specific aspects of the discourse are *highlighted* in the analysis [36, p. 191].
>
> ...[And so], [e]ven a discourse which is clearly argumentative will in many respects not correspond to the ideal model of a critical discussion [36, p. 299, n. 49]. ...In this model, the rules and regularities of *actual* discourse are brought together with *normative* principles of goal-directed discourse. The model of critical discussion is an abstraction, a theoretically motivated system for ideal resolution-oriented discourse [36, p. 311 emphasis added].

Consider now a principle P of the model and a piece of argumentative behaviour B that disconforms to it. If P is genuinely normative then B is a mistake (at a minimum). What is to be done if P's legitimacy is called into question? There are three possibilities to consider.

i. *The truth-in-a-model manoeuvre*
ii. *The reflective equilibrium manoeuvre*
iii. *The pluralism manoeuvre*

We shall consider each of these in turn.

4 Privileging *P*

4.1 The Analyticity Manoeuvre

According to the *analyticity* or *truth-in-a-model manoeuvre*, P enters the ideal model on the strength of the theorist's stipulation or by consequence of such stipulations. P is then true in the model, or analytically true in the theory whose model it is. There is nothing that says that a theorist can't privilege P in this way. But notice that if an outsider challenges P's truth, it will do no good for our theorist to reply that P's truth is assured since P is true in the model. In challenging P's truth the critic does not have it in mind to query whether P is true in the model. The point rather is whether P's truth in the model secures P's *truth*. The theorist's reply therefore commits the ancient fallacy of *secundum quid* or omitting a qualification. He reasons from the premiss that P is true in the model to the conclusion that P is true, omitting the essential qualification "in the model".

It is true and important that in the general case no theorist will stipulate the truth of any proposition he has reason to believe is false.[2] We may take it as given that ideal-modelists who are especially earnest will seek a place in their models for only those propositions that they think are already true and hence genuinely normative. Even so, there are two problems with this. One is that if P is already true and genuinely normative, it doesn't need membership in an ideal model to privilege it. A principal rationale for having ideal models is thereby lost. The other difficulty is that P's extra-model truth and normativity are also no answer to P's challenger, who in making his challenge calls these very features into question. Thus each of the theorist's defences is fallacious. "Because P is true in the model" commits the *secundum quid* fallacy; and "Because P is true and genuinely normative" commits the fallacy of begging the question.

Historically the most prominent version of the truth-in-a model manoeuvre is the axiomatic method. When the axioms of naive set theory were first introduced, they were thought to be analytic of the very idea of set. Then in a fateful letter to Frege in 1903 Russell communicated the news that blew intuitive set theory to pieces; viz., that the intuitive axioms are inconsistent.

Here is a quick example of our present point. In introducing the notion of an ideal reasoner, the theorist might mean (as some theorists have meant) a reasoner who, among other things, adjusts his deductive practice by conforming proper subsets of his inferences to the valid rules of deduction. He might also propose that ideal reasoners close their beliefs under consequence, and also that their beliefs are transparent, i.e., they believe that they believe what they believe, etc. If the theorist now wants to introduce the idea of a normative model of reasoning, he might cash the notion of normativity as follows: reasoning in the normative model

[2] There are exceptions, of course. Gell Mann initially embraced quarks, thinking that there was no good reason to suppose that they exist. Planck thought the same of quanta.

is reasoning done by ideal reasoners. The normativity of the normative model is secured by the ideality of its ideal participants. (This is how reasoning *should* be transacted by ordinary reasoners because this is how reasoning *is* transacted by ideal reasoners, who, among other things, choose deductive strategies licensed by demonstrably valid rules of deduction). If a skeptic asks, "Why *these* rules?", he can be told that they are provably valid and, in standard elementary formulations, complete.

But this is inadequate. Any real reasoner who even attempted to fulfill this deductive ideal would quickly paralyze thought and action and in any event. As [15, 2] and others have pointed out, the rules of deduction are not invariably good rules of inference. What they are is truth-preserving. But there are cognitive goals galore in which either truth-preservation is not a possible goal or it is an attainable but unnecessarily costly standard for the cognitive agent on the ground to be required to hit.[3]

So a condition on our normative model — a negative condition — is this: beyond a certain point, do not even approximate to the behaviour of ideal deducers. Alternatively: constrain your notion of an ideal reasoner in such a way that ordinary reasoners can realistically approximate to it.

We now turn to the *reflective equilibrium manoeuvre*.

4.2 *Reflective Equilibrium*

Nelson Goodman writes in a celebrated passage,

> Principles of deductive inference are justified by their conformity with accepted deductive practice. Their validity depends upon accordance with the particular deductive inferences we actually make and sanction. If a rule yields unacceptable inferences, we drop it as invalid. Justification of general rules thus derives from judgements rejecting or accepting particular deductive inferences. This looks flagrantly circular.... But this circle is a virtuous one. The point is that rules and particular inferences alike are justified by being brought into agreement with each other. A rule is amended if it yields an inference we are unwilling to accept; an inference is rejected if it violates a rule we are unwilling to amend. The process of justification is the delicate one of making mutual adjustments between rules and accepted inferences; and in the agreement achieved lies the only justification needed for either. All this applies equally well to induction [13, pp. 63–64].

Coinage of the term "reflective equilibrium" rests not with Goodman but with Rawls [29, p. 20]. We arrive, Rawls says, at "principles which match our considered judgements duly pruned and adjusted" [29, p. 20]. The matching is reflective since "we know to what principles our judgements conform and the premises of their derivation" [29, p. 20]; and it induces an equilibrium "because our principles and judgements coincide" [29, p. 20]]. It is all a matter of "everything fitting together into one coherent view" [29, p. 21].

We may find ourselves rather provoked by Goodman's casual-seeming assertion that "in the agreement achieved [by reflective equilibrium] lies the only justification needed for either [i.e, our principles and our accepted inferences]". A critic might find that Goodman has confused his modalities; that, confusing necessity with possibility, he should have said, "this is the only justification possible". This, of course, is one of the classical positions. It allows

[3] Any cognitive goal that requires ampliative reasoning is a goal for which validity is an inappropriate standard. Equally if someone's goal is simply to have some reason to believe a proposition, validity is a standard of unnecessary stricture.

that the best we can do by way of justification is not good enough to subdue the subjectivity of belief and, hence is not good enough to count as justification. Intersubjectivity is just a lot of individual concurrence; agreement, even widespread agreement, does not convert *endoxon* (expert opinion) to *epistemē* (genuine knowledge).

Our own view is that the reflective equilibrium approach comes close to getting something right (see below). But as it is conventionally formulated, it is just the community standards approach to rationality, which has nothing more going for it than, say, the community standards approach to pornography.

It is perhaps something less than tactical prudence to make such short shrift with an approach to normativity that has attracted a good deal in the way of sophisticated articulation (e.g., [3], [6] and [1]), and a good deal of philosophical support. A more detailed indication of our reservations can be found in [43, ch. 8]. For present purposes it will suffice to tease out the kernel of our disagreement. Consider, first, a case in which the reflective equilibrium approach clearly works. Suppose that we undertook to know what it is that justifies the norms under which a piece of acoustic disturbance or some scribbling on a page constitutes a *bona fide* sentence of English. The equilibriumist's answer is that the correctiveness of those norms consist in their conformity to present linguistic practice in communities of English speakers. On this view, it is further recognized that when settled practice changes so, too do the rules that proclaim it correct, moreover, and that rule-change is ultimately the resultant of what, by the old rules, was persistent wide-spread accumulation of disconformity. On this approach, there is no *à priori* upper bounded on the number or extent of the changes that correct English might undergo over time. In languages as old as English, the change from Caedmon to Martin Amis is enormous, making Caedmon's hymn quite inaccessible to present-day speakers of English, save those with benefit of specialised tuition. That this is so makes it perfectly intelligible to say, e.g., that string "S" is a good English sentence today, but that it was no such thing in, say, 850 AD. Another way of saying this is that the reflective equilibrium approach to English sentencehood neither requires nor provides a stable extension for the predicate "is a sentence of English". The same is true of any concept for which the reflective equilibrium approach is tenable. And this is quite enough to show that the reflective equilibrium approach cannot work for the concept of reasonable inference. Nothing in the empirical record to indicate that the concept of rationality has anything like the potential for extensional instability that the equilibrium approach, if correct, would require it to have. We should expect there to be radical differences between the year 850 and 2004 concerning how people speak English. We should not expect there to be radical differences — or differences of any real degree of significance — between 850 and 2004 concerning how people infer.[4] Thus, in a stripped-down version, is our case against reflective equilibrium.

4.3 Pluralism

A third possibility is the *pluralism manoeuvre*. It, too, is an approach to normativity occasioned by disagreements over the truth or normative legitimacy of such things as (putative) logical principle. (For concreteness, we might keep in mind the controversy that still rages in some quarters between classical and relevant logicians over the legitimacy of *DS* (Disjunctive Syllogism)). The pluralist manoeuvre is careless resort to what we call the ambiguation strategy.

[4] The same would not, of course, apply to all respects of belief-acquisition. But it would apply to those sectors of belief-dynamics having to do with revision and update in the light of (what is taken as) new information (see section 5.1 below).

Ambiguation strategy. If parties S and S' are locked in a disagreement about whether a principle P is true (or sound or normatively legitimate), then the preferred course is to attribute ambiguity, as in the following example: "DS is valid in classical logic and invalid in relevant logic. Hence DS is classically valid and relevantly invalid". (See Woods [42] and [43].)

By these lights, there can be no question of whether DS is valid *just so*. The validity and invalidity of logical rules is intrinsically a matter of system-relativity. And certainly that a rule is invalid in a given system is not the slightest reason against making it valid in some other system. On the face of it, a plausible enough strategy for handling classical-relevant standoffs concerning DS, it nevertheless lacks a principled reason to bar making, say, affirming the consequence a valid rule, or doing the same with the distribution of negation through conjunction. So we find it necessary to reject the ambiguation strategy, at least in its unconstrainedly relativistic form. But there is another variant of ambiguation which we take to be sound, which can be briefly sketched as follows. We call it the *reconciliation strategy*. In the present example, it bids the disputing parties to try to find things — *different* things — for DS to be true *of*. Thus, one possibility is that DS gives an invariably sound truth condition on the consequence relation but that it is not invariably a good rule of inference (e.g., in contexts such as that in which $\ulcorner \Phi \vee \Psi \urcorner$ arises from Φ, and Ψ arises from $\ulcorner \Phi \vee \Psi \urcorner$ and $\ulcorner \neg \Phi \urcorner$.

We wish to stress that in its first, or *pluralist* version, the ambiguation manoeuvre implies rampant logical relativism. There is *no* P for which a derivation is not possible in some system or other, no matter how retrograde. The relativism is obviously excessive, as we have said. Even so, there is, relatedly, something even less desirable about the pluralist option. For we have it, again, that for every P there is a system S such that P is S–valid. But there is no reason to think that S–validity, for arbitrary S, is normative for *rationality*. Thus the pluralist manoeuvre makes evident *en large* what we should have noticed earlier. It is that logical validity makes no *intrinsic* claim on normativity.

5 Validity

At this point we expect to be met with a forceful objection. It may well be that if we permit the trivialization of validity in the way that proponents of the pluralist approach are powerless to prevent, then validity indeed has no intrinsic claim on normative legitimacy. But surely it could be claimed that there are cases of validity — not the trivialized validity of the pluralist manoeuvre, but cases which reputable logicians are prepared to see as the real thing — where a presumption of normative legitimacy is entirely in order.

We are not so sure.

In standard systems of deductive logic, e.g., monadic first order theory, validity is a decidable property. There are cases in which the would-be rational agent aims for validity in his reasoning. It is not hard to see why. Validity is truth-preserving; and there is little doubt that sometimes the rational thing to aim for is truth-preservation. On the other hand, there are cases in which truth-preservation is less than a desirable good, as we have said.

Truth-preservation is also enabled by argument-structures that are monotonic. Let $\Delta \models \Phi$ be any valid argument, with Φ the conclusion and Δ a (possibly empty) set of premisses. $\Delta \models \Phi$ is valid just in case any valuation that makes every $\alpha \in \Delta$ true also makes Φ true. Suppose now that we supplement Δ by arbitrarily selected propositions arbitrarily many times. Call this new set $\Delta \cup \Sigma$. Let v be any valuation on which all members of Δ are true only if Φ is true. Let v^* be any valuation making every member of $\Delta \cup \Sigma$ true. Then v is properly

included in v^*. Since v provides that $\Delta \models \Phi$ is valid, v^* provides that $\Delta \cup \Sigma \models \Phi$ is valid.

There is a sense, therefore, in which validity is a natural suppressor of enquiry. Once an argument has been determined to be valid, then from the point of view of truth-preservation there is nothing to be said for any further investigation. From that perspective, the distinction between admissible and inadmissible new information is lost, just as there is nothing to be gained by belief-updating. And given, below, that truth-preservation is your cognitive target, there is nothing to be said for belief-revision either.

What, then, justifies our enthusiasm for truth-preservation? Part of the answer lies in the high levels of risk aversion implicit in truth-preserving manoeuvres. Truth-preserving manoeuvres are mother's milk to error-avoidance, though this is not quite the word for it. What is intended is that valid reasoning is guaranteed to present the reasoner with conclusions that are no worse than the worst that can be said of the premiss-sets that imply them. If Δ is an inconsistent premiss-set, deducing Φ from it will not in general make for more inconsistency; and if the premiss-set contains a falsehood, deducing Φ from it won't in general land you with a new falsehood. But here, too, the empirical record suggests that human beings are not routinely committed to such strict routines of error-avoidance.

A third feature of truth-preservation should be mentioned. Truth-preservation is valuable only if there are truths to preserve. At the level of common sense, truths are liberally available. But no one with the slightest interest in the history of human cognitive performance will accept the congruence of what is true and what-is-taken-for-true by common sense. Validity does not preserve what-is-taken-for-true. What it preserves is truth. The value of truth-preservation is parasitic upon the of truth. So determining whether a piece of truth-preservation is valuable is no easier than determining whether a proposition you think is true *is* true.

Where do these points leave us? If we are happy with vacuous truth-preservation, validity is a standing invitation to admit new information indiscriminately and without adjustment. It is an attainable goal, but no one will think it a cognitive virtue. If, on the other hand, non-vacuous truth-preservation is what is wanted, it is as hard to pin-down as are the real truths that are the necessary inputs to the preservation mechanism. (In large ranges of cases these truths are precisely those whose digging up requires the larger resources of institutional agencies. But, in general, individuals lack sufficient command of such resources to make these targets possible, never mind economically attainable.)

This is not to say that a knowledge of classical consequence is not of use to the human reasoner. It helps knowing what follows from what when it comes to database updates and revisions. Logical consequence is the heart and soul of certain varieties of hypothetical reasoning. It is indispensable to a form of counterargument which Aristotle called "refutation." And so on. But where the consequence relation is classical consequence, knowledge of its provenance will in virtually any given case leave the performance of these tasks underdetermined [15, ch. 1 and 2].

6 *The* Actually Happens Rule

6.1 Bounded Rationality

We said two sections ago that the reflective equilibrium manoeuvre resembles an approach to normativity to which we ourselves are drawn. According to the reflective equilibrium approach the rightness or justifiability of a practice is *constituted* by the community standards

that it meets. What we find wanting in this view of the matter is that it leaves it unexplained as to how community standards manage to play so ontologically audacious a role.

We are here met with the ancient *Euthyphro*-problem in modern dress. The *Euthyphro* problem was a dilemma constructed by Socrates in the Platonic dialogue of the same name. It is commonly said that what is holy is what the gods decree. But, asks Socrates, do they command what they command because it is holy? (First horn.) Or is it holy because they command it? (Second horn.) If it is the first, the gods' endorsement has nothing to do with what it is to be holy. If the second, there is no prior constraint on what the gods say.

Once transformed to the question of reflective equilibrium, it is clear that of the two horns, the second is far the less attractive. It evacuates the notion of the holy of any stable content, apart from the utter contingencies of what the gods chance to think; equally it leaves the notion of rationality devoid of stable content, apart from the utter contingencies of what a community of would-be reasoners chances to think. Though not problem-free, the better option is the first horn. It kills the divine-command theory of the holy, which is hardly a theological or ethical disaster; and it kills the 'community-command' theory of rationality. What it leaves is the proposition that normativity inheres in how we act and behave, that normativity in descriptively immanent rather than transcendent.

What makes it plausible to suggest that community standards about rationality are indeed presumptively authoritative is that the behaviour that those standards call for is *already* presumptively adequate. Our view, then, is that as a first, defeasible pass, how cognitive agents actually do behave is (a substantial approximation to) how they should behave. This we call the *Actually Happens Rule*.

We note in passing two important discouragements of our position. One is the long history of loony beliefs held by beings like us. Leading the list is the monstrous *idée fixe* (e.g., Jews are subhuman; those who drown in the moat are innocent; etc.). We lack the time to examine this kind of case in the detail it deserves. Even so, we shouldn't just walk away from this point. We shall make a quick sketch of how we think that the problem could be handled. We note, then, that reasonableness (and also the more heavyweight notion of rationality) is definable both for *belief* and for *inference*. It is possible for beings like us to have among the premisses of inferences we conduct quite reasonably beliefs that aren't at all reasonable. We intend the Actually Happens Rule for the second half of the belief-inference distinction.

One the face of it, this is too swift. It appears that we have saved the Actually Happens Rule by stipulating that the cases that call it into question are cases that exceed its intended reach. Certainly no one would wish to see such exclusions raised to the status of a general procedure for disarming counterexamples. What is required, therefore, is provision of some independent reason for saying that the Actually Happens Rule is not as uniformly plausible a rule when applied to belief as it is when applied to inference. To show this, it will be necessary to say something of what we take rationality to be.

Ours is a resource-based approach to rationality. We see rational agents forming a hierarchy $\mathcal{H}$ partially ordered by the relation *commanding more cognitive resources than* in relation to *targets of greater strictness than*. Such resources include information, time and computational capacity. Individual (or equivalently in our usage) practical agents occupy $\mathcal{H}$'s upper reaches. Not only do beings like ourselves and entities such as NASA or Soviet aeroscience in the 1970s differ in agency type; they also differ in the kinds of standards they must hit in order that their respective cognitive behaviour qualify as satisfactory (competent, rational). Because individual or practical agents command fewer cognitive resources than e.g., NASA, their cognitive agendas must be discharged as economically as is consistent with a generally

satisfactory outcome.[5] It is natural therefore to postulate for agents of this type various strategies for cognitive economizing. We discuss these strategies in some detail in [7] and [45, ch. 1]. Here we will simply list two that bear most directly on the phenomenon of monstrous belief.

1. Reliance on *ad verecundiam* assurances.
2. Employment of *ad populum* reasoning.

Each of these instantiates in a central way the phenomenon of *epistemic trust*. Trust is clearly a factor at all levels of cognitive agency, but at the bottom end, where the need to economize is greatest, the reliance on trust is correspondingly greatest.

For individual or practical agents, belief-formation is accomplished in two main ways: (1) by reasoning from beliefs already possessed; and (2) by forming new beliefs on the basis of new inputs to the cognitive system. The empirical record attests to a generally good track record for practical agents in category (1); it is category (2) where the issue of monstrous belief more naturally arises.

In explaining the phenomenon of monstrous belief, it is well to emphasize that although there is no *à priori* limit on the stupid things an individual can come to believe all on his own, so to speak, the beliefs that have actually qualified in the category of the monstrous have typically met the following conditions.

1. They attained their standing in institutional settings. Thus, they are theorems of allegedly scientific theories, or the 'proofs' of theology, or the 'certainties' of other forms of ideology; and so on.
2. These institutionally sanctioned beliefs have been directed to contingencies further down the hierarchy $\mathcal{H}$ i.e., severally and collectively to individual agents.
3. In promulgating these beliefs down below, the promulgating institutional agents misappropriated cognitive devices (such as a willingness to accept the sayso of authorities, and a general inclination to favour "received opinions") that are constitutive aspects of the individual's rationality at $\mathcal{H}$'s ground zero.

A further condition should also be noted. By and large, the promulgation downward of monstrous beliefs constitute no direct threat to the life, limb or even prosperity of those individual agents who adopt them (although they can in particular cases, such as the Shoa, be catastrophically harmful to selected subgroups). They are, in one meaning of the word, *theoretical beliefs*, as befits their promulgation by institutional (i.e., theoretical) agents.

It has been well-said that it is not individuals who are stupid, but rather institutions [37]. Something of an overstatement, perhaps, but Whatmough's remark is in the spirit of what we are here proposing about the problem of monstrous belief on the part of wholly competent *reasoners*.[6]

Having developed this general position on resource-bound cognition, in which the individual or practical agent frequently must prosecute his cognitive agendas on the cheap, we have found it extremely interesting to discover a similar view in the psychological literature, under the description, "bounded rationality" (e.g. [32], and [11]). Seen this way, human cognizers are not psychobiological realizations of first order logic or classical probability theory or neoclassical economics; and so on. Successful cognition at the low level of $\mathcal{H}$ reflects the

[5] The notion of *cognitive economies* is subject to an emerging literature, building upon some important insights of Peirce [25]. Also important are Rescher [26, 27] and [28], Haskell [16], Goldman and Shaked [12], Hands [14], Sent [31] and Luetge [22]. See also, Gabbay and Woods [7, 8, 9, 10].

[6] The dynamics of belief-formation on the basis of institutional promotion is explored in greater detail in [38, 39, 40, 41].

disposition of the individual or practical agent to be a satisficer rather than an optimizer, as well as a kindred disposition to employ a "fast and frugal" methodology. This enables the agent to adapt to constant and even radical changes in his (or its) immediate environment, or, as robotics theorists sometimes say, which enable the agent to produce "online behaviour". One of the important consequences of this view — a consequence that comports with our own position — is that most of the irrationalities that experimental psychologists attribute to practical agents generically either aren't irrational or aren't instantiated in the actual behaviour of individuals. These include a disposition to commit the gambler's fallacy, regression-to-the-mean mistakes and covariation blunders, among others. A case in point is the idea that individual agents are particularly disposed to fallacious reasoning. We take issue with this. A fallacy is a mistake that appears not to be so. Over the centuries, fallacy theorists have made some headway with saying what mistaken reasoning might be; but by and large, they have been defeated by the requirement that they explain the phenomenon of being a mistake *that appears otherwise*. As we suggest in Gabbay and Woods [7], a resource-based approach to rationality can ameliorate this situation. Take hasty generalization as an example. At the low end of the hierarchy $\mathcal{H}$, hasty generalization is as commonplace as it is (approximately) successful. It is an essential tool in the cognitive economy of practical agents. Higher up, where resources are more abundant, and tasks are correspondingly more exacting, standards are appropriately higher. Here it is plausible to say that hasty generalization is as such a mistake. So, how is it that hasty generalization *is* a mistake yet *appears not to be* a mistake; i.e., is a fallacy? It is because it is a mistake (for NASA, say) yet appears not to be a mistake (since for individual agents it isn't as such a mistake at all).[7]

If, as we believe, we and bounded rationality theorists are right about this, then another source is embarrassment to the Actually Happens Rule is nicely removed.

6.2 Panglossism and Beyond

Still, it remains true that the *Actually Happens Rule* raises the question of the extent to which a logic sets normative standards for rational cognitive practice. Elsewhere we have said that this rule bears some affinity to a position in cognitive science called *panglossism* [8]. We should say something about this. (See here figure 1.)

Contemporary cognitive science marks a distinction among three different models of cognitive performance. They are the normative model N which specify sets standards of optional rational performance, irrespective of the costs — especially the computational costs — of compliance. The prescriptive model P attenuates these standards so as to make them computationally executable by beings like us. The descriptive model D gives a law-governed account of how beings like us actually perform. Following Stanovich [33], it is possible to discern three quite different positions concerning how the three models, N, P, and D, are linked. The principle of linkage is nearness to the goal of *good reasoning*. On the panglossian approach, there is little or nothing to distinguish N, P, and D in relation to the good reasoning goal. At the opposite pole is the "apologist" position in which N meets the goal and both P and D fail it, and do so both seriously and next-to-equally. The "meliorist" position takes up the middle position. N meets the goal. P fails it, but not so badly as to preclude *approximate* realization. D, on the other hand, fails it significantly.

It is not our intention to deal with the panglossian-meliorist-apologist rivalries at length.

[7] A more detailed *apologia* for hasty generalisation can be found in [10].

[8] Figure 1 is adapted from: [33, p. 5].

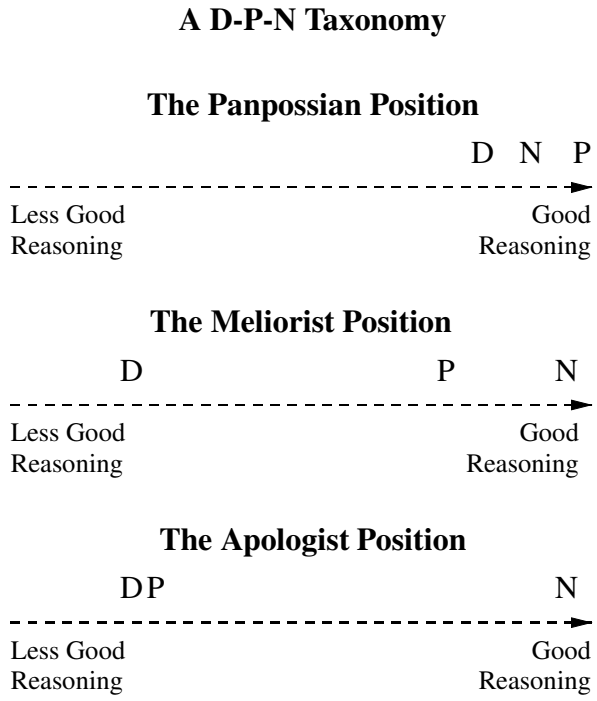


FIG. 1. Three pretheortical positions on human rationality[8]

If we were forced to choose among the three, we would opt for the panglossism. In fact, however, as previously indicated, we find ourselves attracted to a fourth position, which the panglossian position somewhat resembles, but which is also importantly different. Baldly stated, we are disposed to *reject* the normative model, and to *reject* its prescriptive submodel. This makes our own position vacuously panglossian: D reflects good reasoning rather well, and no other model reflects it better (since there are none). What so inclines us is the failure of those who espouse the N, P, D trichotomy to demonstrate that N provides objectively correct standards for optimal rationality and that P provides objectively correct standards for computationally realizable optimal rationality. (Sometimes called "optimization under constraint". See [34].[9])

We will not here debate this issue further (but again see, e.g., [43, ch. 8]). But perhaps an example would help explain our reluctance to accept validity as a dominant cognitive norm. As we have said, it is widely held that a system of logic is a normative model of good reasoning, because it contains provably sound rules for the valid derivation of conclusions from

[9] It is perhaps prudent to note that the optimization-under-constraint concept is sometimes identified with the idea of bounded rationality. (See again [34].) However, it is not bounded rationality in our sense or that of, e.g., Gigerenzer and Selten [11].

premisses (or databases). This presupposes that the hitting of validity-targets is invariably an instance of good reasoning. But we say again that in lots of situations valid deduction from premisses or a database is not the way in which good reasoning proceeds. This does not mean that, on occasion, valid deduction isn't part of what the good reasoner should aim for. For such cases, it is nothing but good to have a reliable account of what it is that is being aimed for. A system of logic may fill the bill by providing a good analysis of validity. Welcome though such an analysis would be, the logic that provided it would not be a plausible model for good reasoning.

But what if the reasoner's task were such as to *require* the use of deduction rules? Wouldn't a system of logic whose deduction rules were provably sound and complete be a convincing model of that particular sort of good reasoning? No. Good reasoning is always good in relation to a goal or an agenda (which may be tacit). Reasoning of the sort in question is good if it meets its goal or closes its agenda using only valid deduction rules. Reasoning validly is never *itself* a goal of good reasoning; otherwise one could always achieve the desired conclusion simply by repeating a premiss or by uttering an arbitrary necessary truth, such as "$p \vee \neg p$".

7 *The* Heuristic Fallacy

Before leaving this matter, it would be well to recur briefly to the reflective equilibrium manoeuvre. Let K be a theory of rational performance with respect to a certain skill, say K–ing.

The reflective equilibrium rationale can briefly be characterized as a balancing act between conservatism and innovation. Consider a proposed new K-principle. It should be rejected if it contradicts received opinion about K-rationality. Equally, consider a piece of K-behaviour. It should be rejected if it contradicts the established K-principles. But a new principle can be adopted if it attracts the requisite change in accepted K-behaviour.

The doctrine of reflective equilibrium pivots on the fact that the K-theorist, like everyone else, begins in *medias res*. He cannot proceed except on the basis of what he already believes about K-hood, and he cannot proceed unless what he initially believes about K-hood is also believed by others who are relevantly situated. The qualification "relevantly situated" is important. If K-theory is a relatively technical matter, then the relevantly situated others could be the community of researchers into K. Their beliefs, and our K-theorist's starting point, are a blend of individual K-judgements and sets of K-principles that are now in reflective equilibrium.

We accept this account as a descriptively adequate representation of how a K-theorist actually proceeds. We concede that at a practical level there is no other way for the K-theorist *to* proceed. These ways of proceedings are an indispensable heuristic for the would-be K-theorist. But it is a mistake to think that, because this is the way that he must go it must follow that the reflective equilibrium from which he does proceed is epistemically or normatively privileged. It is a mistake of a sort that we call the *Heuristic Fallacy*. It is the fallacy of supposing that any belief that is indispensable to the thinking up of a theory is a belief that must be formally *sanctioned* by the theory itself [42, 43, 8].

It is necessary to say a last word about ideal models. In the final Part of Gabbay and Woods [9], we develop formal models of abduction. As we see it, a formal model is an idealized description of what an abductive agent does. As such, it reflects some degree of departure from empirical accuracy. Thus an ideal model I is distinct from a descriptive model D. But

isn't this tantamount to a capitulation to normativity? It is not. Ideality is not normativity in our usage. Ideality is *simplification in support of systemacity*. The laws of frictionless surfaces are idealizations of the the slipperiness of real life. The laws that hold in the ideal model are approximated to by the pre-game ice at Maple Leaf Gardens. But no one thinks on that account that there is something *defective* about the Gardens' ice-conditions. We do not want to minimize the difficulty in getting at the right ideal models of K-behaviour. All that we say now is that it is no condition on the selection of ideal K-models that they hit pre-set standards tricked out in a presumed normative model N.[10]

8 Can Do and Make Do

We don't for a moment doubt the utter allure of the normative models methodology for the opinionated rationality theorist. We may expect that on occasion his enthusiasm for normative models will make him resistant to our blandishments. Witness here Keynes:

> *Progress* in economics consists almost entirely in a progressive improvement in the choice of models.... But it is of the essence of a model that one does *not* fill in real values for the variable functions. To do so would make it useless as a model.... The object of statistical study is not so much to fill in missing variables with a view to prediction, as to test the relevance and validity of the model [18, p. 296].

It would be good to have an understanding of the hold that the normative models approach has on theorists of the human condition. We bring this note to a close with a conjecture. Consider, first, what we call the

> *Can Do Principle:* If an investigator is working on a problem in a domain of enquiry D, and if results that are achievable in another domain D^* bear favourably on the first task, the D–theorist would do well to help achieve those results in D^*.

It is a good principle, liberally and manifestly at work in much of our best science. It is a fallible principle. We cannot always know in advance that the work done in D^* will indeed facilitate the work to be done in D. On occasion, it will become clear to the investigator that his results in D^* don't help in D. For example, an economist may discover that the mathematics of D^* are indiscernible in the behaviour of the actual economic entities that are the subject of D's investigation. Or a decision theorist may discover that theorems of the probability calculus are little evident in the business of real-life probabilistic reasoning. Consider now a certain range of such cases. They are cases in which the theorist doesn't know how to replace the results that he has very successfully produced in D^* with results that actually get the work in D done. In such cases, he may behave in ways that conform (perhaps tacitly) to what we call the *Make Do Principle*, which is a *corruption* of the *Can Do Principle*. It provides that the enquirer persist with his efforts in D^* simply because it is work that he knows how to do; and that he attempt to defend his doing so in one or other of three ways.

1. *Brazen it out.* Insist, without any attempt at demonstration, on the applicability of the D^*–work to his D–tasks.

[10] We acknowledge the considerable difficulty that attaches to defining a usable notion of approximation in contexts of behaviour that purports to be rational. There is a large literature on approximate truth [20, 24, 46]. It is not clear that this literature admits of a wholly straightforward adaptation to approximate rationality, notwithstanding some partial successor in such contexts as machine learning.

2. *Adopt a variation of Keynes' bluff.* Insist that the D^* results aren't really required to conform to or elucidate the phenomena of D.
3. *Seek refuge in ideal models.*

Concerning the first two of these options, we trust that there is obviously nothing good to say. Concerning the third, how to judge it has been the burden of this note to propose.

9 Appendix on Ideal Conditions

We have already made the point that some of the standard idealizations to be found in the belief-revision and decision theoretic literature are excessive. They outreach what an individual agent is able to do in ways that discourage the idea of even approximate conformity. If one takes a resource-bound approach to cognitive competence, there is little serious doubt that the standard idealizations go too far. But not everyone takes the resource-bound approach. Some people see it as the attempt to make a virtue of our cognitive limitations and shortfalls. It would therefore be helpful to the critic of the standard idealizations if there were some independent basis on which to fault them. For the past forty years just such a basis seems to have been furnished by mainline logic itself. But recent developments call this claim into question, leaving the bounded rationality theorist open to the objection that he does indeed seek to make cognitive virtues out of cognitive shortfalls.

Here, then, are the arguments against the standard idealizations.

1. Consider for example the following idealizing assumptions about belief.
   - C1. If X believes P then X believes that he believes that P. (The belief-transparency condition.)
   - C2. If X believes that P he believes that P is true. (The Convention T condition.)
   - C3. If P is a logical truth then X believes that P. (The logical truth condition.)
   - C4. If P is a deductive consequence of what X believes then X believes that P. (The closure under consequence condition.)
2. Montague [23] demonstrated that if the modalities are represented by a simple syntactic predicate (e.g., "is necessary" or "is known") and the above conditions are assumed for "is known", an inconsistency is provable.

   A leading diagnosis of this problem was that the idealizations were excessive.
3. In Thomason [35] it was shown that if these conditions hold for belief and belief is represented by a single syntactic predicate, then if X has consistent beliefs he cannot believe the truth of Robinson arithmetic.

   Here, too, it is natural to assume that the problem is that the idealizations are excessive.

Note, however, that this particular argument against the idealizations begins to slacken with the appearance of

4. Cross [4], in which it is demonstrated that a version of the Knower Paradox [17] is derivable *even if C4 (for knowledge) is suppressed*.

Worse,

5. In Cross [5], it is demonstrated that *even if C1–C4 are suspended*, and belief is represented by a syntactic predicate, then if X has consistent beliefs, he cannot believe Robinson arithmetic.

At this point it becomes natural to see the culprit as the syntactic treatment of belief, rather than the idealizations C1–C4. This being so, we must take care not to leave the impression that in rejecting the ideal model approach to normativity, we leave the general question of how to analyze belief and how to describe belief dynamics in a wholly unproblematic state.

Acknowledgements

This paper was originally presented under the title "Normative Models of Rational Agency" to the International Conference and Research Centre for Computer Science at the Dagstuhl Seminar on the Logic of Rational Agency in December 2001. We are grateful for helpful criticisms and suggestions to the following participants: Mike Wooldridge, Willem van der Hoek, J.-J. Meyer, Johan van Benthem, Barbara Grosz, Luke Hunsberger and John Bell. Subsequent versions of the paper have benefited from comments from David Makinson, Alice ter Meulen, Bas van Fraassen, Dale Jacquette, Nick Griffin, Gabriella Pigozzi, Odinaldo Rodrigues, and Howard Barringer. Research has been supported by the Engineering and Physical Science Research Council of the United Kingdom, The Social Science and Humanities Research Council of Canada, Professor Christopher Nicol, Dean of Arts and Science, University of Lethbridge and Professor Nancy Gallini, Dean of Arts, University of British Columbia.

References

[1] Jonathan Adler. *Belief's Own Ethics*. Cambridge, MA: MIT Press, 2002.

[2] Christopher Cherniak. *Minimal Rationality*. Cambridge, MA: MIT Press, 1986.

[3] L. Jonathan Cohen. Can human irrationality be experimentally demonstrated? *The Behavioral and Brain Sciences*, 4:317–331, 1981.

[4] Charles Cross. The paradox of the knower without epistemic closure. *Mind*, 110:319–333, 2001.

[5] Charles Cross. Syntactic treatments of nonidealized belief. *Synthese*, 129:335–341, 2001.

[6] Catherine Z. Elgin. *Considered Judgement*. Princeton, NJ: Princeton University Press, 1996.

[7] Dov M. Gabbay and John Woods. The new logic. *Logic Journal of the IGPL*, 9:157–190, 2001.

[8] Dov M. Gabbay and John Woods. *Agenda Relevance: A Study in Formal Pragmatics*. Volume one of the series *A Practical Logic of Cognitive Systems*. Amsterdam: North-Holland, 2003.

[9] Dov M. Gabbay and John Woods. *The Reach of Abduction*. Volume two of the series *A Practical Logic of Cognitive Systems*. Amsterdam: North-Holland, to appear, 2004.

[10] Dov M. Gabbay and John Woods. Fallacies as Cognitive Virtues, To appear, 2004.

[11] G. Gigerenzer and R. Selten, editors. *Bounded Rationality: The Adaptive Toolbox*. Cambridge, MA: MIT Press, 2001.

[12] A. Goldman and M. Shaked. An economic model of scientific activity and truth acquisition. *Philosophical Studies*, **63**, 31–55, 1991.

[13] Nelson Goodman. *Fact, Fiction and Forecast*. Indianapolis, IN: Bobbs-Merrill, 1965. Originally published in 1955.

[14] D. W. Hands. *A Reflection without Rules: Economic Mehtodology and Contermporary Science Theory*. Cambridge University Press, Cambridge, 2001.

[15] Gilbert Harman. *Change in View: Principles of Reasoning*. Cambridge, MA: MIT Press, 1986.

[16] T. L. Haskell. Professionalism versus Capitalism: R. H. Tawney, Emile Durkheim and C. S. Peirce on the Disinterestedness of Professional Communities. In T. L. Haskell, ed., *The Authority of Experts: Studies in History and Theory*, pp. 180–225. Indiana University Press, Bloomington, 1984.

[17] David Kaplan and Richard Montague. A paradox regained. *Notre Dame Journal of Formal Logic*, 1:79–90, 1960.

[18] John Maynard Keynes. Letter to R.F. Harrod, 4 July 1938. In Moggridge, editor, *Collected Writings of John Maynard Keynes: The General Theory and After: Part II: Defense and Development*. London, Macmillan, 1973.

[19] P. Kitcher. The division of cognitive labor. *Journal of Philosophy*, **87**, 5–22, 1990.

[20] T.A.G. Kuipers. *What is closer-to-the-truth? A Parade of Approaches to Truthlikeness*. Amsterdam: Rodopi, 1987. *Pozan Studies*, Volume 10.

[21] Richard Lewontin. Science and simplicity. *The New York Review of Books*, 50(7), May 2003.

[22] C. Luetge. Economics in Philosophy of Science, *Synthese*, to appear.

[23] Richard Montague. Syntactical treatments of modality, with corollaries on reflexion principles and finite axiomatizability. *Acta Philosophica Fennica*, 16:153–167, 1963.

[24] I. Niiniluoto. *Truthlikeness*. Dordrecht, Reidel, 1987.

[25] C. S. Perirce. Economy of research: original paper. In A. W. Burks, ed., *Collected Papers of C. S. Peirce*, volume 7, pp. 7.139–7.157, Harvard University Press, MA, 1931–58.

[26] N. Rescher. *Peirce's Philosophy of Science*. University of Notre Dame, Notre Dame, 1978.

[27] N. Rescher. *C ognitive Economy: The Economic Dimension of the Tehory of Knowledge*. University of Pittsburg Press, Pittsburgh, 1989.

[28] N. Rescher. *Priceless Knowledge? Natural Science in Economic Perspective*, Rowan and Littlefield, Lanham, MD, 1996.

[29] John Rawls. *A Theory of Justice*. Cambridge MA: Harvard University Press, 1971.

[30] R. A. Resnik. Seeing, sensing and scrutinizing. *Vision Research*, 40:1469–1487, 2000.

[31] E.-M. Sent. An Economist's glance at Goldman's economics. *Philosophy of Science*, **64**, S139–S148, 1997.

[32] Herbert A. Simon. *Models of Bounded Rationality: Behavioral Economics and Business Organization*, volume 2. Cambridge and London: The MIT Press, 1982.

[33] Keith A. Stanovich. *Who is Rational? Studies of Individual Differences in Reasoning*. Mahawah, NJ: Erlbaum, 1999.

[34] George J. Stigler. The economics of information. *The Journal of Political Economy*, LXIX(3):213–224, 1961.

[35] Richmond H. Thomason. A note on syntactical treatments of modality. *Synthese*, 44:391–395, 1980.

[36] Frans H. van Eemeren, Rob Grootendorst, Francisca Snoeck Henkemans, J. Anthony Blair, Ralph H. Johnson, Erik C.W. Krabbe, Christian Plantin, Douglas N. Walton, Charles A. Willard, John Woods, and David Zarefsky. *Fundamentals of Argumentation Theory: A Handbook of Historical Backgrounds and Contemporary Developments*. Mahwah, NJ: Erlbaum, 1996.

[37] Grant A. Whatmough. Money, machines, energy and wealth. In Kent Peacock, editor, *Living with the Earth: An Introduction to Environmental Philosophy*, pages 352–367. Toronto: Harcourt Brace, 1996.

[38] John Woods. Public policy and standoffs of force five. In E.M. Barth and E.C.W. Krabbe, editors, *Logic and Political Culture*, pages 97–108. Amsterdam: North-Holland, 1992. Reprinted in [44, ch. 12].

[39] John Woods. Deep disagreements and public demoralization. In Dov M. Gabbay and Hans Jürgen Ohlbach, editors, *Practical Reasoning: Springer Notes on Artificial Intelligence*, pages 650–662. Berlin: Springer, 1996. Reprinted in [44, ch. 13].

[40] John Woods. Privatizing death: Metaphysical discouragements of ethical thinking. In Peter A. French and Howard K. Wettstein, editors, *Midwest Studies in Philosophy*, volume XXIV, pages 199–217. Boston: Blackwell, 2000.

[41] John Woods. Slippery slopes and collapsing taboos. *Argumentation*, 4:107–134, 2000.

[42] John Woods. Pluralism about logical consequence: Conflict resolution in logic. In John Woods and Bryson Brown, editors, *Logical Consequences: Rival Approaches*, pages 39–54. Oxford: Hermes Scientific, 2001.

[43] John Woods. *Paradox and Paraconsistency: Conflict Resolution in the Abstract Sciences*. Cambridge and New York: Cambridge University Press, 2002.

[44] John Woods. *The Death of Argument: Fallaciers in Agent-based Reasoning*. Kluwer, Dordrecht and Boston, 2004.

[45] John Woods, Ralph H. Johnson, Dov M. Gabbay, and Hans Jurgen Ohlbach. Logic and the practical turn. In Dov M. Gabbay, Ralph H. Johnson, Hans Jurgen Ohlbach, and John Woods, editors, *Handbook of the Logic of Argument and Inference: The Turn Toward the Practical*. Amsterdam: North-Holland, 2002. Volume one in the series *Studies in Logic and Practical Reasoning*.

[46] S.D. Zwart. *Approach to Truth: Verisimilitude and Truthlikeness*. Amsterdam: Institute for Logic, Language and Computation, 1998.

Received November 2003

Kripke Completeness of First-Order Constructive Logics with Strong Negation

ICHIRO HASUO and RYO KASHIMA, *Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, O-okayama, Meguro, Tokyo 152-8552, Japan.*
E-mail: {hasuo2, kashima}@is.titech.ac.jp

Abstract

This paper considers Kripke completeness of Nelson's constructive predicate logic N3 and its several variants. N3 is an extension of intuitionistic predicate logic Int by an contructive negation operator $\sim$ called *strong negation*. The variants of N3 in consideration are by omitting the axiom $A \to (\sim A \to B)$, by adding the axiom of constant domain $\forall x(A(x) \vee B) \to \forall x A(x) \vee B$, by adding $(A \to B) \vee (B \to A)$, and by adding $\neg\neg(A \vee \sim A)$; the last one we would like to call *the axiom of potential omniscience* and can be interpreted that *we can eventually verify or falsify a statement, with proper additional information*. The proofs of completeness are by the widely-applicable *tree-sequent* method; however, for those logics with the axiom of potential omniscience we can hardly go through with a simple application of it. For them we present two different proofs: one is by an embedding of classical logic, and the other by the TSg method, which is an extension of the tree-sequent method.

Keywords: strong negation, quantified constructive logic, Kripke-type semantics, intermediate logics

1 Introduction

1.1 Strong negation

Strong (or *constructive*) *negation*, denoted by $\sim$ in this paper, is a negation operator in constructive logics introduced by Nelson [16], Markov [15], and von Kutschera [29].[1]

In intuitionistic logic, the negation operator $\neg A$, which we would like to call *Heyting's negation* to make it distinct from the strong one, is an abbreviation for $A \to \bot$, i.e. *A implies absurdity*. This kind of treatment of negation is justified by such a viewpoint as Grzegorczyk's one, "the compound sentences are not a product of experiment, they arise from reasoning. This concerns also negation: we see that the lemon is yellow, we do not see that it is not blue" [8].

However, such an example as Kracht cites yields an alternative definition of negation, especially for constructivists: "we can not only *verify* a simple proposition such as *This door is locked.* by direct inspection, but also *falsify* it" [13]. This motivates the idea of strong negation, that is, taking negative information as primitive as positive one.

Wansing [31] argues that constructive logics with strong negation and without some

[1] Markov refers to Nelson [16] in his short note from 1950, nevertheless it is claimed by many authors that they introduced strong negation independently.

of structural rules (in Gentzen-style sequent systems) can be candidates for the logic of *information structure*, due to the character of strong negation stated above. And in recent years, constructive logics have been paid attention to in the field of logic programming, e.g. [10], [19], [20], [30] and [1].

Nelson's constructive logic N3 is an extension of intuitionistic logic Int by the strong negation operator $\sim$, where the word *logic* designates a pair of a formal language and a set of its formulas which are admitted as theorems. $\sim$ is axiomatized in the Hilbert-style system as follows:

$$
\begin{array}{l}
A \to (\sim A \to B), \\
\sim(A \wedge B) \leftrightarrow \sim A \vee \sim B, \quad \sim(A \vee B) \leftrightarrow \sim A \wedge \sim B, \\
\sim(A \to B) \leftrightarrow A \wedge \sim B, \quad \sim\sim A \leftrightarrow A, \quad \sim\neg A \leftrightarrow A, \\
\sim\forall x A \leftrightarrow \exists x \sim A, \quad \sim\exists x A \leftrightarrow \forall x \sim A.
\end{array}
$$

Introduced by the axioms above, strong negation $\sim$ and Nelson's logic N3 enjoys several properties reflecting its motivation. In what follows we shall see three examples.

One is the principle of *constructible falsity*,

$$\vdash \sim(A \wedge B) \quad \text{iff} \quad \vdash \sim A \quad \text{or} \quad \vdash \sim B,$$

which can be regarded as the negative counterpart of *disjunction property*,

$$\vdash A \vee B \quad \text{iff} \quad \vdash A \quad \text{or} \quad \vdash B.$$

While the former does not hold as to Heyting's negation (i.e. in Int), it holds as to strong negation (i.e. in N3) by Kripke completeness of N3. This is one of the facts that imply that positive and negative information have equal importance in N3.

Another example is logical equivalence in N3. As is shown in [28], if logical equivalence in N3 (denoted by $A \cong_{\mathsf{N3}} B$ here) is defined by $\mathsf{N3} \vdash A \leftrightarrow B$ (just like in logics without $\sim$), then the equivalence theorem fails, i.e. $A \cong_{\mathsf{N3}} B$ does not necessarily imply $C[A] \cong_{\mathsf{N3}} C[B]$. However, if both $\mathsf{N3} \vdash A \leftrightarrow B$ and $\mathsf{N3} \vdash \sim A \leftrightarrow \sim B$ hold, that is, if A and B are equivalent in both positive and negative senses, then $\mathsf{N3} \vdash C[A] \leftrightarrow C[B]$.

The last is seen in the Kripke-type possible world semantics characterizing N3, which is an extension of that for Int by an extra interpretation I^-. I^- is the *falsum* interpretation, designating which formula is *falsified* at each possible world. In a Kripke-model for N3 (we would like to call it an N3-model), the falsity of an atomic formula is not reducible to the *verum* interpretation I^+, but can only be shown by I^-. About N3-models we shall see in detail later.

There have been many suggestions for an interpretation of intuitionistic logic Int: Kripke's one of *development of knowledge* [14], Grzegorczyk's one of *scientific research* [8], Brouwer-Heyting-Kolmogorov (BHK) interpretation in terms of *proofs* [11] [27], and so on. Those three cited above can be easily modified into interpretations of N3, as Wansing shows in [31].

1.2 The axiom of potential omniscience

Since N3 is a conservative extension of Int, it is natural to consider extensions of intermediate logics (which include Int and are included by classical logic Cl) by the

strong negation operator.[2] However, the study of extra axioms which are peculiar to logics with strong negation seems to have been paid less attention to. We will consider one of such axioms in this paper: $\neg\neg(A \vee {\sim}A)$, which we would like to call *the axiom of potential omniscience*. Intuitively it is interpreted that *we can eventually verify or falsify any statement, with proper additional information*. Now we shall see what can be implied by the axiom.

In the Kripke-type interpretation, the axiom of potential omniscience corresponds to the following statement: for every closed formula A and every state of information a, there is a state b which is reachable from a and where A is either verified or refuted. Hence the axiom, especially when combined with the axiom $(A \to B)\vee(B \to A)$ which can be interpreted that *the set of information states is linearly ordered*, seems useful to formalize such cases as some kind of games, e.g. cryptography; consider a game where one player (or a dealer, say Alice) knows the answer and the other player (say Bob) tries to find it. In such a game Bob can reach the correct answer if he obtains the information that Alice has.

The axiom may also be considered as one of the weaker versions of *the law of excluded middle*, $A \vee {\sim}A$.[3]

1.3 What is shown in this paper

This paper is to show Kripke completeness of several variants of Nelson's constructive logic N3, for the quantified case. The proofs are by the usage of a *tree-sequent*, which is a labelled tree each node of which is associated with a sequent. The tree-sequent method can be regarded as a kind of semantic tableaux, and since it aims at completeness as to Kripke-models of a tree-shape, a tree-sequent has its shape. Needless to say the method is also applicable to Kripke completeness of Int and some intermediate predicate logics; the results can be found in [12].

The logics whose Kripke completeness is proved in this paper are presented in Fig. 1, enclosed in a box.[4] For those four logics which are not enclosed, we shall see the difficulty therein in the last section.

Here the basic logics are N3 and N4: N4 is obtained from N3 by omitting the axiom $A \to ({\sim}A \to B)$, which is one of those which axiomatize $\sim$ and may be considered as the constructive version of *the ex falso rule* $\bot \to B$. In Kripke-type semantics for N3 the verum interpretation I^+ and the falsum one I^- are always disjoint, hence for each formula A and each possible world a we have one of the following three: A is *verified* at a, A is *falsified* at a, or A is *neither verified nor falsified* at a. For N4 where I^+ and I^- may intersect, we have another possibility: A is *both verified and falsified* at a. Hence the name of the logics; N4 is for *four-valued*. N4 is known to be *paraconsistent*.[5]

Each of the additional letters d, l, and o designates what follows:

---

[2] For example, Goranko [6], Sendlewski [22], [23], [24], and Kracht [13] discuss unquantified cases using algebraic methods.

[3] $\neg A \vee \neg\neg A$ is said to be *the weak law of excluded middle* and characterizes intermediate logics. Corsi and Ghilardi [4] considers Kripke completeness of Int plus the axiom and its extensions, for the quantified case.

[4] Almukdad and Nelson [2] consider $\neg$-free fragments of N3d, N4 and N4d (denoted by N^+, N^- and N^{+-} respectively) and gives Gentzen-style sequent systems for them.

[5] Priest and Routley [21] give an introduction to paraconsistent logics. See also Odintsov [17], Odintsov and Wansing [18]

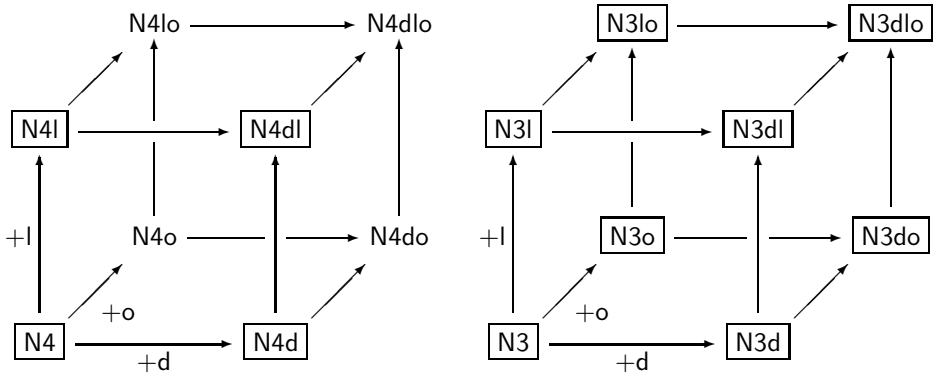


FIG. 1. The N-family

d Adding *the axiom of constant domain*, $\forall x(A(x) \vee B) \rightarrow \forall x A(x) \vee B$, where x have no free occurrences in B. The intermediate logic CD, which is Int plus this axiom, is characterized by the class of Kripke models whose domain is a constant map, as Görnemann [7] shows. d is for *Domain*.

l Adding the axiom $(A \rightarrow B) \vee (B \rightarrow A)$. Dummett's logic LC, which is Int plus this axiom, is characterized by the class of linearly ordered Kripke models, as Corsi [3] shows.[6] l is for *Linear*.

o Adding the axiom of potential omniscience, $\neg\neg(A \vee {\sim}A)$. o is for *Omniscience*.

The family of sixteen logics presented in the lattices above we would like to call the N-*family* in this paper.

While the completeness proofs for those without o are easy by simple applications of the tree-sequent method, the proofs for those with o are not, since the axiom of potential omniscience refers to, so to speak, upper-bounds of a Kripke model. For them we present two different proofs. One proof is applicable only to N3o and N3lo; it is by an embedding of Cl into N3o, and omniscient possible worlds (where every formula is either verified or falsified) are induced by Cl-models (or *structures*). The other is by an extension of the tree-sequent method which we would like to call the *tree-sequent with guardians* (abbreviated TSg) method.

There are some logics in the N-family whose Kripke completeness is already shown: N3 is by Gurevich [9] and by van Dalen [28], N3d is by Thomason [25], and N4 is by Odintsov and Wansing [18]. Nevertheless the authors do not take their own tree-sequent-based proofs as useless; the method used in them is applicable to other logics.

[6] Kripke completeness of CD or LC can be proved more easily using the tree-sequent method. The proof is just the same as that for N3d or N3l, and presented in [12].

1.4 Notations

In this paper we adopt Gentzen-style sequent systems for the formal presentation of logics;[7] a sequent system for logic L is denoted by **G**L. And later we will introduce tree-sequent systems and TSg systems for the proofs of Kripke completeness; the tree-sequent system for logic L is denoted by **T**L, and the TSg system is by **TgL**.

We often denote logics by such a form as $\mathsf{N}^3_4[\mathsf{d}]$: this is for "N3, N3d, N4 and N4d". N4d[o] is for "N4d and N4do".

We do not consider constants or function symbols, which makes the argument simpler without essential loss of generality.

Syntactical equivalence is denoted by $\equiv$. For example, $A \wedge B$ and $B \wedge A$ are logically equivalent in those logics considered in this paper; however, $A \wedge B \not\equiv B \wedge A$.

$A[y/x]$ is a substitution, obtained by replacing every free occurrences of x in A by y. It is not preferable that by substitution new bound variables come to existence; we avoid such cases by taking variants, i.e. replacing bound variables.

As in Tarski-type semantics for classical logic Cl, in defining $\models$ and its variants we will introduce temporary constants each of which designates a certain individual u. This kind of constant is said to be the *name* of u and denoted by $\underline{u}$.

For a finite set of formulas $\Gamma = \{A_1, \ldots, A_m\}$, $\bigwedge\Gamma$ (or $\bigvee\Gamma$) is an abbreviation for $A_1 \wedge \ldots \wedge A_m$ (or $A_1 \vee \ldots \vee A_m$). If $\Gamma = \emptyset$, it is $\top$ (or $\bot$), which is an abbreviation for $A \to A$ (or $\neg(A \to A)$, respectively).

2 Syntax and semantics

2.1 Gentzen-style sequent systems $\mathbf{G}\mathsf{N}^3_4[\mathsf{d}][\mathsf{I}][\mathsf{o}]$

Here we introduce Gentzen-style sequent systems for logics in the N-family. They share one formal language, consisting of the following symbols: countably many variables, $x_1, x_2, \ldots$; countably many m-ary predicate symbols for each $m \in \mathbb{N}$, p^m_1, p^m_2, $\ldots$; and logical connectives, $\wedge$, $\neg$, $\to$, $\sim$ and $\forall$. $\vee$ and $\exists$ are introduced as defined symbols:

$$A \vee B :\equiv \sim(\sim A \wedge \sim B), \qquad \exists x A :\equiv \sim\forall x \sim A.$$

$A \leftrightarrow B$ is an abbreviation of $(A \to B) \wedge (B \to A)$. Terms and formulas are composed in the same way as those of Cl, and note that $\sim$ is unary. The binding strength of the connectives are: $\forall, \neg, \sim \geq \wedge \geq \to$.

A *sequent* is defined as an ordered pair of finite sets of formulas separated by the symbol $\Rightarrow$, hence the rule of exchange or contraction can be omitted.

Now we present the initial sequents and derivation rules of a Gentzen-style sequent system **G**N3 for the logic N3:

$$\frac{}{A \Rightarrow A}\ \text{(Identity, Id)} \qquad \frac{}{A, \sim A \Rightarrow}\ \text{(Ex Falso, Fal)}$$

$$\frac{\Gamma \Rightarrow \Delta}{\Sigma, \Gamma \Rightarrow \Delta, \Pi}\ \text{(Weakening, W)} \qquad \frac{\Gamma \Rightarrow \Delta, A \quad A, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}\ \text{(Cut, C)}$$

[7] The duality of positive and negative information, syntactically A and $\sim A$, motivates another choice of formal presentation of N3: Wansing [32] introduces higher-arity Gentzen systems using four-place sequents, based on Belnap's display logic

$$\frac{A, B, \Gamma \Rightarrow \Delta}{A \wedge B, \Gamma \Rightarrow \Delta}\ (\wedge\mathrm{L}) \qquad \frac{\Gamma \Rightarrow \Delta, A \quad \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \wedge B}\ (\wedge\mathrm{R})$$

$$\frac{\Gamma \Rightarrow \Delta, A \quad B, \Gamma \Rightarrow \Delta}{A \to B, \Gamma \Rightarrow \Delta}\ (\to\mathrm{L}) \qquad \frac{A, \Gamma \Rightarrow B}{\Gamma \Rightarrow A \to B}\ (\to\mathrm{R})_{\mathrm{S}}$$

$$\frac{\Gamma \Rightarrow \Delta, A}{\neg A, \Gamma \Rightarrow \Delta}\ (\neg\mathrm{L}) \qquad \frac{A, \Gamma \Rightarrow}{\Gamma \Rightarrow \neg A}\ (\neg\mathrm{R})_{\mathrm{S}}$$

$$\frac{A[y/x], \Gamma \Rightarrow \Delta}{\forall x A, \Gamma \Rightarrow \Delta}\ (\forall\mathrm{L}) \qquad \frac{\Gamma \Rightarrow A[z/x]}{\Gamma \Rightarrow \forall x A}\ (\forall\mathrm{R})_{\mathrm{S},\ \mathrm{VC}}$$

$$\frac{\sim A, \Gamma \Rightarrow \Delta \quad \sim B, \Gamma \Rightarrow \Delta}{\sim(A \wedge B), \Gamma \Rightarrow \Delta}\ (\sim\wedge\mathrm{L}) \qquad \frac{\Gamma \Rightarrow \Delta, \sim A, \sim B}{\Gamma \Rightarrow \Delta, \sim(A \wedge B)}\ (\sim\wedge\mathrm{R})$$

$$\frac{A, \sim B, \Gamma \Rightarrow \Delta}{\sim(A \to B), \Gamma \Rightarrow \Delta}\ (\sim\to\mathrm{L}) \qquad \frac{\Gamma \Rightarrow \Delta, A \quad \Gamma \Rightarrow \Delta, \sim B}{\Gamma \Rightarrow \Delta, \sim(A \to B)}\ (\sim\to\mathrm{R})$$

$$\frac{A, \Gamma \Rightarrow \Delta}{\sim\neg A, \Gamma \Rightarrow \Delta}\ (\sim\neg\mathrm{L}) \qquad \frac{\Gamma \Rightarrow \Delta, A}{\Gamma \Rightarrow \Delta, \sim\neg A}\ (\sim\neg\mathrm{R})$$

$$\frac{A, \Gamma \Rightarrow \Delta}{\sim\sim A, \Gamma \Rightarrow \Delta}\ (\sim\sim\mathrm{L}) \qquad \frac{\Gamma \Rightarrow \Delta, A}{\Gamma \Rightarrow \Delta, \sim\sim A}\ (\sim\sim\mathrm{R})$$

$$\frac{\sim A[z/x], \Gamma \Rightarrow \Delta}{\sim\forall x A, \Gamma \Rightarrow \Delta}\ (\sim\forall\mathrm{L})_{\mathrm{VC}} \qquad \frac{\Gamma \Rightarrow \Delta, \sim A[y/x]}{\Gamma \Rightarrow \Delta, \sim\forall x A}\ (\sim\forall\mathrm{R})$$

Here the subscript S indicates the condition of no side formulas, that is, the succedent of the lower sequent must consist of only one formula, and VC the *eigenvariable condition*, i.e. the eigenvariable z must not have free occurrences in the lower sequent.

The initial sequents and derivation rules which do not involve $\sim$ precisely coincides with those of Maehara's $\mathbf{LJ}'$, which is equivalent to **LJ** and one of sequent systems for Int.

If a sequent $\Gamma \Rightarrow \Delta$ is derived using the initial sequents and rules above, $\Gamma \Rightarrow \Delta$ is *provable in* **GN3**, denoted by $\mathbf{GN3} \vdash \Gamma \Rightarrow \Delta$. A formula A is *provable in* **GN3**, denoted $\mathbf{GN3} \vdash A$, if $\mathbf{GN3} \vdash \Rightarrow A$.

For variants of N3,[8] we can obtain sequent systems by the following modifications:

N4 Omit the initial sequents (Fal).

d Allow side formulas in the rule ($\forall$R), resulting in the new rule

$$\frac{\Gamma \Rightarrow \Delta, A[z/x]}{\Gamma \Rightarrow \Delta, \forall x A}\ (\forall\mathrm{R})_{\mathrm{VC}}$$

This is equivalent to adding $\Rightarrow \forall x(A(x) \vee B) \to \forall x A(x) \vee B$ as initial sequents, as is well-known about the intermediate logic CD.

l Add the initial sequents

$$\frac{}{\Rightarrow (A \to B) \vee (B \to A)}\ (\mathrm{Li})$$

o Add the initial sequents

$$\frac{}{\Rightarrow \neg\neg(A \vee \sim A)}\ (\text{Potential Omniscience, Om})$$

There are a couple of equivalent additional rules; we shall see them later.

[8] When we speak of *variants of* N3 they include N4[d][l][o].

2.2 Kripke-type possible world semantics

Kripke-type semantics for logics in the N-family are extensions of that for Int, as stated in the previous section. First we shall describe the precise definitions, and then proceed to explain how the definitions are justified by our intuition.

Let $(M, \leq)$ be a poset, W a non-empty set, and U be a map of M into $\mathcal{P}W$, satisfying: $U(a) \neq \emptyset$ for all $a \in M$; $a \leq b$ implies $U(a) \subseteq U(b)$.

For every predicate symbol p (we assume p is m-ary), we define two interpretations of p at $a \in M$, denoted by $p^{I^+(a)}$ and $p^{I^-(a)}$, as subsets of $U(a)^m$, satisfying:

1. $a \leq b$ implies $p^{I^+(a)} \subseteq p^{I^+(b)}$ and $p^{I^-(a)} \subseteq p^{I^-(b)}$;
2. $p^{I^+(a)} \cap p^{I^-(a)} = \emptyset$.

Then the quintuple $\mathcal{M} = (M, \leq, U, I^+, I^-)$ is said to be an N3-*model*.

Given an N3-model $\mathcal{M}$, we extend two interpretations I^+ and I^- into two relations between an element in M and a closed formula, $a \models^+ A$ and $a \models^- A$, inductively on the construction of a closed formula A:

$$
\begin{aligned}
&a \models^+ p(\underline{u_1}, \ldots, \underline{u_m}) \quad \text{iff} \quad (u_1, \ldots, u_m) \in p^{I^+(a)} \ ;\\
&a \models^- p(\underline{u_1}, \ldots, \underline{u_m}) \quad \text{iff} \quad (u_1, \ldots, u_m) \in p^{I^-(a)} \ ;\\
&a \models^+ A \wedge B \quad \text{iff} \quad a \models^+ A \quad \text{and} \quad a \models^+ B \ ;\\
&a \models^- A \wedge B \quad \text{iff} \quad a \models^- A \quad \text{or} \quad a \models^- B \ ;\\
&a \models^+ A \to B \quad \text{iff} \quad \text{for every } b \geq a, \quad b \models^+ A \quad \text{implies} \quad b \models^+ B \ ;\\
&a \models^- A \to B \quad \text{iff} \quad a \models^+ A \quad \text{and} \quad a \models^- B \ ;\\
&a \models^+ \neg A \quad \text{iff} \quad \text{for every } b \geq a, \quad b \not\models^+ A \ ;\\
&a \models^- \neg A \quad \text{iff} \quad a \models^+ A \ ;\\
&a \models^+ {\sim}A \quad \text{iff} \quad a \models^- A \ ;\\
&a \models^- {\sim}A \quad \text{iff} \quad a \models^+ A \ ;\\
&a \models^+ \forall x A \quad \text{iff} \quad \text{for every } b \geq a \text{ and every } u \in U(b), \quad b \models^+ A[\underline{u}/x] \ ;\\
&a \models^- \forall x A \quad \text{iff} \quad \text{for some } u \in U(a), \quad a \models^- A[\underline{u}/x] \ .
\end{aligned}
$$

We write $a \not\models^{+-} A$ if neither $a \models^+ A$ nor $a \models^- A$ holds.

A formula A of N3 is *valid in an* N3-*model* $\mathcal{M}$, denoted by $\mathcal{M} \models A$, if $a \models^+ \forall \vec{x} A$ for every $a \in M$, where $\forall \vec{x} A$ is a universal closure of A. A is *valid*, denoted by $\mathsf{N3} \models A$, if A is valid in every N3-model.

A sequent $\Gamma \Rightarrow \Delta$ of **G**N3 is *valid* (or *valid in* $\mathcal{M}$), denoted by $\mathsf{N3} \models \Gamma \Rightarrow \Delta$ (or $\mathcal{M} \models \Gamma \Rightarrow \Delta$), if the formula $(\bigwedge \Gamma) \to (\bigvee \Delta)$ is valid (or valid in $\mathcal{M}$, respectively).

Now we describe how the definitions above can be understood.

The elements of M can be considered as "points in time (or 'evidential situations'), at which we may have various pieces of information" [14], and are often called *possible worlds*. Then the relation $a \leq b$ can be interpreted that a possible world a can develop into b.

$U(a)$ is the domain of individuals at a, more precisely, individuals whose existence is recognized at a. $p^{I^+(a)}$ (or $p^{I^-(a)}$) designates atomic formulas which are verified (or falsified) at a, by direct inspections. If $a \leq b$, then it is natural to assume that

b inherits all the information available at a, hence $U(a) \subseteq U(b)$, $p^{I^+(a)} \subseteq p^{I^+(b)}$ and $p^{I^-(a)} \subseteq p^{I^-(b)}$.

$a \models^+ A$, $a \models^- A$ or $a \not\models^{+-} A$ is interpreted that a *verifies*, *falsifies* or *neither verifies nor falsifies* A. While whether Heyting's negation holds is reduced (through *reasoning*) to *verum* interpretation I^+, strong negation is reduced to *falsum* interpretation I^-, which is primitive and independent of I^+. We can see the constructive character of $\mathsf{N3}$-models in I^- and $\models^-$, comparing with Int-models.

Note that $(M, \leq)$ need not be linearly-ordered; a may develop into different possible worlds.

Kripke-type semantics for the logics other than $\mathsf{N3}$ is obtained by the following modifications:

$\boxed{\mathsf{N4}}$ Omit the condition $p^{I^+(a)} \cap p^{I^-(a)} = \emptyset$.

$\boxed{\mathsf{d}}$ Add the condition that the domain $U : M \to \mathcal{P}W$ is a constant map.

$\boxed{\mathsf{l}}$ Add the condition that $(M, \leq)$ is linearly ordered, that is, for every $a, b \in M$ either $a \leq b$ or $b \leq a$ holds.

$\boxed{\mathsf{o}}$ Add the condition that for every $a \in M$ and every closed formula A, there exists $b \geq a$ where either $b \models^+ A$ or $b \models^- A$ holds.

In fact completeness shown later is often for smaller class of models; first, we can assume a set of possible worlds $(M, \leq)$ to be a tree, since every model $\mathcal{M}$ can be transformed into an equivalent tree-shape model $\mathcal{M}^t$ by taking each path of $\mathcal{M}$ as a node of $\mathcal{M}^t$. Second, the additional condition for o shall be strengthened as follows: for every $a \in M$, there exists $a_g \geq a$ (we will often call a_g the *guardian* of a) which is *omniscient*, that is, for every predicate symbol p (which is m-ary), $p^{I^+(a_g)} \cup p^{I^-(a_g)} = U(a_g)^m$. Moreover, such a_g shall be restricted to be a node which is an immediate successor of a and maximal in $(M, \leq)$ for $\mathsf{N3o}$ and $\mathsf{N3do}$, or restricted to be the maximum of $(M, \leq)$ for $\mathsf{N3lo}$ and $\mathsf{N3dlo}$.

Several examples of models for logics in the N-family are presented in Fig. 2, where a circle is a possible world, an arrow from a to b denotes that $a \leq b$, the upper colored part of a circle designates the closed formulas verified there, and the lower one those fasified.

The following lemmas which are true to our intuition are easily obtained by induction.

Lemma 2.1 (Heredity) Let A a closed formula of $\mathsf{N3}$ (or equivalently $\mathsf{N4^3_4[d][l][o]}$), $\mathcal{M} = (M, \leq, U, I^+, I^-)$ be an $\mathsf{N4^3_4[d][l][o]}$-model, $a, b \in M$ and $a \leq b$. Then $a \models^+ A$ (or $a \models^- A$) implies $b \models^+ A$ (or $b \models^- A$, respectively).

Lemma 2.2 Let A a closed formula of $\mathsf{N3}$, $\mathcal{M} = (M, \leq, U, I^+, I^-)$ be an $\mathsf{N3[d][l][o]}$-model and $a \in M$. Then it is impossible that both $a \models^+ A$ and $a \models^- A$ hold.

Soundness of the sequent systems introduced in the previous section is again easily obtained by induction on the derivation.

Theorem 2.3 (Kripke soundness of $\mathbf{GN4^3_4[d][l][o]}$) If $\mathbf{GN4^3_4[d][l][o]} \vdash \Gamma \Rightarrow \Delta$, then $\mathsf{N4^3_4[d][l][o]} \models \Gamma \Rightarrow \Delta$, respectively.

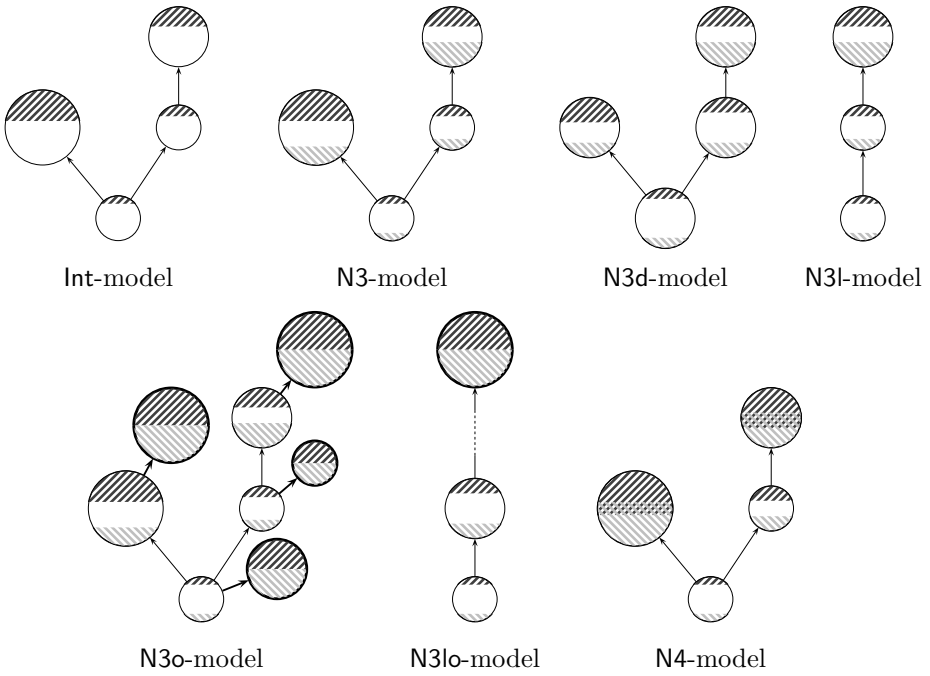


FIG. 2. Kripke models

3 Kripke completeness of $\mathbf{GN}^3_4[\mathsf{d}][\mathsf{l}]$

3.1 Tree-sequent system $\mathbf{TN}^3_4[\mathsf{d}][\mathsf{l}]$

In this section we present proofs of Kripke completeness of eight logics, $\mathsf{N}^3_4[\mathsf{d}][\mathsf{l}]$, which are by the usage of the tree-sequent method. In fact, the proofs are just the same as that for Int, CD or LC. In what follows, the word *tree-sequent* is often abbreviated as *TS*.

First we introduce the TS systems $\mathbf{TN}^3_4[\mathsf{d}][\mathsf{l}]$. Recall that $\mathbf{TN}^3_4$ is for "**TN3** and **TN4**".

Definition 3.1 (Tree-sequent of $\mathbf{TN}^3_4$) A *tree-sequent* $\mathcal{T}$ of $\mathbf{TN}^3_4$ is a finite labelled tree, each node a of which is associated with a sequent $\Gamma_a \Rightarrow \Delta_a$ of **GN3** (or equivalently of $\mathbf{GN}^3_4[\mathsf{d}][\mathsf{l}]$) and a finite set of variables α_a, denoted by $(a : \Gamma_a \overset{\alpha_a}{\Rightarrow} \Delta_a)$, satisfying the following conditions:

1. Let 0 be the root of $\mathcal{T}$, and $a_0(=0), a_1, \ldots, a_n$ be an arbitrary path in $\mathcal{T}$, and $(a_i : \Gamma_i \overset{\alpha_i}{\Rightarrow} \Delta_i)$ for each $i \in [0, n]$. Then $\alpha_0, \alpha_1, \ldots, \alpha_n$ are disjoint. The (disjoint) union $\alpha_0 \cup \alpha_1 \cup \cdots \cup \alpha_n$ is said to be the set of *available variables* at the node a_n.
2. Every variable which has free occurrences in the sequent associated to a is available at a.

In other words, for $(a : \Gamma_a \overset{\alpha_a}{\Rightarrow} \Delta_a)$ α_a is the set of variables which are available at a

for the first time in tracing from the root.

Definition 3.2 (Pre-tree-sequent of $\mathbf{TN}^3_4$) Omitting the condition 2, we obtain the definition of *pre-tree-sequent*, abbreviated as pTS, of $\mathbf{TN}^3_4$.

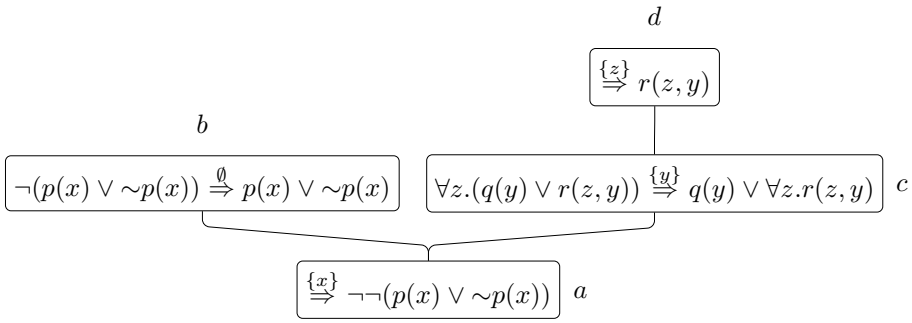
pTS's emerge as subtrees of TS's.

Definition 3.3 (Tree-sequent of $\mathbf{TN}^3_4\mathsf{d}$) A *tree-sequent* $\mathcal{T}$ of $\mathbf{TN}^3_4\mathsf{d}$ is simply a finite labelled tree each node of which is associated with a sequent, just like $(a : \Gamma_a \Rightarrow \Delta_a)$. The definition does not concern availability of variables.

Definition 3.4 (Tree-sequent and pre-tree-sequent of $\mathbf{TN}^3_4[\mathsf{d}]\mathsf{I}$) A *tree-sequent* (or *pre-tree-sequent*) of $\mathbf{TN}^3_4[\mathsf{d}]\mathsf{I}$ is that of $\mathbf{TN}^3_4[\mathsf{d}]$ which is linearly ordered, i.e. each node of which has at most one successor.

As seen in the following proof of compleness, the set of available variables works as a seed of the domain at the node; hence the concept is unnecessary for the case of constant domain models, i.e. logics with d.

Here is an example of a TS of $\mathbf{TN}^3_4$:

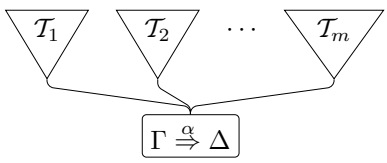


where the sets of available variables at the node a, b, c and d are $\{x\}$, $\{x\}$, $\{x, y\}$ and $\{x, y, z\}$, respectively.

In what follows, we adopt the notation below of a TS, for economy of space:

$$\Big[\ \overset{\{x\}}{\Rightarrow} \neg\neg(p(x) \vee \sim p(x))\ \mid\ \big[\ \neg(p(x) \vee \sim p(x)) \overset{\emptyset}{\Rightarrow} p(x) \vee \sim p(x)\ \big]$$
$$\big[\ \forall z.(q(y) \vee r(z,y)) \overset{\{y\}}{\Rightarrow} q(y) \vee \forall z.r(z,y)\ \mid \overset{\{z\}}{\Rightarrow} r(z,y)\ \big]\ \Big],$$

that is,



is denoted by $[\Gamma \overset{\alpha}{\Rightarrow} \Delta \mid \mathcal{T}_1 \ldots \mathcal{T}_m]$.

However, this notation hardly clarifies the structure of a TS; it will be helpful for to rewrite it in the tree-style.

Since a TS of $\mathbf{TN}^3_4[\mathsf{d}]\mathsf{I}$ is a linearly ordered finite tree, which is nothing but a finite sequence, we often omit [and] in its notation. Hence it is denoted in the form

$$\Gamma_1 \overset{\alpha_1}{\Rightarrow} \Delta_1 \mid \Gamma_2 \overset{\alpha_2}{\Rightarrow} \Delta_2 \mid \cdots \mid \Gamma_k \overset{\alpha_k}{\Rightarrow} \Delta_k.$$

Definition 3.5 (Tree-sequent system TN) Initial tree-sequents and derivation rules of **TN** are presented below, followed by some remarks which describe what they designate.

$$\frac{}{\cdots[A \overset{\alpha}{\Rightarrow} A \mid \cdots} \text{(Id)} \qquad \frac{}{\cdots[A, {\sim}A \overset{\alpha}{\Rightarrow} \mid \cdots} \text{(Fal)}$$

$$\frac{\cdots[\Gamma \overset{\alpha}{\Rightarrow} \Delta \mid \cdots}{\cdots[\Sigma, \Gamma \overset{\alpha}{\Rightarrow} \Delta, \Pi \mid \cdots} \text{(W)} \qquad \frac{\cdots[\Gamma \overset{\alpha}{\Rightarrow} \Delta \mid \cdots[A, \Sigma \overset{\beta}{\Rightarrow} \Pi \mid \cdots]\cdots]\cdots}{\cdots[A, \Gamma \overset{\alpha}{\Rightarrow} \Delta \mid \cdots[\Sigma \overset{\beta}{\Rightarrow} \Pi \mid \cdots]\cdots]\cdots} \text{(Drop, D)}$$

$$\frac{\cdots[A, B, \Gamma \overset{\alpha}{\Rightarrow} \Delta \mid \cdots}{\cdots[A \wedge B, \Gamma \overset{\alpha}{\Rightarrow} \Delta \mid \cdots} (\wedge\text{L}) \qquad \frac{\cdots[\Gamma \overset{\alpha}{\Rightarrow} \Delta, A \mid \cdots \quad \cdots[\Gamma \overset{\alpha}{\Rightarrow} \Delta, B \mid \cdots}{\cdots[\Gamma \overset{\alpha}{\Rightarrow} \Delta, A \wedge B \mid \cdots} (\wedge\text{R})$$

$$\frac{\cdots[\Gamma \overset{\alpha}{\Rightarrow} \Delta, A \mid \cdots \quad \cdots[B, \Gamma \overset{\alpha}{\Rightarrow} \Delta \mid \cdots}{\cdots[A \to B, \Gamma \overset{\alpha}{\Rightarrow} \Delta \mid \cdots} (\to\text{L}) \qquad \frac{\cdots[\Gamma \overset{\alpha}{\Rightarrow} \Delta \mid \cdots[A \overset{\emptyset}{\Rightarrow} B]\cdots]\cdots}{\cdots[\Gamma \overset{\alpha}{\Rightarrow} \Delta, A \to B \mid \cdots} (\to\text{R})_{\text{T}}$$

$$\frac{\cdots[\Gamma \overset{\alpha}{\Rightarrow} \Delta, A \mid \cdots}{\cdots[\neg A, \Gamma \overset{\alpha}{\Rightarrow} \Delta \mid \cdots} (\neg\text{L}) \qquad \frac{\cdots[\Gamma \overset{\alpha}{\Rightarrow} \Delta \mid \cdots[A \overset{\emptyset}{\Rightarrow}]\cdots]\cdots}{\cdots[\Gamma \overset{\alpha}{\Rightarrow} \Delta, \neg A \mid \cdots} (\neg\text{R})_{\text{T}}$$

$$\frac{\cdots[A[y/x], \Gamma \overset{\alpha}{\Rightarrow} \Delta \mid \cdots}{\cdots[\forall x A, \Gamma \overset{\alpha}{\Rightarrow} \Delta \mid \cdots} (\forall\text{L}) \qquad \frac{\cdots[\Gamma \overset{\alpha}{\Rightarrow} \Delta \mid \cdots[\ \overset{\{z\}}{\Rightarrow} A[z/x]\ ]\cdots]\cdots}{\cdots[\Gamma \overset{\alpha}{\Rightarrow} \Delta, \forall x A \mid \cdots} (\forall\text{R})_{\text{T}}$$

$$\frac{\cdots[{\sim}A, \Gamma \overset{\alpha}{\Rightarrow} \Delta \mid \cdots \quad \cdots[{\sim}B, \Gamma \overset{\alpha}{\Rightarrow} \Delta \mid \cdots}{\cdots[{\sim}(A \wedge B), \Gamma \overset{\alpha}{\Rightarrow} \Delta \mid \cdots} (\sim\wedge\text{L}) \qquad \frac{\cdots[\Gamma \overset{\alpha}{\Rightarrow} \Delta, {\sim}A, {\sim}B \mid \cdots}{\cdots[\Gamma \overset{\alpha}{\Rightarrow} \Delta, {\sim}(A \wedge B) \mid \cdots} (\sim\wedge\text{R})$$

$$\frac{\cdots[A, {\sim}B, \Gamma \overset{\alpha}{\Rightarrow} \Delta \mid \cdots}{\cdots[{\sim}(A \to B), \Gamma \overset{\alpha}{\Rightarrow} \Delta \mid \cdots} (\sim\to\text{L}) \qquad \frac{\cdots[\Gamma \overset{\alpha}{\Rightarrow} \Delta, A \mid \cdots \quad \cdots[\Gamma \overset{\alpha}{\Rightarrow} \Delta, {\sim}B \mid \cdots}{\cdots[\Gamma \overset{\alpha}{\Rightarrow} \Delta, {\sim}(A \to B) \mid \cdots} (\sim\to\text{R})$$

$$\frac{\cdots[A, \Gamma \overset{\alpha}{\Rightarrow} \Delta \mid \cdots}{\cdots[{\sim}\neg A, \Gamma \overset{\alpha}{\Rightarrow} \Delta \mid \cdots} (\sim\neg\text{L}) \qquad \frac{\cdots[\Gamma \overset{\alpha}{\Rightarrow} \Delta, A \mid \cdots}{\cdots[\Gamma \overset{\alpha}{\Rightarrow} \Delta, {\sim}\neg A \mid \cdots} (\sim\neg\text{R})$$

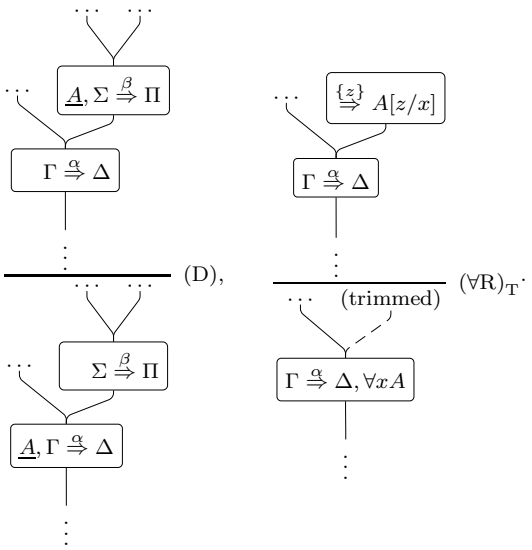
$$\frac{\cdots[A, \Gamma \overset{\alpha}{\Rightarrow} \Delta \mid \cdots}{\cdots[{\sim}\sim A, \Gamma \overset{\alpha}{\Rightarrow} \Delta \mid \cdots} (\sim\sim\text{L}) \qquad \frac{\cdots[\Gamma \overset{\alpha}{\Rightarrow} \Delta, A \mid \cdots}{\cdots[\Gamma \overset{\alpha}{\Rightarrow} \Delta, {\sim}\sim A \mid \cdots} (\sim\sim\text{R})$$

$$\frac{\cdots[{\sim}A[z/x], \Gamma \overset{\alpha\cup\{z\}}{\Rightarrow} \Delta \mid \cdots}{\cdots[{\sim}\forall x A, \Gamma \overset{\alpha}{\Rightarrow} \Delta \mid \cdots} (\sim\forall\text{L})_{\text{VC}} \qquad \frac{\cdots[\Gamma \overset{\alpha}{\Rightarrow} \Delta, {\sim}A[y/x] \mid \cdots}{\cdots[\Gamma \overset{\alpha}{\Rightarrow} \Delta, {\sim}\forall x A \mid \cdots} (\sim\forall\text{R})$$

In the above, a tree-structure and undisplayed nodes are arbitrary; (Id) can be read as every TS which has a node of the form $A \overset{\alpha}{\Rightarrow} A$ is an initial TS.

In the derivation rules, tree-structures and undisplayed nodes are the same between the conclusion and the hypothesis (or hypotheses); (W) is adding formulas to one node of the upper TS.

The rules which are unique to TS's, namely the structural rule Drop (D) and the logical rules with the subscript T, are displayed in the tree-style:



In (D), a formula A in the antecedent of a node is dropped to that of its immedeate successor. In $(\forall \mathrm{R})_{\mathrm{T}}$ a leaf (i.e. a maximal node) $\overset{\{z\}}{\Rightarrow} A[z/x]$ is *trimmed*. Note that the formula $\forall x A$ must obtain its occurrence in the immediate successor of the trimmed leaf, and that the variable z has no free occurrence in $\Gamma \overset{\alpha}{\Rightarrow} \Delta$ since z is available for the first time at $\overset{\{z\}}{\Rightarrow} A[z/x]$. The former remark is also true of the rules $(\neg \mathrm{R})_{\mathrm{T}}$ and $(\rightarrow \mathrm{R})_{\mathrm{T}}$.

In the rule $(\sim \forall \mathrm{L})_{\mathrm{VC}}$, VC is for the condition that the eigenvariable z have no free occurrences in *any* node of the upper TS.

It is seen that except (D) and those rules with the subscript T (which correspond to the rules of **GN** with the subscript S), the rules are of the form that the corresponding rules of **GN** are applied to one node of the upper TS (or TS's).

$\mathbf{TN}^3_4[\mathsf{d}][\mathsf{l}]$ enjoy one remarkable property: they are cut-free. Although it may imply some other proof-theoretical results, they are not considered in this paper.

Definition 3.6 (Tree-sequent system $\mathbf{TN}^3_4[\mathsf{d}][\mathsf{l}]$) Tree-sequent systems $\mathbf{TN}^3_4[\mathsf{d}][\mathsf{l}]$ for the logics $\mathsf{N}^3_4[\mathsf{d}][\mathsf{l}]$ are obtained from **TN3** through the following modifications:

4 Omit the initial TS's (Fal).

d TS's do not involve the concept of availability of variables, as stated above. And the rule $(\forall \mathrm{R})_{\mathrm{T}}$ is replaced by the new rule:

$$\frac{\cdots [\Gamma \Rightarrow \Delta, A[z/x] \mid \cdots}{\cdots [\Gamma \Rightarrow \Delta, \forall x A \mid \cdots} \ (\forall \mathrm{R})_{\mathrm{VC}}.$$

l Those rules with subscript T are replaced. $(\forall \mathrm{R})_{\mathrm{T}}$ is replaced by

$$\frac{\mathcal{T}_1 \quad \mathcal{T}_2 \quad \cdots \quad \mathcal{T}_k}{\cdots \mid \Gamma_1 \overset{\alpha_1}{\Rightarrow} \Delta_1, \forall x A \mid \Gamma_2 \overset{\alpha_2}{\Rightarrow} \Delta_2 \mid \cdots \mid \Gamma_k \overset{\alpha_k}{\Rightarrow} \Delta_k} \ (\forall \mathrm{R})_{\mathrm{K}}$$

where

$$\begin{aligned}
\mathcal{T}_1 \equiv\quad & \cdots \mid \Gamma_1 \overset{\alpha_1}{\Rightarrow} \Delta_1 \mid \quad \overset{\{z\}}{\Rightarrow} A[z/x] \mid \Gamma_2 \overset{\alpha_2}{\Rightarrow} \Delta_2 \mid \cdots \mid \Gamma_k \overset{\alpha_k}{\Rightarrow} \Delta_k,\\
\mathcal{T}_2 \equiv\quad & \cdots \mid \Gamma_1 \overset{\alpha_1}{\Rightarrow} \Delta_1 \mid \Gamma_2 \overset{\alpha_2}{\Rightarrow} \Delta_2 \mid \quad \overset{\{z\}}{\Rightarrow} A[z/x] \mid \cdots \mid \Gamma_k \overset{\alpha_k}{\Rightarrow} \Delta_k,\\
&\cdots,\\
\mathcal{T}_k \equiv\quad & \cdots \mid \Gamma_1 \overset{\alpha_1}{\Rightarrow} \Delta_1 \mid \Gamma_2 \overset{\alpha_2}{\Rightarrow} \Delta_2 \mid \cdots \mid \Gamma_k \overset{\alpha_k}{\Rightarrow} \Delta_k \mid \quad \overset{\{z\}}{\Rightarrow} A[z/x].
\end{aligned}$$

The subscript K is for *sKip*. Note that the eigenvariable z has no free occurrences in the lower TS, because of the shape of $\mathcal{T}_k$ and the definition of a TS. $(\to\mathrm{R})_\mathrm{T}$ and $(\neg\mathrm{R})_\mathrm{T}$ are also replaced by $(\to\mathrm{R})_\mathrm{K}$ and $(\neg\mathrm{R})_\mathrm{K}$ respectively, which are of the same form as $(\forall\mathrm{R})_\mathrm{K}$. For example in $(\to\mathrm{R})_\mathrm{K}$,

$$\mathcal{T}_i \equiv\quad \cdots \mid \Gamma_1 \overset{\alpha_1}{\Rightarrow} \Delta_1 \mid \cdots \mid \Gamma_i \overset{\alpha_i}{\Rightarrow} \Delta_i \mid A \overset{\emptyset}{\Rightarrow} B \mid \cdots \mid \Gamma_k \overset{\alpha_k}{\Rightarrow} \Delta_k.$$

In $\mathbf{TN}^3_4\mathsf{dl}$, the rule $(\forall\mathrm{R})_\mathrm{VC}$ remains the same as that of $\mathbf{TN}^3_4\mathsf{d}$.

3.2 Proof of Kripke completeness

The proofs of Kripke completeness of $\mathbf{GN}^3_4[\mathsf{d}][\mathsf{l}]$ are by a simple application of the tree-sequent method. The sketch is as follows. First we prove Kripke completeness of the corresponding TS system; given an unprovable TS $\mathcal{T}$ (i.e. $\mathbf{TN}^3_4[\mathsf{d}][\mathsf{l}] \nvdash \mathcal{T}$), we can extend $\mathcal{T}$ into a $\mathbf{TN}^3_4[\mathsf{d}][\mathsf{l}]$-*saturated* (possibly infinite) tree-sequent $\mathcal{T}_\omega$, which induces a *counter* $\mathsf{N}^3_4[\mathsf{d}][\mathsf{l}]$-*model* for $\mathcal{T}$. Then we define the *formulaic translation* of a TS $\mathcal{T}$, which we denote by $\mathcal{T}^f$. With the lemma that $\mathbf{TN}^3_4[\mathsf{d}][\mathsf{l}] \vdash \mathcal{T}$ implies $\mathbf{GN}^3_4[\mathsf{d}][\mathsf{l}] \vdash \mathcal{T}^f$ and Kripke completeness of $\mathbf{TN}^3_4[\mathsf{d}][\mathsf{l}]$, we can conclude that if $\mathbf{GN}^3_4[\mathsf{d}][\mathsf{l}] \nvdash A$ then the TS $[\overset{\alpha}{\Rightarrow} A]$ has a counter $\mathsf{N}^3_4[\mathsf{d}][\mathsf{l}]$-model, hence Kripke completeness of $\mathbf{GN}^3_4[\mathsf{d}][\mathsf{l}]$.

Definition 3.7 (Counter model for tree-sequent) Let $\mathcal{M} = (M, \leq, U, I^+, I^-)$ be an $\mathsf{N}^3_4[\mathsf{d}][\mathsf{l}]$-model, $U : M \to \mathcal{P}W$, and $\mathcal{T}$ a TS of $\mathbf{TN}^3_4[\mathsf{d}][\mathsf{l}]$. $\mathcal{M}$ is said to be a *counter model for* $\mathcal{T}$ if:

1. the tree-structure of $\mathcal{T}$ can be embedded in $(M, \leq)$;
2. the set of all variables can be embedded in W;
3. for each node $(a : \Gamma_a \overset{\alpha_a}{\Rightarrow} \Delta_a)$ of $\mathcal{T}$, $a \models^+ A[\vec{x}/\vec{x}]$ for each $A \in \Gamma_a$, and $a \not\models^+ B[\vec{y}/\vec{y}]$ for each $B \in \Delta_a$, where $\vec{x}$ or $\vec{y}$ is an enumeration of the free variables in A or B.

If $\mathcal{T}$ has no counter models, then $\mathcal{T}$ is said to be *unrefutable*.

It is easily seen that every TS derived in $\mathbf{TN}^3_4[\mathsf{d}][\mathsf{l}]$ is unrefutable, which justifies such unfamiliar rules as (D), $(\forall\mathrm{R})_\mathrm{T}$ and $(\forall\mathrm{R})_\mathrm{K}$.

An *infinite tree-sequent* is a TS whose tree-structure, associating sequents, and associating sets of variables are all possibly infinite, satisfying the condition on availability of variables.

$\mathbf{TN}^3_4[\mathsf{d}][\mathsf{l}]$-*saturatedness* is introduced as a natural extension of **LK**-saturatedness. Here **LK** is meant to be a sequent system used to formalize classical logic.[9] The con-

[9] To be precise, **LK** is meant to be the system **G1c** in [26], with the definition of a sequent modified to be just as in this paper and $\neg$ taken primitive instead of $\bot$.

cept of *saturated sequent* with respect to a certain sequent system is often introduced for completeness proofs; see e.g. [12].

Definition 3.8 ($\mathbf{TN}^3_4$[d][l]-saturatedness) An infinite tree-sequent $\mathcal{T}$ of $\mathbf{TN}^3_4$[l] is $\mathbf{TN}^3_4$[l]-*saturated* if it satisfies the following conditions:

1. If $(b : \Gamma_b \overset{\alpha_b}{\Rightarrow} \Delta_b)$ is a successor of $(a : \Gamma_a \overset{\alpha_a}{\Rightarrow} \Delta_a)$ in $\mathcal{T}$, then $\Gamma_a \subseteq \Gamma_b$;
2. For each node $(a : \Gamma_a \overset{\alpha_a}{\Rightarrow} \Delta_a)$ of $\mathcal{T}$,
   (a) [($\wedge$L)-saturated] If $A \wedge B \in \Gamma_a$, then $A \in \Gamma_a$ and $B \in \Gamma_a$;
   (b) [($\wedge$R)-saturated] If $A \wedge B \in \Delta_a$, then $A \in \Delta_a$ or $B \in \Delta_a$;
   (c) [($\rightarrow$L)-saturated] If $A \rightarrow B \in \Gamma_a$, then $A \in \Delta_a$ or $B \in \Gamma_a$;
   (d) [$(\rightarrow\mathrm{R})_{\mathrm{T}}$-saturated] If $A \rightarrow B \in \Delta_a$, then there exists a successor $(b : \Gamma_b \overset{\alpha_b}{\Rightarrow} \Delta_b)$ of a such that $A \in \Gamma_b$ and $B \in \Delta_b$;
   (e) [($\neg$L)-saturated] If $\neg A \in \Gamma_a$, then $A \in \Delta_a$;
   (f) [$(\neg\mathrm{R})_{\mathrm{T}}$-saturated] If $\neg A \in \Delta_a$, then there exists a successor $(b : \Gamma_b \overset{\alpha_b}{\Rightarrow} \Delta_b)$ of a such that $A \in \Gamma_b$;
   (g) [($\forall$L)-saturated] If $\forall x A \in \Gamma_a$, then $A[y/x] \in \Gamma_a$ for every y available at a;
   (h) [$(\forall\mathrm{R})_{\mathrm{T}}$-saturated] If $\forall x A \in \Delta_a$, then there exist a successor $(b : \Gamma_b \overset{\alpha_b}{\Rightarrow} \Delta_b)$ of a and a variable y such that $A[y/x] \in \Delta_b$;
   (i) [($\sim\wedge$L)-saturated] If $\sim(A \wedge B) \in \Gamma_a$, then $\sim A \in \Gamma_a$ or $\sim B \in \Gamma_a$;
   (j) [($\sim\wedge$R)-saturated] If $\sim(A \wedge B) \in \Delta_a$, then $\sim A \in \Delta_a$ and $\sim B \in \Delta_a$;
   (k) [($\sim\rightarrow$L)-saturated] If $\sim(A \rightarrow B) \in \Gamma_a$, then $A \in \Gamma_a$ and $\sim B \in \Gamma_a$;
   (l) [($\sim\rightarrow$R)-saturated] If $\sim(A \rightarrow B) \in \Delta_a$, then $A \in \Delta_a$ or $\sim B \in \Delta_a$;
   (m) [($\sim\neg$L)-saturated] If $\sim\neg A \in \Gamma_a$, then $A \in \Gamma_a$;
   (n) [($\sim\neg$R)-saturated] If $\sim\neg A \in \Delta_a$, then $A \in \Delta_a$;
   (o) [($\sim\sim$L)-saturated] If $\sim\sim A \in \Gamma_a$, then $A \in \Gamma_a$;
   (p) [($\sim\sim$R)-saturated] If $\sim\sim A \in \Delta_a$, then $A \in \Delta_a$;
   (q) [($\sim\forall$L)-saturated] If $\sim\forall x A \in \Gamma_a$, then $\sim A[y/x] \in \Gamma_a$ for some variable y;
   (r) [($\sim\forall$R)-saturated] If $\sim\forall x A \in \Delta_a$, then $\sim A[y/x] \in \Delta_a$ for every y available at a.

An infinite tree-sequent $\mathcal{T}$ of $\mathbf{TN}^3_4$d[l] is $\mathbf{TN}^3_4$d[l]-*saturated* if it satisfies the conditions above, replacing "for every y available at a" by "for every variable y".

Lemma 3.9 (Kripke completeness of $\mathbf{TN}^3_4$[d][l]) Let $\mathbf{TN}^3_4$[d][l] $\nvdash \mathcal{T}$ and at least one variable is available at the root of $\mathcal{T}$. Then $\mathcal{T}$ has a counter model.

Proof. First we extend $\mathcal{T}$ into a $\mathbf{TN}^3_4$[d][l]-saturated (possibly infinite) tree-sequent $\mathcal{T}_\omega$. We construct an infinite sequence of (finite) TS's $\mathcal{T}_0, \mathcal{T}_1, \ldots$ and the union of them (now it possibly becomes infinite) is what we would like to obtain.

Let $B_1, B_2, \ldots$ be an enumeration of all the formulas of N3. We rearrange the sequence and obtain another sequence

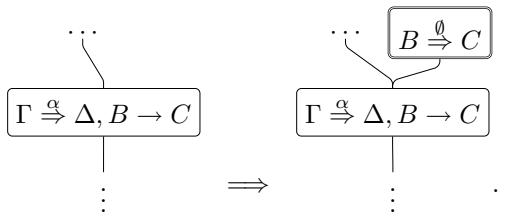
$$B_1 \mid B_1, B_2 \mid B_1, B_2, B_3 \mid \ldots,$$

which we denote by $A_1, A_2, \ldots$. In $A_1, A_2, \ldots$ every formula of N3 appears infinitely many times. Since every term of N3 is nothing but a variable we can enumerate them as $x_1, x_2, \ldots$.

First we consider the case of $\mathbf{TN3}$. The modifications of the proof for its variants we shall describe later.

Let $\mathcal{T}_0 :\equiv \mathcal{T}$. The i-th step, which is the step of extension from $\mathcal{T}_{i-1}$ to $\mathcal{T}_i$, consists of *inheritance* and *reduction* of the formula A_i, using a finite number of variables $x_1, x_2, \ldots, x_i$ (hence the step involves only a finite number of operations). Note that in each operation unprovability of the TS is preserved. The operations executed in the i-th step are as follows:

1. [inheritance] For each node $(a : \Gamma_a \overset{\alpha_a}{\Rightarrow} \Delta_a)$ such that $A_i \in \Gamma_a$, add A_i to the antecedents of all the successors of a. Unprovability is preserved because of the rule (D) of $\mathbf{TN3}$.
2. [reduction] According to the shape of A_i, one of the following operations is executed for each node $(a : \Gamma_a \overset{\alpha_a}{\Rightarrow} \Delta_a)$:
   (a) $[A_i \equiv B \wedge C]$ If $A_i \in \Gamma_a$, then add B and C to Γ_a. Unprovability is preserved because of the rule ($\wedge$L) of $\mathbf{TN3}$. If $A_i \in \Delta_a$, then add either B or C to Δ_a, so that unprovability is preserved. This is possible because of the rule ($\wedge$R); if neither choice preserves unprovability, then by the rule the original TS must be provable, which is a contradiction. Each operation below also preserves unprovability because of the corresponding rule of $\mathbf{TN3}$, although we do not put explicitly.
   (b) $[A_i \equiv B \to C]$ If $A_i \in \Gamma_a$, then add B to Δ_a or C to Γ_a, so that unprovability is preserved. If $A_i \in \Delta_a$, then make a new leaf $(b : B \overset{\emptyset}{\Rightarrow} C)$ as an immediate successor of a:



   (c) $[A_i \equiv \neg B]$ If $A_i \in \Gamma_a$, then add B to Δ_a. If $A_i \in \Delta_a$, then make a new leaf $(b : B \overset{\emptyset}{\Rightarrow} \ )$ as an immediate successor of a.
   (d) $[A_i \equiv \forall x B]$ If $A_i \in \Gamma_a$, then add $B[x_j/x]$ to Γ_a, for each $x_j \in \{x_1, \ldots, x_i\}$ which is available at a. If $A_i \in \Delta_a$, then take a fresh variable x_m and make a new leaf $(b : \overset{\{x_m\}}{\Rightarrow} B[x_m/x])$ as an immediate successor of a; we can take fresh x_m since the original TS is finite.
   (e) $[A_i \equiv \sim(B \wedge C)]$ If $A_i \in \Gamma_a$, add $\sim B$ or $\sim C$ to Γ_a so that unprovability is preserved. If $A_i \in \Delta_a$, add $\sim B$ and $\sim C$ to Δ_a.
   (f) $[A_i \equiv \sim(B \to C)]$ If $A_i \in \Gamma_a$, add B and $\sim C$ to Γ_a. If $A_i \in \Delta_a$, add B or $\sim C$ to Δ_a so that unprovability is preserved.
   (g) $[A_i \equiv \sim\neg B$ or $A_i \equiv \sim\sim B]$ If $A_i \in \Gamma_a$ (or Δ_a), add B to Γ_a (or Δ_a, respectively).
   (h) $[A_i \equiv \sim\forall x B]$ If $A_i \in \Gamma_a$, then take a fresh variable x_m, add $\sim B[x_m/x]$ to Γ_a, and also add x_m to α_a. If $A_i \in \Delta_a$, add $\sim B[x_j/x]$ to Δ_a, for each $x_j \in \{x_1, \ldots, x_i\}$ which is available at a.

For variants of $\mathsf{N3}$, the following modifications are to be made:

4 Nothing.

d Omit conditions which involves availability of variables, and the operation (2d) is replaced by

(d) $[A_i \equiv \forall x B]$ If $A_i \in \Gamma_a$, then add $B[x_j/x]$ to Γ_a, for each x_j in $\{x_1, \ldots, x_i\}$. If $A_i \in \Delta_a$, then take a fresh variable x_m and add $B[x_m/x]$ to Δ_a; unprovability is preserved because of the rule $(\forall\mathrm{R})_{\mathrm{VC}}$.

l The operations (2b), (2c) and (2d) are replaced: (2d) is replaced by

(d) $[A_i \equiv \forall x B]$ If $A_i \in \Gamma_a$, then add $B[x_j/x]$ to Γ_a, for each $x_j \in \{x_1, \ldots, x_i\}$ which is available at a. Assume $A_i \in \Delta_a$. Let the TS to which the operation is about to be applied denoted by

$$\cdots \mid \Gamma_1 \overset{\alpha_1}{\Rightarrow} \Delta_1 (\equiv \Gamma_a \overset{\alpha_a}{\Rightarrow} \Delta_a) \mid \Gamma_2 \overset{\alpha_2}{\Rightarrow} \Delta_2 \mid \cdots \mid \Gamma_k \overset{\alpha_k}{\Rightarrow} \Delta_k.$$

Now take a fresh variable x_m and let the TS's $\mathcal{S}_1, \ldots, \mathcal{S}_k$ be

$$\mathcal{S}_l :\equiv \cdots \mid \Gamma_1 \overset{\alpha_1}{\Rightarrow} \Delta_1 \mid \cdots \mid \Gamma_l \overset{\alpha_l}{\Rightarrow} \Delta_l \mid \overset{\{x_m\}}{\Rightarrow} B[x_m/x] \mid \cdots \mid \Gamma_k \overset{\alpha_k}{\Rightarrow} \Delta_k. \quad (l \in [1,k])$$

Take an unprovable $\mathcal{S}_l$ as a new TS; this is possible because of the rule $(\forall\mathrm{R})_{\mathrm{K}}$.

The modified (2b) and (2c) are just the same as above, inserting a new node in such a position that unprovability is preserved.

Let $\mathcal{T}_\omega$ be the union of $\mathcal{T}_0, \mathcal{T}_1, \ldots$, i.e. the tree-structure, associating sequents and associating sets of variables of $\mathcal{T}_\omega$ are the unions of those of $\mathcal{T}_0, \mathcal{T}_1, \ldots$. It is easily verified that $\mathcal{T}_\omega$ is $\mathbf{TN4^3_4}[\mathsf{d}][\mathsf{l}]$-saturated.

We construct an $\mathsf{N3}$-model $\mathcal{M} = (M, \leq, U, I^+, I^-)$ using syntactical objects, namely the $\mathbf{TN3}$-saturated $\mathcal{T}_\omega$ obtained above. Let $(M, \leq)$ be the tree-structure of $\mathcal{T}_\omega$, $U(a)$ be the set of available variables at a. For each node $(a : \Gamma_a \overset{\alpha_a}{\Rightarrow} \Delta_a)$ of $\mathcal{T}_\omega$ and each predicate symbol p (which is m-ary), the verum / falsum interpretation of p at a is defined by:

$$p^{I^+(a)} := \{(y_1, \ldots, y_m) \mid p(y_1, \ldots, y_m) \in \Gamma_a\},$$
$$p^{I^-(a)} := \{(y_1, \ldots, y_m) \mid {\sim}p(y_1, \ldots, y_m) \in \Gamma_a\}.$$

It is easily verified that $\mathcal{M}$ satisfies the conditions for $\mathsf{N3}$-model. Indeed, the condition on $\mathcal{T}$ that some variables are available at its root yields that $U(a) \neq \emptyset$ for every $a \in M$. If $a \leq b$ and $(y_1, \ldots, y_m) \in p^{I^+(a)}$, then $p(y_1, \ldots, y_m) \in \Gamma_a$ and by the operation [inheritance] $p(y_1, \ldots, y_m) \in \Gamma_b$, hence $(y_1, \ldots, y_m) \in p^{I^+(b)}$. And the fact that every finite sub-tree-sequent of $\mathcal{T}_\omega$ is unprovable in $\mathbf{TN3}$ and the initial TS (Fal) yield that $p^{I^+(a)} \cap p^{I^-(a)} = \emptyset$.

For $\mathbf{N4^3_4}[\mathsf{l}]$ $\mathcal{M}$ is constructed in the same way. For those with d (i.e. $\mathbf{N4^3_4}[\mathsf{d}][\mathsf{l}]$), $U(a)$ is defined to be just the set of all variables for every $a \in M$.

Since $\mathcal{T}$ is a sub-tree-sequent of $\mathcal{T}_\omega$, we can take the identity map as an embedding of the tree-structure of $\mathcal{T}$ in $(M, \leq)$, and also can take the set of all variables as W, where $U : M \to \mathcal{P}W$. Then it is again easily verified by induction on the construction of formulas that $\mathcal{M}$ is a counter model for $\mathcal{T}$. ∎

Definition 3.10 (Formulaic translation of TS) Let $\mathcal{T}$ be a pre-tree-sequent of $\mathsf{N}^3_4[\mathsf{I}]$. The *formulaic translation* of $\mathcal{T}$, denoted by $\mathcal{T}^f$, is defined inductively on the height of $\mathcal{T}$:

$$[\Gamma \overset{\alpha}{\Rightarrow} \Delta \mid \mathcal{T}_1 \dots \mathcal{T}_m]^f :\equiv \forall \overrightarrow{\alpha}\big((\bigwedge \Gamma) \to (\bigvee \Delta) \vee \mathcal{T}_1^f \vee \dots \vee \mathcal{T}_m^f \big).$$

Let $\mathcal{T}$ be a TS of $\mathsf{N}^3_4\mathsf{d}[\mathsf{I}]$. The *formulaic translation* of $\mathcal{T}$, again denoted by $\mathcal{T}^f$, is a universal closure of $\mathcal{T}^p$, which in turn is defined inductively on the height of $\mathcal{T}$:

$$[\Gamma \Rightarrow \Delta \mid \mathcal{T}_1 \dots \mathcal{T}_m]^p :\equiv (\bigwedge \Gamma) \to (\bigvee \Delta) \vee \mathcal{T}_1^p \vee \dots \vee \mathcal{T}_m^p$$

Lemma 3.11 If $\mathbf{TN}^3_4[\mathsf{d}][\mathsf{I}] \vdash \mathcal{T}$, then $\mathbf{GN}^3_4[\mathsf{d}][\mathsf{I}] \vdash \mathcal{T}^f$.

To prove this lemma we prove a couple of sublemmas, whose proofs are easy by induction on the height of a in $\mathcal{T}$.

Sublemma 3.12 Let $\mathcal{T}$ be a TS of $\mathbf{TN}^3_4[\mathsf{I}]$, a a node of $\mathcal{T}$, $\mathcal{T}'$ be a pTS which consists of a and all of its successors, $\mathcal{T}'_1, \dots, \mathcal{T}'_k$ pre-tree-sequents of $\mathbf{TN}^3_4[\mathsf{I}]$, and $\mathcal{T}_1, \dots, \mathcal{T}_k$ be a tree-sequent obtained by replacing $\mathcal{T}'$ by $\mathcal{T}'_1, \dots, \mathcal{T}'_k$, respectively (Fig. 3).

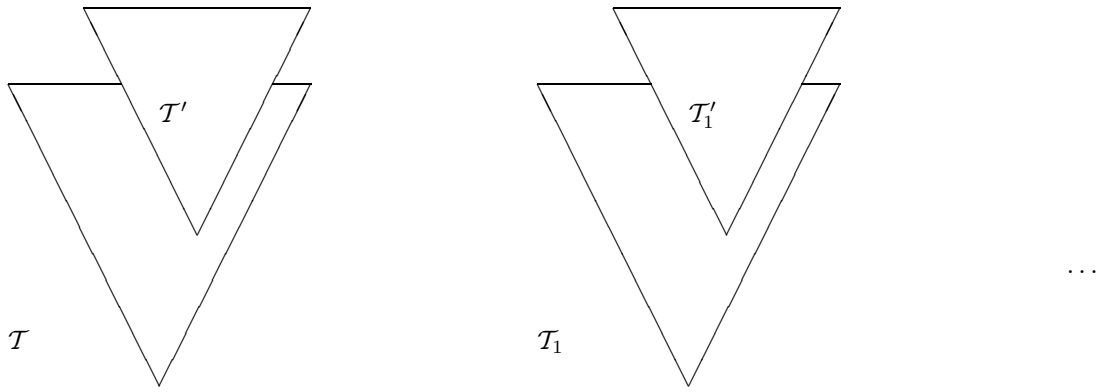


FIG. 3. $\mathcal{T}, \mathcal{T}_1, \dots, \mathcal{T}_k$

If $\mathbf{GN}^3_4[\mathsf{I}] \vdash \mathcal{T}'^f_1, \dots, \mathcal{T}'^f_k \Rightarrow \mathcal{T}'^f$, then $\mathbf{GN}^3_4[\mathsf{I}] \vdash \mathcal{T}^f_1, \dots, \mathcal{T}^f_k \Rightarrow \mathcal{T}^f$.

Let $\mathcal{T}$ be a TS of $\mathbf{TN}^3_4\mathsf{d}[\mathsf{I}]$, $\mathcal{T}', \mathcal{T}'_1, \dots, \mathcal{T}'_k, \mathcal{T}_1, \dots, \mathcal{T}_k$ be just as above. If $\mathbf{GN}^3_4\mathsf{d}[\mathsf{I}] \vdash \mathcal{T}'^p_1, \dots, \mathcal{T}'^p_k \Rightarrow \mathcal{T}'^p$, then $\mathbf{GN}^3_4\mathsf{d}[\mathsf{I}] \vdash \mathcal{T}^p_1, \dots, \mathcal{T}^p_k \Rightarrow \mathcal{T}^p$.

Sublemma 3.13 Let $\mathcal{T}$ and $\mathcal{T}'$ be just as in the sublemma above. If $\mathbf{GN}^3_4[\mathsf{d}][\mathsf{I}] \vdash \mathcal{T}'^f$, then $\mathbf{GN}^3_4[\mathsf{d}][\mathsf{I}] \vdash \mathcal{T}^f$.

PROOF. (of Lemma 3.11) The case for N^3_4 is easy. The proof is by induction on the derivation of $\mathcal{T}$ in $\mathbf{TN}^3_4$. When $\mathcal{T}$ is an initial TS, use the second sublemma. For the

step cases where $\mathcal{T}$ is derived by a derivation rule with a hypothesis (or hypotheses), use the first sublemma.

For those logics with I the proofs are just the same; however, the step case where $\mathcal{T}$ is derived by the rule $(\to\mathrm{R})_{\mathrm{K}}$, $(\neg\mathrm{R})_{\mathrm{K}}$ or $(\forall\mathrm{R})_{\mathrm{K}}$ is rather complicated. Here we present the case for $(\to\mathrm{R})_{\mathrm{K}}$. It suffices to prove that

$$\begin{aligned}&\mathbf{GN4I}\ (\text{hence } \mathbf{GN}^3_4[\mathsf{d}]\mathsf{I}) \vdash E \to F \vee \forall\vec{x}(C \to D \vee (A \to B)),\\ &\quad E \to F \vee (A \to B \vee \forall\vec{x}(C \to D)) \Rightarrow E \to F \vee (A \to B) \vee \forall\vec{x}(C \to D) \qquad (\clubsuit)\end{aligned}$$

where $\vec{x}$ have their free occurrences only in C or D. Using this fact repeatedly, we can finish the proof.

$$(A \to B) \to \forall\vec{x}(C \to D), \forall\vec{x}(C \to D \vee (A \to B)) \Rightarrow \forall\vec{x}(C \to D) \qquad (\diamondsuit)$$
$$\forall\vec{x}(C \to D) \to (A \to B), A \to B \vee \forall\vec{x}(C \to D) \Rightarrow A \to B \qquad (\heartsuit)$$

The above two sequents are derivable in $\mathbf{GN}^3_4[\mathsf{d}]\mathsf{I}$, as is easily verified.

$$\dfrac{\dfrac{\dfrac{}{((A \to B) \to \forall\vec{x}(C \to D)) \vee (\forall\vec{x}(C \to D) \to (A \to B))}\ (\mathrm{Li}) \quad \dfrac{(\diamondsuit)\quad(\heartsuit)}{\cdots}\ (\vee\mathrm{L})}{\forall\vec{x}(C \to D \vee (A \to B)), A \to B \vee \forall\vec{x}(C \to D) \Rightarrow (A \to B) \vee \forall\vec{x}(C \to D)}\ (\mathrm{C})}{(\clubsuit)}$$

For those logics with d, the proof is again almost the same; however, we cannot apply the first sublemma when $\mathcal{T}$ is derived by the rule $(\forall\mathrm{R})_{\mathrm{VC}}$. Instead we use the character of logics with D, that is:

$$\forall x(B \to C(x)) \cong_{\mathbf{N}^3_4\mathsf{d}[\mathsf{I}]} B \to \forall x C(x), \quad \forall x(B \vee C(x)) \cong_{\mathbf{N}^3_4\mathsf{d}[\mathsf{I}]} B \vee \forall x C(x),$$

where x has no free occurrences in B and $\cong$ denotes logical equivalence. Suppose $\mathbf{GN}^3_4\mathsf{d}[\mathsf{I}] \vdash (\cdots[\Gamma \Rightarrow \Delta, A[z/x] \mid \cdots)^f$ as an induction hypothesis. Then $(\cdots[\Gamma \Rightarrow \Delta, A[z/x] \mid \cdots)^f$ is in the form

$$\forall\vec{x}\forall z(B_1 \to C_1 \vee (B_2 \to C_2 \vee (\cdots \vee (B_n \to C_n \vee A[z/x])\cdots))),$$

where z has no free occurrence in any B_i or C_i. We repeatedly apply the logical equivalences above and obtain

$$\begin{aligned}(\cdots[\Gamma \Rightarrow \Delta, A[z/x] \mid \cdots)^f &\equiv \forall\vec{x}\forall z(B_1 \to C_1 \vee (\cdots \vee (B_n \to C_n \vee A[z/x])\cdots))\\ &\cong_{\mathbf{N}^3_4\mathsf{d}[\mathsf{I}]} \forall\vec{x}(B_1 \to C_1 \vee (\cdots \vee (B_n \to C_n \vee \forall x A)\cdots))\\ &\equiv (\cdots[\Gamma \Rightarrow \Delta, \forall x A \mid \cdots)^f.\end{aligned}$$

Then the induction hypothesis yields $\mathbf{GN}^3_4\mathsf{d}[\mathsf{I}] \vdash (\cdots[\Gamma \Rightarrow \Delta, \forall x A \mid \cdots)^f$. ∎

Using Kripke completeness of $\mathbf{TN}^3_4[\mathsf{d}][\mathsf{I}]$ and the results on translations obtained above, we can conclude Kripke completeness of $\mathbf{GN}^3_4[\mathsf{d}][\mathsf{I}]$.

Theorem 3.14 (Kripke completeness of $\mathbf{GN}^3_4[\mathsf{d}][\mathsf{I}]$) If $\mathsf{N}^3_4[\mathsf{d}][\mathsf{I}] \models A$, then $\mathbf{GN}^3_4[\mathsf{d}][\mathsf{I}] \vdash A$.

Proof. Let $\mathbf{GN}^3_4[\mathsf{d}][\mathsf{I}] \nvdash A$, and $\mathcal{T}$ be a TS which consists of only its root; for $\mathsf{N}^3_4[\mathsf{I}]$ $\mathcal{T} :\equiv [\ \overset{\alpha}{\Rightarrow} A]$ where α is a nonempty finite set of variables which contains every free variable in A, and for $\mathsf{N}^3_4\mathsf{d}[\mathsf{I}]$ $\mathcal{T} :\equiv [\ \Rightarrow A]$. Then $\mathbf{TN}^3_4[\mathsf{d}][\mathsf{I}] \nvdash \mathcal{T}$ by Lemma 3.11, and by Lemma 3.9 there exists a counter model $\mathcal{M}$ for $\mathcal{T}$, which is also a counter model for A, i.e. $\mathcal{M} \not\models A$. ∎

4 Kripke completeness of GN3[d][l]o

4.1 Proof by an embedding of Cl *– for* GN3[l]o

As stated in the introduction, Kripke completeness of **GN3[d][l]o** can hardly be shown by a simple application of the tree-sequent method. For **GN3[d][l]o** we present two different proofs in this section; given an unprovable formula A, we construct a counter **N3[d][l]o**-model for it. Again as stated above, a counter model $\mathcal{M} = (M, \leq, U, I^+, I^-)$ we shall construct is such that: each $a \in M$ which is not omniscient has an immediate successor a_g which is maximal in $(M, \leq)$ and is omniscient. Since **GN3[d][l]o** is stronger than **GN3[d][l]**, A is also unprovable in **GN3[d][l]**; hence by completeness in the previous section there exists a counter **GN3[d][l]**-model $\mathcal{M}'$. The problem is how to obtain an omniscient possible world a_g which will be associated to each node a of $\mathcal{M}'$. The first proof, presented in this subsection, utilizes an embedding of Cl in **GN3[l]o** for this purpose. As we point out later, this method is not applicable to **GN3d[l]o**.

First we present some derivations possible in $\mathbf{GN}^3_4$**[d][l]o**:

Lemma 4.1 In **GN4o** (hence also in $\mathbf{GN}^3_4$**[d][l]o**), we can make the following derivations:

$$\frac{}{\neg\sim A, \neg A \Rightarrow}\ (\mathrm{Om1}) \qquad \frac{\sim A, \Gamma \Rightarrow}{\neg A, \Gamma \Rightarrow}\ (\mathrm{Om2}) \qquad \frac{}{A \to B \Rightarrow \neg\neg(\sim A \vee B)}\ (\mathrm{Om3})$$

Proof. (Om1)

$$\dfrac{\dfrac{}{\Rightarrow \neg\neg(A \vee \sim A)}\ (\mathrm{Om}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{A \Rightarrow A}\ (\mathrm{Id})}{A, \neg\sim A, \neg A \Rightarrow}\ (\mathrm{W})(\neg\mathrm{L}) \quad \dfrac{\dfrac{}{\sim A \Rightarrow \sim A}\ (\mathrm{Id})}{\sim A, \neg\sim A, \neg A \Rightarrow}\ (\mathrm{W})(\neg\mathrm{L})}{A \vee \sim A, \neg\sim A, \neg A \Rightarrow}\ (\vee\mathrm{L})}{\neg\neg(A \vee \sim A), \neg\sim A, \neg A \Rightarrow}\ (\neg\mathrm{R})(\neg\mathrm{L})}{\neg\sim A, \neg A \Rightarrow}\ (\mathrm{C})$$

(Om2)

$$\dfrac{\dfrac{\dfrac{}{\neg\sim A, \neg A \Rightarrow}\ (\mathrm{Om1})}{\neg A \Rightarrow \neg\neg\sim A}\ (\neg\mathrm{R}) \qquad \dfrac{\dfrac{}{\sim A, \Gamma \Rightarrow}\ \text{hyp.}}{\neg\neg\sim A, \Gamma \Rightarrow}\ (\neg\mathrm{R},\ \mathrm{L})}{\neg A, \Gamma \Rightarrow}\ (\mathrm{C})$$

(Om3)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \Rightarrow A}\ (\mathrm{Id}) \quad \dfrac{}{B \Rightarrow B}\ (\mathrm{Id})}{A \to B, A \Rightarrow \sim A \vee B}\ (\to\mathrm{L})}{A \to B, A, \neg(\sim A \vee B) \Rightarrow}\ (\neg\mathrm{L}) \qquad \dfrac{\dfrac{\dfrac{}{\sim A \Rightarrow \sim A}\ (\mathrm{Id})}{A \to B, \sim A \Rightarrow \sim A \vee B}\ (\neg\mathrm{L})}{A \to B, \sim A, \neg(\sim A \vee B) \Rightarrow}\ (\neg\mathrm{L})}{A \to B, A \vee \sim A, \neg(\sim A \vee B) \Rightarrow}\ (\vee\mathrm{L})}{A \to B, \neg\neg(A \vee \sim A), \neg(\sim A \vee B) \Rightarrow}\ (\neg\mathrm{R})(\neg\mathrm{L})}{A \to B, \neg\neg(A \vee \sim A) \Rightarrow \neg\neg(\sim A \vee B)}\ (\neg\mathrm{R})}{A \to B \Rightarrow \neg\neg(\sim A \vee B)}\ (\mathrm{C})\ (\mathrm{Om})$$

∎

In fact (Om) is equivalent to (Om3): take $B \equiv A$ in (Om3), and use (C) with $\Rightarrow A \to A$. Moreover, where (Fal) is available (i.e. in $\mathsf{N3[d][l]o}$), (Om1), (Om2) and (Om) are all equivalent. Since the proof above shows that (Om)⇒(Om1)⇒(Om2), it suffices to show (Om2)⇒(Om):

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\sim A, \sim\sim A \Rightarrow}\,(\mathrm{Fal})}{\sim(A \vee \sim A) \Rightarrow}\,(\sim\vee\mathrm{L})}{\neg(A \vee \sim A) \Rightarrow}\,(\mathrm{Om2})}{\Rightarrow \neg\neg(A \vee \sim A)}\,(\neg\mathrm{R})$$

Now we make some remarks. All of $\vee$, $\to$ and $\exists$ can be introduced as defined symbols in Cl in terms of $\wedge$, $\neg$ and $\forall$; however in this paper we assume the logical connectives of Cl include $\to$, to make correspondence with $\mathsf{N3[d][l]o}$. In what follows a *structure* in Tarski-type semantics for Cl is often called a Cl-*model*. Cl-model can be regarded as an Int-model consisting of only one possible world. We denote the domain of a Cl-model $\mathcal{A}$ by $|\mathcal{A}|$, and the interpretation of a predicate symbol p by $p_{\mathcal{A}}$. We will often denote a formula of Cl by $A_\neg$ in order to make clear that A contains no $\sim$'s. $A_{\sim\neg}$ is a formula of $\mathsf{N3}$ which is obtained by replacing some (possibly all or no) $\neg$'s in A by $\sim$. The subscript $_{\sim\neg}$ is also often used to make clear that it is a formula of $\mathsf{N3}$. Then $A_\neg$ is a formula obtained from $A_{\sim\neg}$ by replacing every $\sim$ therein by $\neg$.

Lemma 4.2 Let $\mathcal{A}$ be a Cl-model, and $\mathcal{M}_{\mathcal{A}} = (M, \leq, U, I^+, I^-)$ an $\mathsf{N3[d][l]o}$-model defined by $M := \{0\}$, $U(0) := |\mathcal{A}|$, $p^{I^+(0)} := p_{\mathcal{A}}$ and $p^{I^-(0)} := U(0)^m \setminus p_{\mathcal{A}}$. Then for an arbitrary closed formula $A_\neg$ of Cl, $\mathcal{A} \models A_\neg$ iff $0 \models^+ A_{\sim\neg}$ (or equivalently $\mathcal{M} \models A_{\sim\neg}$). Moreover, $\mathcal{A} \not\models A_\neg$ iff $0 \models^- A_{\sim\neg}$.

Proof. It is easily verified by induction that, for an omniscient possible world a (i.e. $p^{I^+(a)} \cup p^{I^-(a)} = U(a)^m$) which is maximal in $(M, \leq)$ and a closed formula A of $\mathsf{N3}$, exactly one of $a \models^+ A$ or $a \models^- A$ holds; hence the first *iff* instantly yields the second one.

The first *iff* is shown by induction on the construction of A; we present just one step case. Assume $A_\neg \equiv \neg B_\neg$. Then $A_{\sim\neg} \equiv \neg B_{\sim\neg}$ or $\sim B_{\sim\neg}$. $\mathcal{A} \models A_\neg$ iff $\mathcal{A} \not\models B_\neg$, which is equivalent to $0 \not\models^+ B_{\sim\neg}$ by the induction hypothesis. This is equivalent to $0 \models^+ \neg B_{\sim\neg}$ since 0 has no successors, and is also equivalent to $0 \models^+ \sim B_{\sim\neg}$ by the above argument. ∎

Using the lemmas above, we can introduce the main lemma in this subsection:

Lemma 4.3 (Embedding of LK in G$\mathsf{N3[d][l]o}$) The following are all equivalent:

1. $\mathbf{LK} \vdash \Gamma_\neg \Rightarrow \Delta_\neg$;
2. $\mathbf{G}\mathsf{N3[d][l]o} \vdash \Gamma_{\sim\neg}, \sim\Delta_{\sim\neg} \Rightarrow$;
3. $\mathbf{G}\mathsf{N3[d][l]o} \vdash \Gamma_{\sim\neg}, \neg\Delta_{\sim\neg} \Rightarrow$.

Proof. [2 ⇔ 3] is obvious by (Om2) of Lemma 4.1 and $\mathbf{G}\mathsf{N3[d][l]o} \vdash \sim A \Rightarrow \neg A$.

[2 ⇒ 1] is shown semantically. Let $\mathcal{A}$ an arbitrary Cl-model. Kripke soundness of $\mathbf{G}\mathsf{N3[d][l]o}$ yields $\mathcal{M}_{\mathcal{A}} \models \Gamma_{\sim\neg}, \sim\Delta_{\sim\neg} \Rightarrow$, and by Lemma 4.2 we have $\mathcal{A} \models \Gamma_\neg, \neg\Delta_\neg \Rightarrow$, hence $\mathcal{A} \models \Gamma_\neg \Rightarrow \Delta_\neg$. Then by completeness of $\mathbf{LK}$, $\mathbf{LK} \vdash \Gamma_\neg \Rightarrow \Delta_\neg$.

$[1 \Rightarrow 2]$ is by the induction on derivation in **LK**. Note that we can restrict initial sequents (Id) of **LK** to the form $p(\vec{x}) \Rightarrow p(\vec{x})$, an atomic formula in each side. In the following $A_{\sim\neg}$ is denoted by A' to avoid proofs being too wide. The derivation rule of **LK** applied at last is presented in the left, and the corresponding proof figure in **GN3[d][l]o** is in the right:

$$\frac{}{p(\vec{x}) \Rightarrow p(\vec{x})}\ (\mathrm{Id}) \qquad \frac{}{p(\vec{x}), \sim p(\vec{x}) \Rightarrow}\ (\mathrm{Fal})$$

The case of (W) or a rule introducing $\wedge$ or $\forall$ is easy.

$$\frac{\Gamma \Rightarrow \Delta, A}{\neg A, \Gamma \Rightarrow \Delta}\ (\neg\mathrm{L}) \qquad \frac{\dfrac{\text{ind. hyp.}}{\Gamma', \sim\Delta', \sim A' \Rightarrow}}{\neg A', \Gamma', \sim\Delta' \Rightarrow}\ (\mathrm{Om2}) \quad \text{and} \quad \frac{\text{ind. hyp.}}{\sim A', \Gamma', \sim\Delta' \Rightarrow}$$

$$\frac{A, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \neg A}\ (\neg\mathrm{R}) \qquad \frac{\dfrac{\text{ind. hyp.}}{\Gamma', \sim\Delta', A' \Rightarrow}}{\Gamma', \sim\Delta', \sim\neg A' \ (\text{or } \sim\sim A') \Rightarrow}\ (\sim\neg\mathrm{L}) \text{ or } (\sim\sim\mathrm{L})$$

$$\frac{\Gamma \Rightarrow \Delta, A \quad B, \Gamma \Rightarrow \Delta}{A \to B, \Gamma \Rightarrow \Delta}\ (\to\mathrm{L}) \qquad \frac{\dfrac{\dfrac{\dfrac{\text{ind. hyp.}}{\sim A', \Gamma', \sim\Delta' \Rightarrow} \quad \dfrac{\text{ind. hyp.}}{B', \Gamma', \sim\Delta' \Rightarrow}}{\sim A' \vee B', \Gamma', \sim\Delta' \Rightarrow}\ (\vee\mathrm{L})}{\neg\neg(\sim A' \vee B'), \Gamma', \sim\Delta' \Rightarrow}\ (\neg\mathrm{R})(\neg\mathrm{L})}{A' \to B', \Gamma', \sim\Delta' \Rightarrow}\ (\mathrm{Om3})\ (\mathrm{C})$$

$$\frac{A, \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \to B}\ (\to\mathrm{R}) \qquad \frac{\dfrac{\text{ind. hyp.}}{\Gamma', \sim\Delta', A', \sim B' \Rightarrow}}{\Gamma', \sim\Delta', \sim(A' \to B') \Rightarrow}\ (\sim\to\mathrm{L})$$

∎

Corollary 4.4 If $\mathbf{GN3[d][l]o} \nvdash \Gamma_{\sim\neg} \Rightarrow \Delta_{\sim\neg}$, then $\mathbf{LK} \nvdash \Gamma_{\neg} \Rightarrow$.

We prove another several lemmas needed later.

Lemma 4.5 Let $\Gamma \Rightarrow \Delta$ be an infinite sequent of **LK** which is *consistent*, that is, for every finite subsets $\Gamma' \subseteq \Gamma$ and $\Delta' \subseteq \Delta$ we have $\mathbf{LK} \nvdash \Gamma' \Rightarrow \Delta'$. Then there exists a Cl-model $\mathcal{A}$ such that:

1. the set of all variables can be embedded in $|\mathcal{A}|$;
2. $\mathcal{A} \models A[\underline{\vec{x}}/\vec{x}]$ for every $A \in \Gamma$;
3. $\mathcal{A} \not\models B[\underline{\vec{x}}/\vec{x}]$ for every $B \in \Delta$.

PROOF. First increase variables twofold, adding x'_i for each original variable x_i. Then an infinite number of variables $x'_1, x'_2, \ldots$ have no occurrence in $\Gamma \Rightarrow \Delta$, which is an (infinite) sequent of the original formal language. Now with the condition that $\Gamma \Rightarrow \Delta$ is consistent we can extend it to a **LK**-saturated infinite sequent $\widetilde{\Gamma} \Rightarrow \widetilde{\Delta}$, which induces a Cl-model $\mathcal{A}$ where $|\mathcal{A}| = \{x_1, x'_1, x_2, x'_2, \ldots\}$ and the conditions 2. and 3. are satisfied. ∎

Definition 4.6 (Tree-sequent system TN3[l]o) A *tree-sequent* of **TN3**[l]o is the same as that of **TN3**[l], respectively, and initial sequents and derivation rules of **TN3**[l]o are those of **TN3**[l] plus

$$\frac{}{\cdots[\overset{\alpha}{\Rightarrow}\neg\neg(A\vee{\sim}A)\mid\cdots}\ (\mathrm{Om}),\qquad \frac{\cdots[\Gamma\overset{\alpha}{\Rightarrow}\Delta,A\mid\cdots\quad\cdots[A,\Sigma\overset{\alpha}{\Rightarrow}\Pi\mid\cdots}{\cdots[\Gamma,\Sigma\overset{\alpha}{\Rightarrow}\Delta,\Pi\mid\cdots}\ (\mathrm{C}).$$

The cut rule (C) is necessary in the proof of the following lemma. If it is the case (we do not know whether or not) that (C) can be eliminated in **GN3**[l]o, then so is (C) of **TN3**[l]o.

Lemma 4.7 Let $\mathcal{T}$ be a tree-sequent of **TN3**[l]o. If $\mathcal{T}$ has a node $(a:\Gamma_a\overset{\alpha}{\Rightarrow}\Delta_a)$ such that **GN3**[l]o $\vdash\Gamma_a\Rightarrow\Delta_a$, then **TN3**[l]o $\vdash\mathcal{T}$.

Proof. By induction on derivation in **GN3**[l]o. For initial sequents (Li) of **GN3**lo,

$$\frac{\mathcal{T}_1\quad\cdots\quad\dfrac{\mathcal{T}_{i,1}\quad\cdots\quad\mathcal{T}_{i,k+1}}{\mathcal{T}_i}\ (\to\mathrm{R})_{\mathrm{K}}\quad\cdots\quad\mathcal{T}_k}{\cdots\mid\overset{\alpha_1}{\Rightarrow}(A\to B)\vee(B\to A)\mid\Gamma_2\overset{\alpha_2}{\Rightarrow}\Delta_2\mid\cdots\mid\Gamma_k\overset{\alpha_k}{\Rightarrow}\Delta_k}\ (\to\mathrm{R})_{\mathrm{K}},(\vee\mathrm{R})$$

where

$$\mathcal{T}_i\equiv\cdots\mid\overset{\alpha_1}{\Rightarrow}B\to A\mid\cdots\mid\Gamma_i\overset{\alpha_i}{\Rightarrow}\Delta_i\mid A\overset{\emptyset}{\Rightarrow}B\mid\cdots\mid\Gamma_k\overset{\alpha_k}{\Rightarrow}\Delta_k,\qquad i\in[1,k]$$

$$\mathcal{T}_{i,j}\equiv\cdots\mid\overset{\alpha_1}{\Rightarrow}\ \mid\cdots\mid A\overset{\emptyset}{\Rightarrow}B\mid\cdots\mid B\overset{\emptyset}{\Rightarrow}A\mid\cdots\mid\Gamma_k\overset{\alpha_k}{\Rightarrow}\Delta_k.\qquad j\in[1,k+1]$$

Every $\mathcal{T}_{i,j}$ is provable in **TN3**lo by (Id), (W) and (D).

It is easy to check the cases for the other initial sequents or the rules which admit side formulas in its conclusion, i.e. other than $(\ \ )_{\mathrm{S}}$. For the rule (C) of **GN3**[l]o we need (C) of **TN3**[l]o.

We present only the case for $(\to\mathrm{R})_{\mathrm{S}}$; for $(\neg\mathrm{R})_{\mathrm{S}}$ and $(\forall\mathrm{R})_{\mathrm{S,\ VC}}$ proofs are just the same.

For $\dfrac{A,\Gamma\Rightarrow B}{\Gamma\Rightarrow A\to B}\ (\to\mathrm{R})_{\mathrm{S}}$, we have $\dfrac{\dfrac{\dfrac{\text{ind. hyp.}}{\cdots[\overset{\alpha}{\Rightarrow}\mid\cdots[A,\Gamma\overset{\emptyset}{\Rightarrow}B]\cdots]\cdots}}{\cdots[\Gamma\overset{\alpha}{\Rightarrow}\mid\cdots[A\overset{\emptyset}{\Rightarrow}B]\cdots]\cdots}\ (\mathrm{D})}{\cdots[\Gamma\overset{\alpha}{\Rightarrow}A\to B\mid\cdots}\ (\to\mathrm{R})_{\mathrm{T}}$ in **GN3**o.

In **GN3**lo, the proof is similar; first drop Γ, then apply $(\to\mathrm{R})_{\mathrm{K}}$. ∎

We define the *formulaic translation* of a TS $\mathcal{T}$ of **TN3**[l]o, denoted by $\mathcal{T}^f$, as that of **TN3**[l]: $[\Gamma\overset{\alpha}{\Rightarrow}\Delta\mid\mathcal{T}_1\ldots\mathcal{T}_m]^f:\equiv\forall\overrightarrow{\alpha}\big(\ (\bigwedge\Gamma)\to(\bigvee\Delta)\vee\mathcal{T}_1^f\vee\cdots\vee\mathcal{T}_m^f\ \big)$.

Lemma 4.8 If **TN3**[l]o $\vdash\mathcal{T}$, then **GN3**[l]o $\vdash\mathcal{T}^f$.

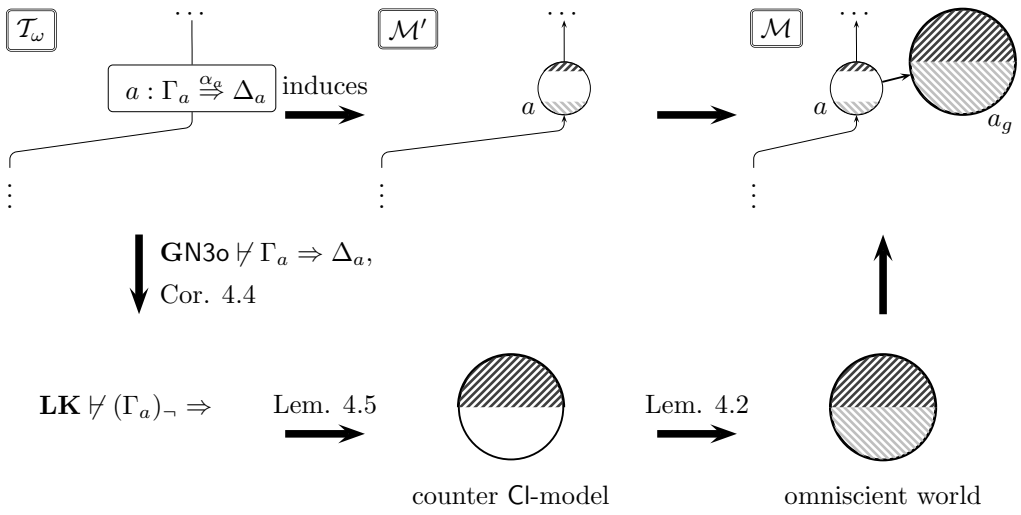
Proof. Similar to that of Lemma 3.11. ∎

Theorem 4.9 (Kripke completeness of GN3[l]o) If N3[l]o $\models A$, then **GN3**[l]o $\vdash A$.

PROOF. Let **G**N3[l]o $\nvdash A$. We shall construct a counter N3[l]o-model for A in the following way. The case of **G**N3o is presented ahead of **G**N3lo.

For **G**N3o, first we construct a counter N3-model $\mathcal{M}'$ by extending the TS $\mathcal{T} :\equiv [\overset{\alpha}{\Rightarrow} A]$ into a **T**N3-saturated infinite TS $\mathcal{T}_\omega$, as in the proof of Kripke completeness of **T**N3. By Lemma 4.8 $\mathcal{T}$ is unprovable in **T**N3o, and obviously the procedures of extension preserve unprovability of the TS; hence every finite sub-tree-sequent of $\mathcal{T}_\omega$ is unprovable in **T**N3o. Now Lemma 4.7 yields that for each node $(a : \Gamma_a \overset{\alpha_a}{\Rightarrow} \Delta_a)$ of $\mathcal{T}_\omega$, an infinite sequent $\Gamma_a \Rightarrow \Delta_a$ is consistent in **G**N3o, i.e. every finite subsequent of $\Gamma_a \Rightarrow \Delta_a$ is unprovable in **G**N3o.

Next we associate an omniscient possible world a_g to each node a of N3-model $\mathcal{M}'$ and obtain a counter N3o-model $\mathcal{M}$. Such omniscient worlds are obtained as follows: the fact stated above that every $\Gamma_a \Rightarrow \Delta_a$ (which is a node of $\mathcal{T}_\omega$) is consistent in **G**N3o and Corollary 4.4 yield that an infinite sequent $(\Gamma_a)_\neg \Rightarrow$ is consistent in **LK**. Now by Lemma 4.5 we obtain a counter Cl-model for $(\Gamma_a)_\neg \Rightarrow$ with a sufficiently large domain, which in turn induces an omniscient possible world a_g of N3o-model by Lemma 4.2. For each node a of $\mathcal{M}'$, we refer to the node $(a : \Gamma_a \overset{\alpha_a}{\Rightarrow} \Delta_a)$ of $\mathcal{T}_\omega$ which induces it, then add an omniscient world a_g obtained as above (i.e. by the fact that $(\Gamma_a)_\neg \Rightarrow$ is consistent in **LK**) as an immediate successor of a (see Fig. 4.1).



For **G**N3lo, the proof is just the same. First extend the TS $\mathcal{T} :\equiv [\overset{\alpha}{\Rightarrow} A]$ into a **T**N3l-saturated infinite TS $\mathcal{T}_\omega$, just as in the proof of completeness of **T**N3l. Let $\mathcal{M}'$ be a counter N3l-model for A induced by $\mathcal{T}_\omega$. Let us denote $\mathcal{T}_\omega$ by $\Gamma_1 \overset{\alpha_1}{\Rightarrow} \Delta_1 \mid \Gamma_2 \overset{\alpha_2}{\Rightarrow} \Delta_2 \mid \cdots$ and $\Gamma_\omega := \bigcup_i \Gamma_i$, $\Delta_\omega := \bigcup_i \Delta_i$. Then by Lemma 4.7 each $\Gamma_i \Rightarrow \Delta_i$ is consistent in **G**N3lo, hence $\Gamma_\omega \Rightarrow \Delta_\omega$ is also. Corollary 4.4 yields that $(\Gamma_\omega)_\neg \Rightarrow$ is consistent in **LK**, and induces an omniscient world g in the same way as above. By adding g to $\mathcal{M}'$ as the maximum, we obtain a counter N3lo-model $\mathcal{M}$ for A.

It remains to be proved that $\mathcal{M}$ is certainly an N3[l]o-model, and $\mathcal{M} \not\models A$. In the following only the case for **G**N3o is in consideration; for **G**N3lo the proof is just the same.

Since $U(a)$ is the set of the available variables and $U(a_g)$ is $\{x_1, x'_1, x_2, x'_2, \dots\}$ by Lemma 4.5, $U(a) \subseteq U(a_g)$. If $\vec{x} \in p^{I^+(a)}$ (or $\vec{x} \in p^{I^-(a)}$), then by the definition of $\mathcal{M}'$, $p(\vec{x}) \in \Gamma_a$ (or $\sim p(\vec{x}) \in \Gamma_a$) where $(a : \Gamma_a \overset{\alpha_g}{\Rightarrow} \Delta_a)$ is the corresponding node of $\mathcal{T}_\omega$; hence $p(\vec{x})$ is in $(\Gamma_a)_\neg$ (or not in $(\Gamma_a)_\neg$, respectively). By the construction of a_g, i.e. by Lemma 4.2, $\vec{x} \in p^{I^+(a_g)}$ (or $\vec{x} \in p^{I^-(a_g)}$). It is easy to verify that $\mathcal{M}$ satisfies the other conditions of N3o-models.

In order to show $\mathcal{M} \not\models A$, it suffices to show that: for each node $(a : \Gamma_a \overset{\alpha_g}{\Rightarrow} \Delta_a)$ of $\mathcal{T}_\omega$, $B \in \Gamma_a$ (or $B \in \Delta_a$) implies that $a \models^+ B[\underline{\vec{x}}/\vec{x}]$ (or $a \not\models^+ B[\underline{\vec{x}}/\vec{x}]$, respectively) holds also in $\mathcal{M}$ (it holds in $\mathcal{M}'$ by definition). The proof is easy by induction; the point is that the addition of a_g does not affect such conditions as that for $a \not\models^+ \neg B$, that is, "there exists *at least one* world $b \geq a$ such that $b \models^+ B$". ∎

Remark 4.10 The method above is not applicable to **G**N3d[l]o. The problem lies in the proof of Lemma 4.5; we cannot construct a counter Cl-model without infinitely many additional variables.

4.2 Proof by TSg method – for **G**N3[d][l]o

The method in the previous subsection is not applicable to **G**N3d[l]o; that is because in the method we must construct omniscient possible worlds *after we have finished* extending an unprovable formula into an *infinite* TS. To avoid this problem, it seems reasonable to consider a method in which we extend both the TS and the seeds of omniscient possible worlds *simultaneously*; this is the idea of the proofs in this subsection.

In the proofs here we make usage of a *tree-sequent with guardians*, TSg in short. Roughly speaking, a TSg is a TS each node of which has an extra sequent and a finite set of variables; we denote a node a which is associated with $\Gamma_a \overset{\alpha_g}{\Rightarrow} \Delta_a$ and an extra $\Sigma_a \overset{\beta_g}{\Rightarrow} \Pi_a$ by $(a : \Gamma_a \overset{\alpha_g}{\Rightarrow} \Delta_a \uparrow \Sigma_a \overset{\beta_g}{\Rightarrow} \Pi_a)$. An extra sequent $\Sigma_a \overset{\beta_g}{\Rightarrow} \Pi_a$ (presented in the right) is said to be the *guardian* of a, and is the seed of an omniscient world a_g which will be assigned to a. The TS-translation of a TSg, which we shall define later, will make clear the idea of TSg.

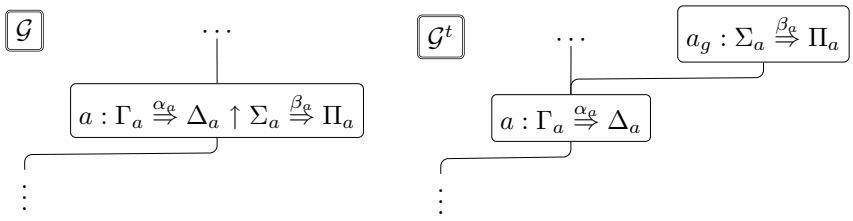
Using TSg's, we can prove Kripke completeness in the same way as that of $\mathbf{G}\mathsf{N}^3_4$[d][l]. Completeness of the corresponding TSg system is shown by extending an unprovable TSg into a saturated one, which induces a counter model. Then using the formulaic translation of a TSg, we can conclude completeness of the Gentzen-style sequent system.

Now we proceed to precise arguments, defining TSg systems **Tg**N3[d][l]o.

Definition 4.11 (TSg of TgN3[d][l]o**)** A *tree-sequent with guardians* $\mathcal{G}$ of **Tg**N3o is a finite tree, each node a of which is associated with two sequents of **G**N3o and finite sets of variables, denoted by $(a : \Gamma_a \overset{\alpha_g}{\Rightarrow} \Delta_a \uparrow \Sigma_a \overset{\beta_g}{\Rightarrow} \Pi_a)$, and satisfies the following condition:

Let $\mathcal{G}^t$ be a labelled tree obtained by, for each node a, omitting the guardian

$\Sigma_a \overset{\beta_a}{\Rightarrow} \Pi_a$ and adding a new immediate successor $(a_g : \Sigma_a \overset{\beta_a}{\Rightarrow} \Pi_a)$ to a:



Then $\mathcal{G}^t$ is a tree-sequent of **T**N3; that is, $\mathcal{G}^t$ satisfies the conditions as to availability of variables.

We shall call $\Gamma_a \overset{\alpha_a}{\Rightarrow} \Delta_a$ the *sequent* (or *left sequent*) of a, and $\Sigma_a \overset{\beta_a}{\Rightarrow} \Pi_a$ the *guardian* (or *guardian sequent*). The TS $\mathcal{G}^t$ is called the *TS-translation* of a TSg $\mathcal{G}$. A variable x is said to be *l-available at* a node a of $\mathcal{G}$ if it is available at a in $\mathcal{G}^t$. x is *g-available at* a if it is available at a_g in $\mathcal{G}^t$.

For variants, the definition is modified as follows:

d A TSg is just a labelled tree each node of which is associated with two sequents.

l A TSg $\mathcal{G}$ and its TS-translation $\mathcal{G}^t$ are simpler:

$$\mathcal{G} \equiv \Gamma_1 \overset{\alpha_1}{\Rightarrow} \Delta_1 \mid \Gamma_2 \overset{\alpha_2}{\Rightarrow} \Delta_2 \mid \cdots \mid \Gamma_k \overset{\alpha_k}{\Rightarrow} \Delta_k \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi,$$
$$\mathcal{G}^t \equiv \Gamma_1 \overset{\alpha_1}{\Rightarrow} \Delta_1 \mid \Gamma_2 \overset{\alpha_2}{\Rightarrow} \Delta_2 \mid \cdots \mid \Gamma_k \overset{\alpha_k}{\Rightarrow} \Delta_k \mid \Sigma \overset{\beta}{\Rightarrow} \Pi,$$

for **Tg**N3lo. For **Tg**N3dlo, omit the sets of variables. Here $\Sigma \overset{\beta}{\Rightarrow} \Pi$ is the only guardian of $\mathcal{G}$, and denoted by g.

For example, it follows immediately from the definition above that $\alpha_a \cap \beta_a = \emptyset$ for each node $(a : \Gamma_a \overset{\alpha_a}{\Rightarrow} \Delta_a \uparrow \Sigma_a \overset{\beta_a}{\Rightarrow} \Pi_a)$ of a TSg.

Omitting the condition which involves availability of variables, we obtain the definition of *pre-TSg* of **Tg**N3[l]o, pTSg in short. Again pTSg's emerge as subtrees of TSg's.

In what follows we denote a TSg in the same way as we do a TS, using [,] and |.

Definition 4.12 (TSg system TgN3[d][l]o**)** Initial TSg's and derivation rules of **Tg**N3o are presented below, ahead of those for its variants, followed by some remarks:

$$\frac{}{\cdots[A \overset{\alpha}{\Rightarrow} A \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi \mid \cdots}\ (\mathrm{Id}) \qquad \frac{}{\cdots[A, {\sim}A \overset{\alpha}{\Rightarrow} \ \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi \mid \cdots}\ (\mathrm{Fal})$$

$$\frac{}{\cdots[\Gamma \overset{\alpha}{\Rightarrow} \Delta \uparrow A \overset{\alpha}{\Rightarrow} A \mid \cdots}\ (\mathrm{gId}) \qquad \frac{}{\cdots[\Gamma \overset{\alpha}{\Rightarrow} \Delta \uparrow A, {\sim}A \overset{\alpha}{\Rightarrow} \ \mid \cdots}\ (\mathrm{gFal})$$

$$\frac{}{\cdots[\Gamma \overset{\alpha}{\Rightarrow} \Delta \uparrow \overset{\beta}{\Rightarrow} A \vee {\sim}A \mid \cdots}\ (\mathrm{gOm})$$

TgN3o has all the derivation rules of **T**N3, which are applied only to left sequents:

(W), (D), ($\wedge$L), ($\wedge$R), ($\rightarrow$L), ($\rightarrow$R)$_{\mathrm{T}}$, ($\neg$L), ($\neg$R)$_{\mathrm{T}}$, ($\forall$L), ($\forall$R)$_{\mathrm{T}}$, ($\sim\wedge$L), ($\sim\wedge$R), ($\sim\rightarrow$L), ($\sim\rightarrow$R), ($\sim\neg$L), ($\sim\neg$R), ($\sim\sim$L), ($\sim\sim$R), ($\sim\forall$L)$_{\mathrm{VC}}$ and ($\sim\forall$R).

For example, (W) of **TgN3o** is: $\dfrac{\cdots[\Gamma \overset{\alpha}{\Rightarrow} \Delta \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi \mid \cdots}{\cdots[\Gamma', \Gamma \overset{\alpha}{\Rightarrow} \Delta, \Delta' \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi \mid \cdots}$ (W)

(S) is a structural rule unique to TSg's; considering the TS-translations, (S) can be regarded as a special case of (D).

$$\frac{\cdots[\Gamma \overset{\alpha}{\Rightarrow} \Delta \uparrow A, \Sigma \overset{\beta}{\Rightarrow} \Pi \mid \cdots}{\cdots[A, \Gamma \overset{\alpha}{\Rightarrow} \Delta \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi \mid \cdots} \text{ (Slide, S)}$$

The rules (g$\cdots$) involve only guardians but no left sequents; for them we indicate only guardians.

$$\frac{\cdots \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi \mid \cdots}{\cdots \uparrow \Sigma', \Sigma \overset{\beta}{\Rightarrow} \Pi, \Pi' \mid \cdots} \text{ (gW)} \qquad \frac{\cdots \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi, A \mid \cdots \quad \cdots \uparrow A, \Sigma \overset{\beta}{\Rightarrow} \Pi \mid \cdots}{\cdots \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi \mid \cdots} \text{ (gC)}$$

$$\frac{\cdots \uparrow A, B, \Sigma \overset{\beta}{\Rightarrow} \Pi \mid \cdots}{\cdots \uparrow A \wedge B, \Sigma \overset{\beta}{\Rightarrow} \Pi \mid \cdots} \text{ (g}\wedge\text{L)} \qquad \frac{\cdots \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi, A \mid \cdots \quad \cdots \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi, B \mid \cdots}{\cdots \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi, A \wedge B \mid \cdots} \text{ (g}\wedge\text{R)}$$

$$\frac{\cdots \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi, A \mid \cdots \quad \cdots \uparrow B, \Pi \overset{\beta}{\Rightarrow} \Sigma \mid \cdots}{\cdots \uparrow A \to B, \Sigma \overset{\beta}{\Rightarrow} \Pi \mid \cdots} \text{ (g}\to\text{L)} \qquad \frac{\cdots \uparrow A, \Sigma \overset{\beta}{\Rightarrow} \Pi, B \mid \cdots}{\cdots \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi, A \to B \mid \cdots} \text{ (g}\to\text{R)}$$

$$\frac{\cdots \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi, A \mid \cdots}{\cdots \uparrow \neg A, \Sigma \overset{\beta}{\Rightarrow} \Pi \mid \cdots} \text{ (g}\neg\text{L)} \qquad \frac{\cdots \uparrow A, \Sigma \overset{\beta}{\Rightarrow} \Pi \mid \cdots}{\cdots \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi, \neg A \mid \cdots} \text{ (g}\neg\text{R)}$$

$$\frac{\cdots \uparrow A[y/x], \Sigma \overset{\beta}{\Rightarrow} \Pi \mid \cdots}{\cdots \uparrow \forall x A, \Sigma \overset{\beta}{\Rightarrow} \Pi \mid \cdots} \text{ (g}\forall\text{L)} \qquad \frac{\cdots \uparrow \Sigma \overset{\beta\cup\{z\}}{\Rightarrow} \Pi, A[z/x] \mid \cdots}{\cdots \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi, \forall x A \mid \cdots} \text{ (g}\forall\text{R)}_{\text{VC}}$$

$$\frac{\cdots \uparrow \sim A, \Sigma \overset{\beta}{\Rightarrow} \Pi \mid \cdots \quad \cdots \uparrow \sim B, \Sigma \overset{\beta}{\Rightarrow} \Pi \mid \cdots}{\cdots \uparrow \sim(A \wedge B), \Sigma \overset{\beta}{\Rightarrow} \Pi \mid \cdots} \text{ (g}\sim\wedge\text{L)}$$

$$\frac{\cdots \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi, \sim A, \sim B \mid \cdots}{\cdots \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi, \sim(A \wedge B) \mid \cdots} \text{ (g}\sim\wedge\text{R)}$$

$$\frac{\cdots \uparrow A, \sim B, \Sigma \overset{\beta}{\Rightarrow} \Pi \mid \cdots}{\cdots \uparrow \sim(A \to B), \Sigma \overset{\beta}{\Rightarrow} \Pi \mid \cdots} \text{ (g}\sim\to\text{L)}$$

$$\frac{\cdots \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi, A \mid \cdots \quad \cdots \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi, \sim B \mid \cdots}{\cdots \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi, \sim(A \to B) \mid \cdots} \text{ (g}\sim\to\text{R)}$$

$$\frac{\cdots \uparrow A, \Sigma \overset{\beta}{\Rightarrow} \Pi \mid \cdots}{\cdots \uparrow \sim\neg A, \Sigma \overset{\beta}{\Rightarrow} \Pi \mid \cdots} \text{ (g}\sim\neg\text{L)} \qquad \frac{\cdots \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi, A \mid \cdots}{\cdots \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi, \sim\neg A \mid \cdots} \text{ (g}\sim\neg\text{R)}$$

$$\frac{\cdots \uparrow A, \Sigma \overset{\beta}{\Rightarrow} \Pi \mid \cdots}{\cdots \uparrow \sim\sim A, \Sigma \overset{\beta}{\Rightarrow} \Pi \mid \cdots}\ (\mathrm{g}\sim\sim\mathrm{L}) \qquad \frac{\cdots \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi, A \mid \cdots}{\cdots \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi, \sim\sim A \mid \cdots}\ (\mathrm{g}\sim\sim\mathrm{R})$$

$$\frac{\cdots \uparrow \sim A[z/x], \Sigma \overset{\beta\cup\{z\}}{\Rightarrow} \Pi \mid \cdots}{\cdots \uparrow \sim\forall x A, \Sigma \overset{\beta}{\Rightarrow} \Pi \mid \cdots}\ (\mathrm{g}\sim\forall\mathrm{L})_{\mathrm{VC}} \qquad \frac{\cdots \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi, \sim A[y/x] \mid \cdots}{\cdots \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi, \sim\forall x A \mid \cdots}\ (\mathrm{g}\sim\forall\mathrm{R})$$

It may be seen that the logical rules applied to guardians are just as those of **LK**; it is reasonable in view of the fact that a guardian sequent is a seed of a guardian possible world, which is omniscient and maximal in the tree-structure.

For variants, the following modifications are made:

d Replace the rule $(\forall\mathrm{R})_{\mathrm{T}}$ by $(\forall\mathrm{R})_{\mathrm{VC}}$.

l $(\to\mathrm{R})_{\mathrm{T}}$, $(\neg\mathrm{R})_{\mathrm{T}}$ and $(\forall\mathrm{R})_{\mathrm{T}}$ are replaced by $(\to\mathrm{R})_{\mathrm{K}}$, $(\neg\mathrm{R})_{\mathrm{K}}$ and $(\forall\mathrm{R})_{\mathrm{K}}$, respectively. And (S) is replaced by

$$\frac{\cdots \mid \Gamma \overset{\alpha}{\Rightarrow} \Delta \mid \cdots \uparrow A, \Sigma \overset{\beta}{\Rightarrow} \Pi}{\cdots \mid A, \Gamma \overset{\alpha}{\Rightarrow} \Delta \mid \cdots \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi}\ (\mathrm{S})$$

First we prove Kripke completeness of the TSg systems.

Definition 4.13 (Counter model for TSg)
An N3[d][l]o-model $\mathcal{M}$ is a *counter model for* a TSg $\mathcal{G}$ of **Tg**N3[d][l]o if $\mathcal{M}$ is a counter model for the TS-translation $\mathcal{G}^t$ in the sense of Definition 3.7.

Definition 4.14 (TgN3[d][l]o**-saturatedness)**
An infinite TSg $\mathcal{G}$ is **Tg**N3[d][l]o-*saturated* if it satisfies the following conditions:

1. the TS-translation, $\mathcal{G}^t$, is **T**N3[d][l]-saturated;
2. for each guardian $\Sigma \overset{\beta}{\Rightarrow} \Pi$ and every atomic formula $p(\vec{x})$, if $\vec{x}$ are available there, then either $p(\vec{x})$ or $\sim p(\vec{x})$ is in Σ.

The second condition is necessary for a guardian to induce an omniscient possible world.

Lemma 4.15 (Kripke completeness of TgN3[d][l]o**)** Let **Tg**N3[d][l]o $\nvdash \mathcal{G}$ and at least one variable is l-available at the root of $\mathcal{G}$. Then $\mathcal{G}$ has a counter model.

Proof. We extend $\mathcal{G}$ into a **Tg**N3[d][l]o-saturated infinite TSg step by step, just as the proof of Lemma 3.9. Let $A_1, A_2, \ldots$ the same sequence, $x_1, x_2, \ldots$ also, and $\mathcal{G}_0 :\equiv \mathcal{G}$. The operation done in the i-th step, the step from $\mathcal{G}_{i-1}$ to $\mathcal{G}_i$ as to the formula A_i, is as follows. Again note that unprovability is preserved in each operation.

1. Apply the same operations [inheritance] and [reduction] to the left-sequents of $\mathcal{G}$. For example, if $(a : \Gamma_a \overset{\alpha_a}{\Rightarrow} \Delta_a \uparrow \Sigma_a \overset{\beta_a}{\Rightarrow} \Pi_a)$ and $A_i \in \Gamma_a$, then add A_i to the antecedent of the left sequent of each successor of a, not involving guardians;
2. [slide] For each node $(a : \Gamma_a \overset{\alpha_a}{\Rightarrow} \Delta_a \uparrow \Sigma_a \overset{\beta_a}{\Rightarrow} \Pi_a)$, if $A_i \in \Gamma_a$ then add A_i to Σ_a. This operation preserves unprovability because of the rule (S), and can again be regarded as a special case of [inheritance], considering $\mathcal{G}^t$. For those logics with l, the operation is as follows: if $\cdots \mid \Gamma \overset{\alpha}{\Rightarrow} \Delta \mid \cdots \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi$ and $A_i \in \Gamma$, then add A_i to Σ;

3. [g-reduction] Reduction of A_i which appears in guardians. According to the shape of A_i, one of the following operations is executed to each node $(a : \Gamma_a \overset{\alpha_a}{\Rightarrow} \Delta_a \uparrow \Sigma_a \overset{\beta_a}{\Rightarrow} \Pi_a)$, or $\uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi$ for those logics with l:

(a) $[A_i \equiv p(\vec{x})]$ If $\vec{x}$ are g-available at a, then add $p(\vec{x})$ or $\sim p(\vec{x})$ to Σ_a, so that unprovability is preserved. The choice is possible: if not, we can derive a contradiction as follows.

$$\dfrac{\dfrac{\cdots \uparrow p(\vec{x}), \Sigma_a \overset{\beta_a}{\Rightarrow} \Pi_a \mid \cdots \quad \cdots \uparrow \sim p(\vec{x}), \Sigma_a \overset{\beta_a}{\Rightarrow} \Pi_a \mid \cdots}{\cdots \uparrow p(\vec{x}) \vee \sim p(\vec{x}), \Sigma_a \overset{\beta_a}{\Rightarrow} \Pi_a \mid \cdots}\,(\mathrm{g}\vee\mathrm{L})}{\cdots \uparrow \Sigma_a \overset{\beta_a}{\Rightarrow} \Pi_a \mid \cdots}\,(\mathrm{gC})\quad (\mathrm{gOm})$$

(b) $[A_i \equiv B \wedge C]$ If $A_i \in \Sigma_a$, then add both B and C to Σ_a. Unprovability is preserved by the rule (g$\wedge$L). If $A_i \in \Pi_a$, then add B or C to Π_a, so that unprovability is preserved. This is possible by (g$\wedge$R);

(c) $[A_i \equiv B \to C]$ If $A_i \in \Sigma_a$, then add B to Π_a or C to Σ_a, so that unprovability is preserved. If $A_i \in \Pi_a$, then add B to Σ_a and C to Π_a;

(d) $[A_i \equiv \neg B]$ If $A_i \in \Sigma_a$ (or Π_a), then add B to Π_a (or Σ_a, respectively);

(e) $[A_i \equiv \forall x B]$ If $A_i \in \Sigma_a$, then add $B[y/x]$ to Σ_a, for every y which is g-available at a and is in $\{x_1, \ldots, x_i\}$. If $A_i \in \Pi_a$, then take a fresh variable x_m, add $A[x_m/x]$ to Π_a and also add x_m to β_a;

(f) $[A_i \equiv \sim(B \wedge C)]$ If $A_i \in \Sigma_a$, add $\sim B$ or $\sim C$ to Σ_a, so that unprovability is preserved. If $A_i \in \Pi_a$, add $\sim B$ and $\sim C$ to Π_a;

(g) $[A_i \equiv \sim(B \to C)]$ If $A_i \in \Sigma_a$, add B and $\sim C$ to Σ_a. If $A_i \in \Pi_a$, add B or $\sim C$ to Π_a so that unprovability is preserved;

(h) $[A_i \equiv \sim\neg B$ or $A_i \equiv \sim\sim B]$ If $A_i \in \Sigma_a$ (or Π_a), add B to Σ_a (or Π_a, respectively);

(i) $[A_i \equiv \sim\forall x B]$ If $A_i \in \Sigma_a$, then take a fresh x_m, add $\sim B[x_m/x]$ to Σ_a, and also add x_m to β_a. If $A_i \in \Pi_a$, add $\sim B[y/x]$ to Π_a for every y which is g-available at a and in $\{x_1, \ldots, x_i\}$.

For $\mathbf{Tg}\mathsf{N3d}[\mathsf{l}]\mathsf{o}$, omit conditions which involve availability of variables.

Let $\mathcal{G}_\omega$ be the union of $\mathcal{G}_0, \mathcal{G}_1, \ldots$. Then $\mathcal{G}_\omega$ is $\mathbf{Tg}\mathsf{N3}[\mathsf{d}][\mathsf{l}]\mathsf{o}$-saturated: condition 1. is easily verified, and 2. is by the operation (3a).

Since an infinite TS ${\mathcal{G}_\omega}^t$ is $\mathbf{T}\mathsf{N3}[\mathsf{d}][\mathsf{l}]$-saturated, it induces an $\mathsf{N3}[\mathsf{d}][\mathsf{l}]$-model $\mathcal{M}$ in the same way as the proof of Lemma 3.9. Moreover, condition 2. of $\mathbf{Tg}\mathsf{N3}[\mathsf{d}][\mathsf{l}]\mathsf{o}$-saturatedness yields that $\mathcal{M}$ is actually an $\mathsf{N3}[\mathsf{d}][\mathsf{l}]\mathsf{o}$-model, where the omniscient world a_g for each world a (or the only omniscient world g for those logics with l) is induced by a_g of ${\mathcal{G}_\omega}^t$ (or g of it, respectively). By its construction $\mathcal{M}$ is a counter model for $\mathcal{G}$, which completes the proof. ∎

Now we introduce the formulaic translation of a TSg, and with the lemma above conclude Kripke completeness of $\mathbf{G}\mathsf{N3}[\mathsf{d}][\mathsf{l}]\mathsf{o}$.

Definition 4.16 (Formulaic translation of TSg) The *formulaic translation* of a pre-TSg $\mathcal{G}$ of $\mathbf{Tg}\mathsf{N3o}$, denoted by $\mathcal{G}^f$, is defined inductively on the height of $\mathcal{G}$:

$$[\Gamma \overset{\alpha}{\Rightarrow} \Delta \uparrow \Sigma \overset{\beta}{\Rightarrow} \Pi \mid \mathcal{G}_1 \ldots \mathcal{G}_m]^f$$
$$:\equiv \forall\overrightarrow{\alpha}\Big(\big(\bigwedge \Gamma\big) \to \big(\bigvee \Delta\big) \ \vee\ \forall\overrightarrow{\beta}\neg\neg\Big(\big(\bigwedge \Sigma\big) \to \big(\bigvee \Pi\big) \Big) \ \vee \mathcal{G}_1^f \vee \cdots \vee \mathcal{G}_m^f \Big).$$

The *formulaic translation* of a TSg $\mathcal{G}$ of **Tg**N3do, again denoted by $\mathcal{G}^f$, is a universal closure of $\mathcal{G}^p$, which in turn is defined inductively on the height of $\mathcal{G}$:

$$[\Gamma \Rightarrow \Delta \uparrow \Sigma \Rightarrow \Pi \mid \mathcal{G}_1 \dots \mathcal{G}_m]^p$$
$$:\equiv (\bigwedge \Gamma) \to (\bigvee \Delta) \ \vee \ \neg\neg\Big((\bigwedge \Sigma) \to (\bigvee \Pi) \Big) \ \vee \mathcal{G}_1^p \vee \dots \vee \mathcal{G}_m^p.$$

For a pre-TSg of **Tg**N3lo, its formulaic translation is defined inductively by:

$$(\Gamma \overset{\alpha}{\Rightarrow} \Delta \uparrow \Sigma \overset{\beta}{\Rightarrow} \Delta)^f :\equiv \forall\overrightarrow{\alpha}\Big((\bigwedge \Gamma) \to (\bigvee \Delta) \vee \forall\overrightarrow{\beta}\neg\neg\big((\bigwedge \Sigma) \to (\bigvee \Pi) \big) \Big),$$
$$(\Gamma \overset{\alpha}{\Rightarrow} \Delta \mid \mathcal{G})^f :\equiv \forall\overrightarrow{\alpha}\big((\bigwedge \Gamma) \to (\bigvee \Delta) \vee \mathcal{G}^f \big).$$

For a pre-TSg of **Tg**N3dlo, $\mathcal{G}^p$ is defined as follows, and $\mathcal{G}^f$ is its universal closure:

$$(\Gamma \Rightarrow \Delta \uparrow \Sigma \Rightarrow \Pi)^p :\equiv (\bigwedge \Gamma) \to (\bigvee \Delta) \vee \neg\neg\big((\bigwedge \Sigma) \to (\bigvee \Pi) \big),$$
$$(\Gamma \Rightarrow \Delta \mid \mathcal{G})^f :\equiv (\bigwedge \Gamma) \to (\bigvee \Delta) \vee \mathcal{G}^f.$$

Lemma 4.17 If **Tg**N3[d][l]o $\vdash \mathcal{G}$, then **G**N3[d][l]o $\vdash \mathcal{G}^f$.

Proof. Since the counterparts of lemma 3.12 and 3.13 are easily verified, we can assume that the node to which a derivation rule is applied (or the node which is in the form indicated in an initial TSg) is nothing but the root. Now we prove the lemma by the induction on the derivation of $\mathcal{G}$ in **Tg**N3[d][l]o.

The cases for (Id), (Fal), (gId) and (gFal) are easy.

For the (gOm), by (Om) of **G**N3[d][l]o.

For the rules which are common in **T**N3[d] and **Tg**N3[d]o such as (D) or ($\sim\forall$R), the proof is just as that of lemma 3.11.

For the remaining rules involving guardians, we present only the proofs for complicated cases here. First we prove the following fact needed later:

$$\mathsf{Int}\ (\text{hence } \mathbf{G}\mathsf{N}_4^3[\mathsf{d}][\mathsf{l}][\mathsf{o}]) \vdash \neg\neg(A \vee \neg A) \qquad (\star)$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{A \Rightarrow A}{A \Rightarrow A \vee \neg A}\,(\vee\mathrm{R})}{A, \neg(A \vee \neg A) \Rightarrow}\,(\neg\mathrm{L})}{\neg(A \vee \neg A) \Rightarrow \neg A}\,(\neg\mathrm{R})}{\neg(A \vee \neg A) \Rightarrow A \vee \neg A}\,(\vee\mathrm{R})}{\neg(A \vee \neg A), \neg(A \vee \neg A) \Rightarrow}\,(\neg\mathrm{L})}{\Rightarrow \neg\neg(A \vee \neg A)}\,(\neg\mathrm{R})$$

For (g$\neg$R), it suffices to show that **G**N3[d][l]o $\vdash \neg\neg(C \wedge A \to D) \Rightarrow \neg\neg(C \to D \vee \neg A)$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{C, A, C \wedge A \to D \Rightarrow D \vee \neg A} \quad \dfrac{\vdots}{C, \neg A, C \wedge A \to D \Rightarrow D \vee \neg A}\,(\vee\mathrm{L})}{C, A \vee \neg A, C \wedge A \to D \Rightarrow D \vee \neg A}\,(\to\mathrm{R})}{A \vee \neg A, C \wedge A \to D \Rightarrow C \to D \vee \neg A}}{\neg\neg(A \vee \neg A), \neg\neg(C \wedge A \to D) \Rightarrow \neg\neg(C \to D \vee \neg A)}\,(\neg\mathrm{L})(\neg\mathrm{R})}{\neg\neg(C \wedge A \to D) \Rightarrow \neg\neg(C \to D \vee \neg A)}\,(\mathrm{C})\ (\star)$$

For (g→R), the proof is similar to above, using ($\star$).
For (g∀R), it suffices to show that

$$\mathbf{G}\mathsf{N3}[\mathsf{d}][\mathsf{l}]\mathsf{o} \vdash \forall z\neg\neg(C \to D \vee A[z/x]) \Rightarrow \neg\neg(C \to D \vee \forall x A)$$

where z is free in neither C nor D.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{C \Rightarrow C}\,(\mathrm{Id}) \quad \dfrac{\dfrac{}{D, \sim D \Rightarrow}\,(\mathrm{Fal}) \quad \dfrac{}{A[z/x], \sim A[z/x] \Rightarrow}\,(\mathrm{Fal})}{D \vee A[z/x], \sim D, \sim A[z/x] \Rightarrow}\,(\vee\mathrm{L})}{C \to D \vee A[z/x], C, \sim D, \sim A[z/x] \Rightarrow}\,(\to\mathrm{L})}{\forall z\neg\neg(C \to D \vee A[z/x]), C, \sim D, \sim A[z/x] \Rightarrow}\,(\neg\mathrm{R})(\neg\mathrm{L})(\forall\mathrm{L})}{\forall z\neg\neg(C \to D \vee A[z/x]), C, \sim D, \sim \forall x A \Rightarrow}\,(\sim\forall\mathrm{L})_{\mathrm{VC}}}{\forall z\neg\neg(C \to D \vee A[z/x]), \sim(C \to D \vee \forall x A) \Rightarrow}\,(\sim\vee\mathrm{L})(\sim\to\mathrm{L})}{\forall z\neg\neg(C \to D \vee A[z/x]), \neg(C \to D \vee \forall x A) \Rightarrow}\,(\mathrm{Om2})}{\forall z\neg\neg(C \to D \vee A[z/x]) \Rightarrow \neg\neg(C \to D \vee \forall x A)}\,(\neg\mathrm{R})$$

For (S), it suffices to show that

$$\mathbf{G}\mathsf{N3}[\mathsf{l}]\mathsf{o} \vdash \forall\vec{x}\big(C \to D \vee \forall\vec{y}\neg\neg(A \wedge E \to F)\big) \Rightarrow \forall\vec{x}\big(A \wedge C \to D \vee \forall\vec{y}\neg\neg(E \to F)\big)$$

where $\vec{y}$ have no free occurrences in C, D or A. This is easy. ∎

Theorem 4.18 (Kripke completeness of G$\mathsf{N3}[\mathsf{d}][\mathsf{l}]\mathsf{o}$**)**
If $\mathsf{N3}[\mathsf{d}][\mathsf{l}]\mathsf{o} \models A$, then $\mathbf{G}\mathsf{N3}[\mathsf{d}][\mathsf{l}]\mathsf{o} \vdash A$.

Proof. Similar to that of Theorem 3.14. Take $\mathcal{G} :\equiv [\quad \overset{\alpha}{\Rightarrow} A \uparrow \overset{\emptyset}{\Rightarrow}]$, and use Lemma 4.15 and 4.17. ∎

5 Remarks on logics with $\mathsf{N4}[\mathsf{d}][\mathsf{l}]\mathsf{o}$

As stated in the introduction, Kripke completeness of logics $\mathsf{N4}[\mathsf{d}][\mathsf{l}]\mathsf{o}$ is remain unproved in this paper. Here we are to show that their proofs cannot be done using our methods.

The key is an axiom $\forall x\neg\neg A \to \neg\neg\forall x A$, called the *double negation shift*, *DNS* in short. This is a theorem of those logics $\mathsf{N3}[\mathsf{d}][\mathsf{l}]\mathsf{o}$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A[z/x], \sim A[z/x] \Rightarrow}\,(\mathrm{Fal})}{\forall x\neg\neg A, \sim A[z/x] \Rightarrow}\,(\neg\mathrm{R}), (\neg\mathrm{L}), (\forall\mathrm{L})}{\forall x\neg\neg A, \sim\forall x A \Rightarrow}\,(\sim\forall\mathrm{L})}{\forall x\neg\neg A, \neg\forall x A \Rightarrow}\,(\mathrm{Om2})}{\Rightarrow \forall x\neg\neg A \to \neg\neg\forall x A}\,(\neg\mathrm{R}), (\to\mathrm{R})$$

On the other hand, DNS is not provable in even the strongest system among $\mathbf{G}\mathsf{N4}[\mathsf{d}][\mathsf{l}]\mathsf{o}$, i.e. $\mathbf{G}\mathsf{N4dlo}$, since it has the following counter $\mathsf{N4dlo}$-model $\mathcal{M} = (M, \leq, U, I^+, I^-)$: $(M, \leq) = (\omega, \leq)$, $U(n) = \omega$ for every $n \in \omega$, p a unary predicate symbol, $p^{I^+(n)} = \{1, 2, \ldots, n\}$ and $p^{I^-(n)} = \omega$. Then $\mathcal{M} \not\models \forall x\neg\neg p(x) \to \neg\neg\forall x p(x)$.

The intermediate logic MH, which is Int plus DNS, is characterized by the class of Int-models such that: for each possible world a, there exists $b \geq a$ which is maximal $(*)$ [5]. Counter models which the two methods in this paper construct all have the property $(*)$; omniscient worlds are always maximal. However, since the axiom DNS does not involve strong negation, DNS is valid in every N4-model (hence also in every $\mathsf{N4}^3_4$[d][l][o]-model) which has the property $(*)$. Hence we cannot construct a counter N4o-model for DNS by the methods.

Acknowledgements

The authors are grateful to the referee for his comments and detailed historical remarks. The first author would like to thank Norihiro KAMIDE, Izumi TAKEUTI, Takeshi YAMAGUCHI and Tatsuya SHIMURA for their invaluable comments and encouragements.

References

[1] Seiki Akama. Tableaux for logic programming with strong negation. In Didier Galmiche, editor, *Automated Reasoning with Analytic Tableaux and Related Methods, International Conference, TABLEAUX'97*, volume 1227 of *Lecture Notes in Computer Science*, pages 31–42. Springer-Verlag, May 1997.

[2] Ahmad Almukdad and David Nelson. Constructible falsity and inexact predicates. *The Journal of Symbolic Logic*, 49(1):231–233, 1984.

[3] Giovanna Corsi. Completeness theorem for Dummett's LC quantified and some of its extensions. *Studia Logica*, 51:317–335, 1992.

[4] Giovanna Corsi and Silvio Ghilardi. Directed frames. *Archive for Mathematical Logic*, 29:53–67, 1989.

[5] Dov M. Gabbay. *Semantical Investigations in Heyting's Intuitionistic Logic*, volume 148 of *Synthese Library*. D. Reidel, 1981.

[6] Valentine Goranko. The craig interpolation theorem for propositional logics with strong negation. *Studia Logica*, 44:291–317, 1985.

[7] Sabine Görnemann. A logic stronger than intuitionism. *The Journal of Symbolic Logic*, 36:249–261, 1971.

[8] Andrzej Grzegorczyk. A philosophically plausible formal interpretation of intuitionistic logic. *Indagationes Mathematicae*, 26:596–601, 1964.

[9] Yuri Gurevich. Intuitionistic logic with strong negation. *Studia Logica*, 36:49–59, 1977.

[10] Heinrich Herre and David Pearce. Disjunctive logic programming, constructivity and strong negation. In *Logic in AI. Proceedings of the European Workshop JELIA 92*, volume 633 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1992.

[11] Arend Heyting. *Intuitionism. An Introduction*. North-Holland, 1956.

[12] Ryo Kashima. Sequent calculi of non-classical logics – Proofs of completeness theorems by sequent calculi (in Japanese). In *Proceedings of Mathematical Society of Japan Annual Colloquium of Foundations of Mathematics*, pages 49–67, 1999.

[13] Marcus Kracht. On extensions of intermediate logics by strong negation. *Journal of Philosophical Logic*, 27:49–73, 1998.

[14] Saul A. Kripke. Semantical analysis of intuitionistic logic I. In J. Crossley and M. Dummett, editors, *Formal Systems and Recursive Functions*, pages 92–129. North-Holland, 1965.

[15] A. A. Markov. Konstruktivnaja logika. *Uspehi Matematičeskih Nauk*, 5:187–188, 1950.

[16] David Nelson. Constructible falsity. *The Journal of Symbolic Logic*, 14:16–26, 1949.

[17] Sergei P. Odintsov. Algebraic semantics for paraconsistent Nelson's logic. *Journal of Logic and Computation*, 13(4):453–468, 2003.

[18] Sergei P. Odintsov and Heinrich Wansing. Inconsistency-tolerant description logic, Motivation and basic systems. In Vincent F. Hendricks and Jacek Malinowski, editors, *Trends in Logic, 50 Years of Studia Logica*, volume 21 of *Trends in Logic*, pages 301–335. Kluwer Academic Publishers, Dordrecht, 2003.

[19] David Pearce and Gerd Wagner. Reasoning with negative information, I: Strong negation in logic programs. In *Language, Knowledge and Intentionality (Acta Philosophica Fennica 49)*, pages 405–439, 1990.

[20] David Pearce and Gerd Wagner. Logic programming with strong negation. In Peter Schroeder-Heister, editor, *Extensions of Logic Programming*, volume 475 of *Lecture Notes in Artificial Intelligence*, pages 311–326. Springer-Verlag, 1991.

[21] Graham Priest and Richard Routley. Introduction: Paraconsistent logics. *Studia Logica*, 43:3–16, 1984.

[22] Andrzej Sendlewski. Some investigations of varieties of N-lattices. *Studia Logica*, 43:257–280, 1984.

[23] Andrzej Sendlewski. Nelson algebras through Heyting ones. *Studia Logica*, 49:106–126, 1990.

[24] Andrzej Sendlewski. Axiomatic extensions of the constructive logic with strong negation and the disjunction property. *Studia Logica*, 55:377–388, 1995.

[25] Richmond H. Thomason. A semantical study of constructive falsity. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 15:247–257, 1969.

[26] Anne S. Troelstra and Helmut Schwichtenberg. *Basic Proof Theory*. Number 43 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1996.

[27] Anne S. Troelstra and Dirk van Dalen. *Constructivism in Mathematics, Vol. I*. North-Holland, 1988.

[28] Dirk van Dalen. Intuitionistic logic. In D. M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic III*, volume 166 of *Synthese Library*, pages 225–340. D. Reidel, 1986.

[29] Franz von Kutschera. Ein verallgemeinerter Widerlegungsbegriff für Gentzenkalküle. *Archiv für Mathematische Logik und Grundlagenforschung*, 12:104–118, 1969.

[30] Gerd Wagner. *Vivid Logic. Knowledge-Based Reasoning with Two Kinds of Negation*, volume 764 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1994.

[31] Heinrich Wansing. *The Logic of Information Structures*, volume 681 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1993.

[32] Heinrich Wansing. Higher-arity Gentzen systems for Nelson's logics. In Julian Nida-Rümelin, editor, *Rationality, Realism, Revision, Proceedings of the 3rd international congress of the Society for Analytical Philosophy*, volume 23 of *Perspectives in Analytical Philosophy*, pages 105–109. Walter de Gruyter, September 1999.

Received 10 September 2003. Revised 14 November 2003

Automatic Learning of Proof Methods in Proof Planning

MATEJA JAMNIK, *University of Cambridge Computer Laboratory, J.J. Thomson Avenue, Cambridge CB3 0FD, England, UK. E-mail: Mateja.Jamnik@cl.cam.ac.uk.*

MANFRED KERBER, *School of Computer Science, The University of Birmingham, Birmingham B15 2TT, England, UK. URL: http://www.cs.bham.ac.uk/ mmk.*

MARTIN POLLET, *Fachbereich Informatik, Universität des Saarlandes, 66041 Saarbrücken, Germany. E-mail: pollet@ags.uni-sb.de.*

CHRISTOPH BENZMÜLLER, *Fachbereich Informatik, Universität des Saarlandes, 66041 Saarbrücken, Germany. E-mail: chris@ags.uni-sb.de.*

Abstract

In this paper we present an approach to automated learning within mathematical reasoning systems. In particular, the approach enables proof planning systems to automatically learn new proof methods from well-chosen examples of proofs which use a similar reasoning pattern to prove related theorems. Our approach consists of an abstract representation for methods and a machine learning technique which can learn methods using this representation formalism. We present an implementation of the approach within the ΩMEGA proof planning system, which we call LEARNΩMATIC. We also present the results of the experiments that we ran on this implementation in order to evaluate if and how it improves the power of proof planning systems.

Keywords: automated reasoning, theorem proving, proof planning, knowledge acquisition, machine learning

1 Introduction

Proof planning [3] is an approach to theorem proving which uses so-called proof methods rather than low-level logical inference rules to find a proof of a theorem at hand. A proof method specifies a general reasoning pattern that can be used in a proof, and typically expands to a number of individual inference rules. For example, an induction strategy can be encoded as a proof method. Proof planners search for a proof plan of a theorem which consists of applications of several methods. An object-level logical proof may be generated from a successful proof plan. Proof planning is a powerful technique because it often dramatically reduces the search space, since the search is done on the level of abstract methods rather than on the level of several inference rules that make up a method [4, 20]. The advantage is that search with

methods can be much better structured according to the particular requirements of mathematical domains.

Proof planning also allows reuse of the same proof methods for different proofs, and, moreover, generates proofs where the reasoning patterns of proofs are transparent. When methods are designed appropriately, the level of proof plans can capture the level of communication of proofs amongst mathematicians. Hence proof plans can offer an intuitive appeal to a human mathematician.

One of the ways to extend the power of a proof planning system is to enlarge the set of available proof methods. This is particularly beneficial when a class of theorems can be proved in a similar way, hence a new proof method can encapsulate the general reasoning pattern of a proof for such theorems. Methods are typically implemented and added by the developer of a system. The development and encoding of proof methods by hand, however, is a laborious task. In this work, we show how a system can learn new methods automatically given a number of well-chosen (positive) examples of related proofs of theorems. This is a significant improvement, since examples (e.g., in the form of classroom example proofs) exist typically in abundance, while the extraction of methods from these examples can be considered as a major bottleneck of the proof planning methodology.[1] In this paper we therefore present a hybrid proof planning system LEARNΩMATIC [14], which combines the existing proof planner ΩMEGA [1] with our own machine learning system [13]. This enhances the ΩMEGA system with an automated capability to learn new proof methods.

Automated learning by reasoning systems is a difficult and ambitious problem. Our work demonstrates one way of starting to address this problem, and by doing so, it presents several contributions to the field.

1. Although machine learning techniques have been around for a while, they have been relatively little used in reasoning systems. Making a reasoning system learn proof patterns from examples, much like students learn to solve problems from examples demonstrated to them by the teacher, is hard. Our work makes an important step in a specialised domain towards a proof planning system that can reason *and* learn.
2. Proof methods have complex structures, and are hence very hard to learn by the existing machine learning techniques. We approach this problem by abstracting as much information from the proof method representation as needed, so that the machine learning techniques can tackle it. Later, after the reasoning pattern is learnt, the abstracted information is restored as much as possible.
3. Unlike in some of the existing related work (see Section 5), we are not aiming to improve ways of directing proof search within a fixed set of primitives. Rather, we aim to learn the primitives themselves, and to investigate whether this improves the framework and reduces the search space within the proof planning environment. Instead of searching amongst numerous low-level proof methods, a proof planner can now search with a newly learnt proof method which encapsulates several of these low-level primitive methods.

[1] Note that in this paper, we do not provide a systematic and automated way of choosing good examples – in our system, this is still the user's task, which does require some expert knowledge. Choosing good examples automatically is discussed as future work in Section 6.

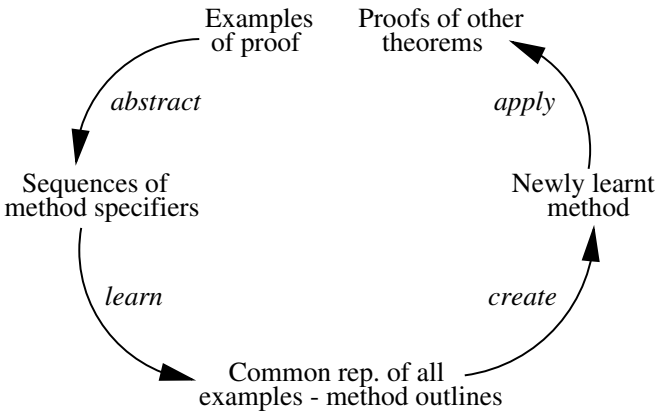


FIG. 1. Approach to learning proof methods.

FIGURE 1 gives the structure of our approach to learning proof methods, and hence an outline of the rest of this paper. In Section 2 we examine what needs to be learnt and give some examples of proofs that use a similar reasoning pattern. Then, in Section 3, we present the entire learning process. First, in Section 3.1, we simplify the method representation to ease the learning task. Second, we present our machine learning algorithm in Section 3.2. Third, in Section 3.3 we revisit our method representation and enrich it so that the newly learnt methods can be used in the ΩMEGA proof planner for proofs of other theorems. In order to assess the success of our approach, we go on in Section 4 to present some results of the evaluation tests that we ran on LEARNΩMATIC. Finally, we relate our work to that of others in Section 5, and conclude with some future directions in Section 6.

2 Motivation with Examples

A proof method in proof planning consists of a triple – preconditions, postconditions and a tactic.[2] A tactic is a program which given that the preconditions are satisfied, transforms an expression representing a subgoal in a way that the postconditions are satisfied by the transformed subgoal. If no method on an appropriate level is available in a given planning state, then a number of lower-level methods (with inference rules corresponding to the lowest-level methods) have to be applied in order to prove a given theorem. It often happens that a pattern of lower-level methods is applied time and time again in proofs of different problems. In this case it is sensible and useful to encapsulate this reasoning pattern in a new proof method. Such a higher-level proof method based on lower-level methods can be implemented and added to the system either by the user or by the developer of the system. However, this is a very knowledge intensive task. Hence, we present an alternative, namely a framework in which these methods can be learnt by the system automatically.

[2]This is an idealised view of a proof method. In practise, postconditions of proof methods are typically determined by executing the tactic part of the methods. So, when we speak of postconditions, it would be more appropriate and precise to speak of the effects of a method.

The idea is that the system starts with learning simple proof methods. As the database of available proof methods grows, the system can learn more complex proof methods. Inference rules can be treated as methods by assigning to them pre- and postconditions. Thus, from a learning perspective we can have a unified view of inference rules and methods as given sequences of primitives from which the system is learning a pattern. We will refer to all the existing methods available for the construction of proofs as primitive methods. As new methods are learnt from primitive methods, these too become primitive methods from which yet more new methods can be learnt. Clearly, there is a trade-off between the increased search space due to a larger number of methods, and increasingly better directed search possibilities for subproofs covered by the learnt methods. Namely, on the one hand, if there are more methods, then the search space is potentially larger. On the other hand, the organisation of a planning search space can be arranged so that the newly learnt, more complex methods are searched with first. If a learnt method is found to be applicable, then instead of a number of planning steps (that correspond to the lower-level methods encapsulated by the learnt method), a proof planner needs to make one step only. On the other hand, if a learnt method is applicable only seldom, then this may have negative effects on some performance criteria of the system (e.g., run time behaviour), but may not negatively affect others (e.g., even in the worst case, when a learnt method is not applicable or does not lead to a valid proof plan, the length of the generated proof plan does not increase, but remains unchanged). Generally, proof plans consisting of higher-level methods will be shorter than their corresponding plans that consist of lower-level methods. Hence, the search for a complete proof plan can be expected to be performed in a shallower, but also bushier search space. In order to measure this trade-off between the increased search space and better directed search, an empirical study was carried out and is reported in Section 4. Typically, shorter proofs have a general advantage, since they are better suited for a user-adaptive presentation. We discuss this in Section 4.5.

The methods that LEARNΩMATIC learns are on a higher-level than the existing ones. Hence, the proofs constructed using them are not overwhelmed with unintuitive low-level proof steps, and can therefore be presented at a more abstract level. In this sense, such proofs reflect a higher-level idea of the proof, and can therefore be viewed as more human-oriented.

We demonstrate our ideas with examples that we used to develop and test LEARNΩMATIC. Most of the example conjectures can be automatically planned for in ΩMEGA with the MULTI proof planner [19]. However, they demonstrate our approach, namely, they show how a proof planner can learn new methods automatically.

2.1 Group theory examples

The proofs of our first set of examples consist of simplifying an expression using a number of primitive simplification methods such as both (left and right) axioms of identity, both axioms of inverse, and the axioms of associativity (where e is the identity element, i is the inverse function, and LHS $\Rightarrow$ RHS stands for rewriting LHS to RHS).

$$\begin{array}{rcll} (X \circ Y) \circ Z & \Rightarrow & X \circ (Y \circ Z) & \mathsf{(assoc\text{-}r)} \\ X \circ (Y \circ Z) & \Rightarrow & (X \circ Y) \circ Z & \mathsf{(assoc\text{-}l)} \\ e \circ X & \Rightarrow & X & \mathsf{(id\text{-}l)} \end{array} \qquad \begin{array}{rcll} X \circ e & \Rightarrow & X & \mathsf{(id\text{-}r)} \\ X \circ X^i & \Rightarrow & e & \mathsf{(inv\text{-}r)} \\ X^i \circ X & \Rightarrow & e & \mathsf{(inv\text{-}l)} \end{array}$$

Here are two examples of proof steps which simplify given expressions and the inferences that are used:

$$\begin{array}{r} a \circ ((a^i \circ c) \circ b) \\ \Downarrow \mathsf{(assoc\text{-}l)} \\ (a \circ (a^i \circ c)) \circ b \\ \Downarrow \mathsf{(assoc\text{-}l)} \\ ((a \circ a^i) \circ c) \circ b \\ \Downarrow \mathsf{(inv\text{-}r)} \\ (e \circ c) \circ b \\ \Downarrow \mathsf{(id\text{-}l)} \\ c \circ b \end{array} \qquad \begin{array}{r} a^i \circ (a \circ b) \\ \Downarrow \mathsf{(assoc\text{-}l)} \\ (a^i \circ a) \circ b \\ \Downarrow \mathsf{(inv\text{-}l)} \\ e \circ b \\ \Downarrow \mathsf{(id\text{-}l)} \\ b \end{array}$$

Other examples include proofs for theorems such as $(a \circ (((a^i \circ b) \circ (c \circ d)) \circ f)) = (b \circ (c \circ d)) \circ f$. These three examples can be summarised in the following proof traces which are lists of method identifiers:

1. [assoc-l,assoc-l,inv-r,id-l],
2. [assoc-l,inv-l,id-l],
3. [assoc-l,assoc-l,assoc-l,inv-r,id-l].

It is clear that all three examples have a similar structure which could be captured in a new simplification method. Informally, one application of such a simplification method could be described as follows:

Precondition: *There are subterms in the initial term that are inverses of each other, and that are not separated by other subterms, but only by brackets.*

Tactic:

1. *Apply associativity* (assoc-l) *for as many times as necessary (including* 0 *times) to bring the subterms which are inverses of each other together, and then*
2. *apply inverse inference rule* (inv-r) *or* (inv-l) *to reduce the expression, and then*
3. *apply the identity inference rule* (id-l).

Postcondition: *The initial term is reduced, i.e., it consists of fewer subterms.*

The formal representation of the learnt method in our framework will be presented in Section 3.2.1.

Note that this is not the most general simplification method, because it does not use methods such as (assoc-r) and (id-r), but it is the one that is the least general generalisation of the given examples above. Note also that the application of this method may fail if the precondition is not strong enough. For instance, two terms

may have to be brought together by the application of the (assoc-r) rule, which is not covered by the learnt method, since no example of this type has been provided. Also, should we want our system to learn a repeated application of this simplification method, then this can be achieved in another round of learning with suitable examples and methods. Alternatively, our set of initial examples that the system is learning from needs to include proofs of theorems such as $(c \circ (b \circ (a^i \circ (a \circ b^i)))) \circ (((d \circ a) \circ a^i) \circ f) = c \circ (d \circ f)$ which applies the above described simplification method three times.

2.2 Residue classes conjectures

There is a large class of residue class theorems in group theory that can be proved using the same pattern of reasoning. Their use is well documented in [18]. Here are examples of three residue class theorems: (where $\mathbf{Z}_i$ is the residue class of integers modulo i)

1. *commutative-under*$(\mathbf{Z}_2, +)$
2. *associative-under*$(\mathbf{Z}_3, \times)$
3. *commutative-under*$(\mathbf{Z}_3, +)$

The pattern of reasoning to prove them is as follows. First, the definitions (e.g., *commutative-under, associative-under,*) are expanded (defn-exp), and quantifiers eliminated ($\forall_i$-sort). Then, all of the statements on residue classes are rewritten into corresponding statements on integers by transferring the residue class set into a set of corresponding integers (convert-resclass-to-num). Then, the proofs diverge: if the statements are universally quantified an exhaustive case analysis over all elements of the set is carried out (using a combination of elimination of disjuncts (or-e-rec), simplification (simp-num-exp), and reflexivity (reflex)). If the statements are existentially quantified, then all elements of the set are examined until one is found for which the statements hold (using a combination of disjunction introduction from left or right (ori-r, ori-l), simplification and reflexivity; see choose method in Section 3.2.1). Note that the three example theorems above are all universally quantified, but the set of theorems used in the evaluation tests (see Section 4) contains the existentially quantified theorems as well.

The proof trace for the above three theorems consist of a list of method identifiers used in the proof plans:

1. [defn-exp, $\forall_i$-sort, $\forall_i$-sort, convert-resclass-to-num, or-e-rec, simp-num-exp, simp-num-exp, simp-num-exp, simp-num-exp, reflex, reflex, reflex, reflex]
2. [defn-exp, $\forall_i$-sort, $\forall_i$-sort, $\forall_i$-sort, convert-resclass-to-num, or-e-rec, simp-num-exp, reflex, . . . , reflex]
3. [defn-exp, $\forall_i$-sort, $\forall_i$-sort, convert-resclass-to-num, defn-exp, or-e-rec, simp-num-exp, simp-num-exp, simp-num-exp, simp-num-exp, simp-num-exp, simp-num-exp, simp-num-exp, simp-num-exp, simp-num-exp, reflex, reflex, reflex, reflex, reflex, reflex, reflex, reflex, reflex]

The learnt generalisations for these proof traces are presented in Section 3.2.1.

2.3 Set theory conjectures

Another problem domain that we experimented with includes some theorems and non-theorems from set theory:

1. $\forall x, y, z_{\bullet}(x \cup y) \cap z = (x \cap z) \cup (y \cap z)$
2. $\forall x, y, z_{\bullet}(x \cup y) \cap z = (x \cup z) \cap (y \cup z)$
3. $\forall x, y, z_{\bullet}(x \cap y) \cap z = x \backslash (y \cup z)$

Although these problems are not very hard for automated theorem provers if a suitable representation is chosen, they may be hard to prove or disprove for existing automated theorem provers if attempted in a naive way. Their proofs consist of eliminating (introducing, in backwards reasoning) the universal quantifiers ($\forall_{\mathsf{i}}$), then applying set extensionality (set-ex) and definition expansions (defni) in order to get propositional or first order clauses (i.e., transforming statements about sets to statements about elements of sets), and then proving (with the Otter theorem prover, atp-otter) or disproving (with the Satchmo model generator, counterex-satchmo) these clauses. Here are the abstracted lists of method identifiers that describe these proofs:

1. $[\forall_{\mathsf{i}}, \forall_{\mathsf{i}}, \forall_{\mathsf{i}}, \mathsf{set\text{-}ext}, \forall_{\mathsf{i}}, \mathsf{defni}, \mathsf{defni}, \mathsf{atp\text{-}otter}]$
2. $[\forall_{\mathsf{i}}, \forall_{\mathsf{i}}, \forall_{\mathsf{i}}, \mathsf{set\text{-}ext}, \forall_{\mathsf{i}}, \mathsf{defni}, \mathsf{defni}, \mathsf{counterex\text{-}satchmo}]$
3. $[\forall_{\mathsf{i}}, \forall_{\mathsf{i}}, \forall_{\mathsf{i}}, \mathsf{set\text{-}ext}, \forall_{\mathsf{i}}, \mathsf{defni}, \mathsf{defni}, \mathsf{defni}, \mathsf{counterex\text{-}satchmo}]$

The learnt generalisations for these proof traces are presented in Section 3.2.1.

3 Learning

The representation of a problem is of crucial importance for the ability to solve it – a good representation of a problem often renders the search for its solution easy [25]. The difficulty is in finding a good representation. Our problem is to devise a mechanism for learning methods. Hence, the representation of a method is important and should make the learning process easy enough that we can learn useful information.

We start by presenting in Section 3.1 a simple representation formalism which abstracts away some detailed information in order to ease the learning process. Then, in Section 3.2 we describe the learning algorithm. Finally, we show in Section 3.3 how the necessary information is restored as much as possible so that the proof planner can use the newly learnt method. Some information may be irrecoverably lost. In this case, extra search in the application of the newly learnt methods will typically be necessary.

3.1 Method outline representation

The methods we aim to learn are complex and are beyond the complexity that can typically be tackled in the field of machine learning. Therefore, we first simplify the problem and aim to learn (using a variation of an existing learning technique) the

so-called *method outlines*, and second, we reconstruct the full information as far as possible. Method outlines are expressed in the language that we describe here.

Let us define the following language L, where P is a set of known identifiers of primitive methods used in a method that is being learnt:

- for any $p \in P$, let $p \in L$,
- for any $l_1, l_2 \in L$, let $[l_1, l_2] \in L$,
- for any $l_1, l_2 \in L$, let $[l_1|l_2] \in L$,
- for any $l \in L$, let $l^* \in L$,
- for any $l \in L$ and $n \in \mathbb{N}$, let $l^n \in L$,
- for $list = (l_1, \ldots, l_k)$ such that $l_i \in L$ and $1 < i \leq k$, let $T(list) \in L$.

"[" and "]" are auxiliary symbols used to separate subexpressions, "," denotes a *sequence*, "|" denotes a *disjunction*, "$*$" denotes a *repetition* of a subexpression any number of times (including 0), n a fixed number of times, and T is a constructor for a branching point ($list$ is a list of branches), i.e., for proofs which are not sequences but branch into a tree.[3] Let the set of primitives P be {assoc-l, assoc-r, inv-l, inv-r, id-l, id-r}. Using this language, the tactic of our simplification method described by the three group theory examples above can be expressed as:

$$\mathsf{simplify} \equiv \big[\mathsf{assoc\text{-}l}^{\,*}, [\mathsf{inv\text{-}r}|\mathsf{inv\text{-}l}\,], \mathsf{id\text{-}l}\,\big].$$

We refer to expressions in language L which describe compound methods as *method outlines*. simplify is a typical method outline that we aim our system to learn automatically.

3.2 Machine learning algorithm

Method outlines are abstract methods which have a simple representation that is amenable to learning. We now present an algorithm which can learn method outlines from a set of well-chosen examples. The algorithm is based on least general generalisation [23, 24], and on the generalisation of the simultaneous compression of well-chosen examples.

As with compression algorithms in general, we have to compromise the expressive power of the language used for compression with the time and space efficiency of the compression process. Optimal compression – in the sense of Kolmogorov complexity – can be achieved by using a Turing-complete programming language. However, optimal compression is not computable in general, that is, there is no algorithm which finds the shortest program to represent any particular string. As a compromise we selected regular expressions with explicit exponents and branching points, which seem to offer a framework that is on the one hand, general enough for our purpose, and on the other

[3] Note the difference between the disjunction and the tree constructors: for disjunction the proofs covered by the method outline consist of applying either the left or the right disjunct – this is commonly known as the OR branch. However, with the tree constructor every proof branches at that particular node to all the branches in the list – this is commonly known as the AND branch.

Note also, that there is no need for an empty primitive as it can be encoded with the use of existing language. E.g., let ϵ be an empty primitive and we want to express $[a, b, [\epsilon|c], d]$. Then an equivalent representation without the empty primitive is $[a, [b|[b, c]], d]$. We avoid using the empty primitive as it introduces a large number of unwanted generalisation possibilities.

hand, (augmented with appropriate heuristics) sufficiently efficient.[4] There are some disadvantages to our technique, mostly related to the run time speed of the algorithm relative to the length of the examples considered for learning. The algorithm can deal with relatively small examples such as those we presented without the use of any heuristic.

Our learning technique considers some number of positive examples[5] which are represented in terms of lists of identifiers for primitive methods, and generalises them so that the learnt pattern is in language L. The pattern is of smallest size with respect to this defined measure of size, which essentially counts the number of primitives in an expression (where $l_1, l_2, p \in L$, $p \in P$, $n \in \mathbb{N}$ for some finite n, len is length of a list function, and $list$ is a list of expressions from L).

$$\begin{aligned} size([l_1, l_2]) &= size(l_1) + size(l_2) \\ size([l_1|l_2]) &= size(l_1) + size(l_2) \\ size(T(list)) &= \textstyle\sum_{i=1}^{len(list)} size(l_i) \text{ where } l_i \in list \\ size(l_1^n) &= size(l_1) \\ size(p) &= 1 \\ size(l_1^*) &= size(l_1) \end{aligned}$$

This is a heuristic measure of size and the intuition for it is that a good generalisation is one that reduces the sequences of method identifiers to the smallest number of primitives (e.g., $[a^2]$ is better than $[a, a]$).

The pattern is also most specific (or equivalently, least general) with respect to the definition of specificity *spec* which is measured in terms of the number of nestings for each part of the generalisation.

$$\begin{aligned} spec([l_1, l_2]) &= 1 + spec(l_1) + spec(l_2) \\ spec([l_1|l_2]) &= 1 + spec(l_1) + spec(l_2) \\ spec(T(list)) &= 1 + \textstyle\sum_{i=1}^{len(list)} spec(l_i) \text{ where } l_i \in list \\ spec(l_1^n) &= 1 + spec(l_1) \\ spec(p) &= 0 \\ spec(l_1^*) &= 1 + spec(l_1) \end{aligned}$$

Again, this is a heuristic measure. The intuition for this measure is that we give nested generalisations a priority since they are more specific and hence less likely to overgeneralise.

In our experiments, we take both, the size first (choose smallest size), and the specificity second (choose highest specificity) into account when choosing the generalisation. If the generalisations considered have the same rating according to the

[4] Our chosen language L (see Section 3.1) cannot express all method outlines. For example, we cannot express an outline that a method m_1 (e.g., definition unfolding) should be applied as often as possible, then a different method m_2 should be applied, and finally a third method m_3 (e.g., definition folding) should be applied exactly as often as the first method m_1. In our language we would have to overgeneralise this to $[m_1^*, m_2, m_3^*]$ (unless we know the number of method applications explicitly and this stays the same in all example proofs).

[5] We use positive examples only, because these are readily available. Negative examples are useful only if they are "near-misses" of proof attempts which uncover important features of the proof. Constructing and choosing informative negative examples is non-trivial, requires a lot of analysis and reasoning, and detracts from the main goal of our research. However, it would be an interesting topic for future research.

two measures, then we return all of them. For example, consider two possible generalisations: $[[a^2]^*]$ and $[a^*]$. According to size, $size([[a^2]^*]) = 1$ and $size([a^*]) = 1$. However, according to specificity, $spec([[a^2]^*]) = 2$ and $spec([a^*]) = 1$. Hence, the algorithm chooses $[[a^2]^*]$.

Note that there are other ways of selecting a generalisation and finding a different compromise between size (keeping learnt expressions small) and specificity (keeping learnt expressions close to the examples). For instance, one could vary the value of the following formula $\alpha \cdot size(l_i) + (1 - \alpha) \cdot spec(l_i)$ by changing the value of α in order to select a suitable generalisation l_i. The value of α could depend on the degree to which the generalisation should be concise and general/specific (e.g., sometimes it may be beneficial to overgeneralise). Moreover, there are other possible heuristic measures to select a generalisation. We defined and chose size and specificity that are suitable measures in our problem domains and with our set of theorems. In the range between specificity and generality, we tend to (slightly) overgeneralise, but the test results in Section 4 demonstrate that our choice is a suitable one.

Here is the learning algorithm. Given some number of examples e_i (e.g., $e_1 = [a, a, a, a, b, c]$ and $e_2 = [a, a, a, b, c]$):

1. For every example e_i, split it into sublists of all possible lengths plus the rest of the list. We get a list of pattern lists pl_i, each of which contains patterns p_i.[6] E.g.:
   - for e_1: $\{[[a], [a], [a], [a], [b], [c]]$, $[[a, a], [a, a], [b, c]]$, $[[a], [a, a],\ [a, b],\ [c]]$, $[[a, a, a], [a, b, c]]$, $[[a], [a, a, a],\ [b, c]], \ldots\}$
   - for e_2: $\{[[a], [a], [a], [b], [c]]$, $[[a, a], [a, b], [c]]$, $[[a],\ [a, a],\ [b, c]]$, $[[a, a, a],\ [b, c]]$, $[[a], [a, a, b], [c]], \ldots\}$
2. If there is any branching in the examples, then recursively repeat this algorithm on every element of the list of branches.
3. For every example e_i and for every pattern list pl_i find sequential repetitions of the same patterns p_i in the same example. Using an exponent denoting the number of repetitions, compress them into p_i^c and hence pl_i^c. E.g.:
   - $pl_1^c = \{[[a]^4, [b], [c]], [[[a, a]^2], [b, c]], \ldots\}$
   - $pl_2^c = \{[[a]^3, [b], [c]], [[a], [a, a], [b, c]], \ldots\}$
4. For every compressed pattern $p_i^c \in pl_i^c$ of every example e_i, compare it with p_j^c in all other examples e_j, and find matching m_k with the same constituent pattern, which may occur a different number of times. E.g.:
   - $m_1 = (pl_1^{c1}, pl_2^{c1})$, due to $[a]^4$ and $[a]^3$
   - $m_2 = (pl_1^{c2}, pl_2^{c2})$, due to $[b, c]$ and $[b, c]$, etc.
5. If there are no matches m_k in the previous step, then generalise the examples by joining them disjunctively using the "|" constructor.
6. For every p_i^c in a matching, generalise different exponents to a "$*$" constructor, and the same exponents n to a constant n, and hence obtain p_g. E.g.:[7]
   - for m_1: $[a]^4$ and $[a]^3$ are generalised to $p_g = [a]^*$
   - for m_2: $[b, c]$ and $[b, c]$ are generalised to $p_g = [b, c]$

[6] Notice that there are n mod m ways of splitting an example of length n into different sublists of length m. Namely, the sublists of length m can start in positions $1, 2, \ldots, n$ mod m.

[7] Notice that here is a point where our generalisation technique can overgeneralise. For instance, when there is a pattern in the exponents, e.g., all exponents are prime numbers, then this is ignored and just a $*$ is selected.

7. For every p_g of a match, transform the rest of the pattern list on the left and on the right of p_g back to the example list, and recursively repeat the algorithm on them. E.g.:
   - for m_1 in e_1: LHS= [ ], $p_g = [a]^*$, repeat on RHS=$[b, c]$
   - for m_1 in e_2: LHS= [ ], $p_g = [a]^*$, repeat on RHS=$[b, c]$
   - for m_2 in e_1: repeat on LHS= $[a, a, a, a]$, $p_g = [b, c]$, RHS= [ ]
   - for m_2 in e_2: repeat on LHS= $[a, a, a]$, $p_g = [b, c]$, RHS= [ ].
8. If there is more than one generalisation remaining at the end of the recursive steps, then pick the ones with the smallest size and among these the ones with the largest specificity. E.g.: after the algorithm is repeated on the rest of our examples, the learnt method outline will be $[[a]^*, [b, c]]$.

The learning algorithm is implemented in SML. Its inputs are the sequences of method identifiers from proofs that were constructed in ΩMEGA. Its output are method outlines which are passed back to ΩMEGA. The algorithm was tested on several examples of proofs and it successfully produced the required method outlines.

3.2.1 Learnt method outlines for the examples

For the examples introduced in Section 2 our learning algorithm generates the following method outlines:

- Group theory:

$$\mathsf{simplify} \equiv \big[\mathsf{assoc\text{-}l}^*, [\mathsf{inv\text{-}r}|\mathsf{inv\text{-}l}], \mathsf{id\text{-}l}\big].$$

- Residue classes:

$$\begin{aligned}\mathsf{tryanderror} \equiv\ & \big[\mathsf{defn\text{-}exp}, [\forall_i\text{-}\mathsf{sort}]^*, \mathsf{convert\text{-}resclass\text{-}to\text{-}num},\\ & [[\mathsf{or\text{-}e\text{-}rec}]|[\mathsf{defn\text{-}exp}, \mathsf{or\text{-}e\text{-}rec}]], \mathsf{simp\text{-}num\text{-}exp}^*, \mathsf{reflex}^*\big]\end{aligned}$$

- Set theory:

$$\mathsf{learnt\text{-}set} \equiv \big[[\forall_\mathsf{i}]^3, \mathsf{set\text{-}ext}, \forall_\mathsf{i}, \mathsf{defni}^*, [\mathsf{atp\text{-}otter}|\mathsf{counterex\text{-}satchmo}]\big]$$

As mentioned before, the method outline simplify for the group theory examples is not the most general one, as the examples that it was learnt from did not contain the use of the right identity method, for example. Furthermore, it is only a single application of simplification. However, we tested our learning algorithm also on examples that use this single simplify method several times. As expected, the learning mechanism learnt a method outline which is a repeated application of simplify, namely rep-simplify = simplify*. We also tested the learning mechanism on examples that use methods such as right identity and right associativity, and altogether learnt five new method outlines, some of which are repeated applications of others.

In the domain of residue classes, the learning mechanism also learnt another method outline called choose. When fully fleshed into an ΩMEGA method (see Section 3.3), this method proves a subpart of proofs for theorems of residue classes. Namely, given a theorem with an existential quantifier, statements on integers are combined using a disjunction in a particular normal form (from the right side). Then, each disjunct has to be checked until one is found that is true for the statement. Hence, the method

choose starts inspecting the right disjuncts until either the right (ori-r) or the left (ori-l) one is true, which is then followed with the rest of the proof, in this case with the application of reflexivity (reflex). This proof pattern is learnt and captured in the method outline:

$$\mathsf{choose} = \left[\mathsf{defn\text{-}exp},\ \mathsf{ori\text{-}r}^*, [\mathsf{reflex} \mid [\mathsf{ori\text{-}l},\ \mathsf{reflex}]]\right]$$

The method corresponding to the third method outline *learnt-set*, i.e., for set theory examples, transforms a higher-order problem into a propositional logic one, which is much easier to prove or disprove, since it is a decidable problem. The method does not eliminate search altogether, but makes it, in this case, much more tractable. Notice also, that the method outline learnt-set applies the elimination of the universal quantifier ($\forall_{\mathsf{i}}$) only three times. This is consistent with the examples from which the method outline was learnt, but in general the quantifier elimination would be applied any required number of times, which could be denoted with a star construct in the method outline. This shows that the quality of a method outline learnt from the examples depends on the quality of the input examples. Hence, it is important to use well-chosen examples when learning new methods. Note, however, that sometimes a slight over-generalisation might be beneficial. Also note that any learning can work only if in the domain there is some structure or regularity which can be exploited.

3.2.2 Properties

Let us look at some properties of our learning algorithm:

Property 3.1 Given a number of examples, the algorithm learns a generalisation which is at least as general as all examples.

In order to see this property let a language expression r stand for the set of all expressions that are just sequences of primitive expressions. Then an expression r_1 is more general than another r_2 if each primitive expression of the set of sequences for r_2 is a proper subset of that for r_1. In the algorithm only the steps (5) and (6) are critical since all others do not change the generality of the expressions. Only steps (5) and (6) perform a generalisation, (5) in form of a disjunction, (6) in form of a star. Since a disjunction covers each of its disjuncts, and a star each of its components as well, the property follows.

Property 3.2 The learning algorithm is exponential.

In terms of computational complexity, the algorithm is quadratic in step (1)[8] and is exponential in step (7), since we try every possible combination here. All other steps are linear. The complexity of step (7) could be improved by using the initially computed information about all sublists of an example list, rather than recomputing it in every recursive step.

Since step (7) is exponential, our learning algorithm does not run efficiently for large examples.[9] In case the algorithm needs to be used for very large examples,

[8] An example list of length n is split into all together n^2 different sublists: there are n sublists of length 1, $n-1$ sublists of length 2, $n-2$ of length 3, $n-k+1$ of length k and so on, and 1 sublist of length n. Hence, in total, there are n^2 sublists of different fixed lengths. Notice that there exist algorithms, e.g., suffix trees, which run this step in linear time.

[9] However, we argue that proof methods that are being learnt typically do not consist of a large number of low-level methods. Indeed our algorithm runs efficiently on all the tested examples.

we implemented some heuristic optimisations. These prune the number of generated matches. Good heuristics are those which select matches that make a big impact on the size of the final generalisations. For example, a good heuristic is to pick a pattern match whose pattern of smallest size forms a maximal sublist of the original example. This enables the algorithm to deal with very large examples (e.g., lists of length 2000) which are way beyond the length of examples that we expect for learning our method outlines. Clearly, using such heuristic learning may miss the best generalisation (according to the measures defined above). The user of our LEARNΩMATIC system can choose whether to use the heuristic optimisations in the learning mechanism or not. Users could also define their own heuristics, but this is left for future work.

3.3 Using learnt methods

Method outlines that have been learnt so far do not contain all the information which is needed for the proof planner to use them. For instance, they do not specify what the pre- and postconditions of methods are, they also do not specify how the number of loop applications of methods is instantiated when used to prove a theorem. In our approach, we restore the missing information by search.

In the particular case of our implementation in the ΩMEGA proof planning system, important information which is needed for the application of methods – but which is lost in the abstraction process – are parameters for the methods that constitute the newly learnt method. Concretely, the methods which make up the new learnt method in ΩMEGA take some (or no) parameters. These can be in the form of position information indicating where in the expression the method is applied, or a term naming the concept for which the definition should be expanded, or instantiating a term used by the method, etc. The parameters of a method are supplied by control-rules to reduce and to direct the search performed by the proof planner. For example, the parameter in the definition expansion method *defn-exp* names the concept that should be expanded. The possible relevant control-rules can be of the form 'Expand only definitions of the current theory' or 'Prefer definition expansion of the head symbol of the formula to be proved'.

A set of methods together with a set of control-rules defines a planning strategy of ΩMEGA's multi-strategy proof planner MULTI [19]. Note that control-rules of a strategy are used not only for determining the parameters of methods, but also to prefer or reject methods according to the current proof situation.

For each learnt method outline we automatically build a method. The precondition of a learnt method employs search that is guided and structured by the method outline; that is, we perform search guided by the method outlines in order to analyse whether the learnt method is applicable.

The postcondition introduces the new open goals and hypotheses resulting from applying the methods of the sequence to the current goal. We will call this kind of method a *learnt method*.

The precondition of a learnt method cannot be extracted from the pre- and post-conditions of the *uninstantiated* methods in the method outline, because the formulae introduced by the postcondition depend on the formulae that fulfil the preconditions. We actually have to apply a method to produce a proof situation for which we can

test the preconditions of the subsequent method in the method outline. That is, we have to perform proof planning guided by the learnt pattern which is captured by the method outline.[10]

In detail, the applicability test is realised by the following algorithm:

1. Copy the current proof situation. Initialise a stack with a pair $(P_0; \emptyset)$, where P_0 is the initial learnt method outline and $\emptyset$ stands for the empty history.
2. Take the first pair from the stack:
   (a) If this pair is $([[P_1|P_2], P']; \mathcal{H})$, then put $([P_1, P']; \mathcal{H})$ and $([P_2, P']; \mathcal{H})$ back on the stack. For $([P^n, P']; \mathcal{H})$ put $([P, P^{n-1}, P']; \mathcal{H})$ on the stack. In the case of $([P^*, P']; \mathcal{H})$, return $(P'; \mathcal{H})$ and $([[P, P^*], P']; \mathcal{H})$.[11]
   (b) If the pair is $([m, P']; \mathcal{H})$ where m is a method-name, then test the precondition of m for all open goals (and for all possible instantiations of method parameters, if the method contains parameters). Each satisfied test of preconditions results in a partial matching μ_i of m for the corresponding goal (and parameter). The partial matchings $([\mu_1, P'], \mathcal{H}), \ldots, ([\mu_n, P'], \mathcal{H})$ are put on the stack. If m is not applicable, then backtrack the difference between the current history $\mathcal{H}$ and the history of the next pair of the stack.
   (c) If the pair is $([\mu_i, P']; \mathcal{H})$ where μ_i is a partially instantiated method, then apply the postconditions of μ_i to the copied proof and put $(P'; (\mu_i, \mathcal{H}))$ on the stack.
   (d) If the pair is $([\,]; \mathcal{H})$ where $[\,]$ denotes the empty outline, an instantiation of the learnt outline is found. That means, a particular sequence of methods, corresponding to the method outline, has been successfully applied and be found in $\mathcal{H}$.
3. If the stack is empty, then it was not possible to apply the learnt method outline; otherwise continue with step (2).

Notice that the application of the method introduces new open lines and new hypotheses resulting from the application of methods in $\mathcal{H}$ into the proof.

The learnt method may contain other learnt methods. That is, the applicability test in (2)(b) may recursively call this same algorithm again within the applicability test of an embedded learnt method.

Our implementation of the applicability test causes an overhead in the run time behaviour of the system. This is because the current proof is firstly copied in step (1) of the applicability test of the learnt method, and secondly in case of an application of the method the new open goals and hypotheses are copied back into the original proof. These two copying steps are carried out in order to avoid an interference between the planning process of MULTI in the current proof situation, and the planning process inside the applicability test of the method outline. The inefficiency due to overhead could be avoided in a complete re-implementation of the MULTI proof planner.

---

[10] One may suggest that our system learns tactics rather than methods as we have not mechanised the learning of preconditions. Such a suggestion is not entirely correct, since we can use the learnt methods in proof planning. It is true, however, that because of the increasing complexity of methods the originally clear difference between tactics and methods is getting increasingly blurred – not only in our approach but in proof planning in general.

[11] There is a counter for the operator $*$, the evaluation of this operator is only performed until an upper bound is reached. This guarantees the termination of the applicability test.

4 Evaluation and Experiments

In order to evaluate our approach, we carried out an empirical study in different problem domains. In particular, we tested our framework on examples of group theory, residue classes and set theory. The aim of these experiments was to investigate if the proof planner ΩMEGA enhanced with the learnt methods can perform better than the standard ΩMEGA planner. The learnt methods were added to the search space in conjunction with a heuristic (control-rule) specifying that their applicability is checked first, that is, before the existing standard methods.

The measures that we consider are:

1. *matchings* – the number of all true and false attempts to match methods that are candidates for application in the proof plan;
2. *proof length* – the number of steps in the proof plan;
3. *timing* – the time it takes to prove a theorem;
4. *coverage* – the ability to prove theorems.

In order to perform these tests we have built different counters in the program. The counter *matchings* counts the successful and unsuccessful application tests of methods. It also contains the method matchings checked by the search engine in the applicability tests of learnt methods (see Section 3.3). *Matchings* provides an important measure, since on the one hand, it indicates how directed the search for a proof is. On the other hand, checking the candidate methods that may be applied in the proof is by far the most expensive part of the proof search. Hence, *matchings* is a good measure to compare the performance of the two approaches (i.e., with and without learnt methods) while it is also independent of potential implementation inefficiencies.

The development set usually consists of a small number of examples, in particular, for the examples in the domains discussed in this paper it consisted of three example theorems (see Section 2). The test set consists of a number of theorems, which are new, more complex, and significantly diverse from the development set. It excludes the proofs from which the new methods were learnt.

The size of our test sample was relatively small in group theory: we tested our learnt methods on 8 theorems, but large in other domains: we had 881 theorems of residue classes and 120 conjectures of set theory.

Moreover, we chose our test set to be characteristic of the problem domain in general. Furthermore, notice that some evaluation measures, e.g., *proof length* and *coverage* are independent of the size of the test set. Namely, some inspection of the approach clearly indicates that the proof plans that use learnt methods will be shorter, and from the domain of group theory, it is clear that new theorems are proved that otherwise could not be.

Table 1 compares the values of *matchings* and *proof length* for the three problem domains. In each problem domain we break down the results according to the type of theorems under consideration (e.g., how complex they are, what pattern of reasoning or proof methods their proofs may use, how many variables are in them). The table compares the values for these measures when the planner searches for the proof with the standard set of available methods (column marked with S), and when in addition to these, there are also our newly learnt methods available to the planner (column marked with L). "—" means that the planner ran out of resources (i.e., four hours of

Domain	Type of Theorems	Matchings S	Matchings L	Matchings $\frac{S}{L}$	Length S	Length L	Length $\frac{S}{L}$
Group theory	simple	94.2	79.0	1.19	15.5	8.3	1.87
	complex	—	189.6	—	—	9.8	—
Residue Class $\mathbb{Z}_3$	choose	691.0	656.0	1.05	39.3	33.0	1.19
	tryanderror	425.3	82.1	5.80	38.6	2.0	19.30
	both	552.9	323.2	1.71	39.7	19.0	2.09
Residue Class $\mathbb{Z}_6$	choose	751.2	713.8	1.05	35.2	29.1	1.21
	tryanderror	2309.5	402.9	5.73	218.2	2.0	109.10
	both	2807.8	1419.3	1.98	185.9	73.0	2.55
Residue Class $\mathbb{Z}_9$	choose	1996.1	1640.5	1.22	111.2	78.4	1.42
	tryanderror	4769.1	1132.2	4.21	453.1	2.0	226.55
	both	6931.6	3643.4	1.90	438.7	163.0	2.69
Set theory	1 variable	26.9	42.0	0.64	6.6	6.6	1.00
	3 variables	45.6	14.9	3.06	10.9	2.0	5.45
	5 variables	48.1	28.7	1.68	12.7	4.0	3.17

TABLE 1. Evaluation results.

CPU time) and could not find a proof plan.

4.1 Group theory domain

In the group theory domain, our learning mechanism learnt five new methods, but since some are repeated applications of others, we only tested the planner by using two newly learnt complex compound methods.[12]

The methods simplify group theory expressions by applying associativity left and right methods, and then reduce the expressions by applying appropriate inverse and identity methods (see Section 2.1). The entries in Table 1 refer to two types of examples. First, we give the average figures for simple theorems that can be proved with standard *and* with learnt methods. Second, we give the average figures for complex theorems that can be proved *only* when the planner has our learnt methods.

It is evident from Table 1 that the number of *matchings* is improved, but it is only reduced by about 15%. We noticed that the simpler the theorem, the smaller the improvement. In fact, for some very simple theorems, a larger number of *matchings* is required if the learnt methods are available in the search space. The reason for this behaviour is that there are only a few standard methods available initially in the group theory domain. Hence, any additional learnt method will noticeably increase the search space. Also, the application test for learnt methods may be expensive, especially when a learnt method is not applicable, but still all possible interpretations of the learnt method outline have to be checked by the search engine. However, for more complex examples, this is no longer the case, and an improvement is noticed. This is because the search within the applicability test of the learnt method is more directed compared to the search performed by the proof planner. The improvement increases

[12] In general, it is a good heuristic to keep the size of the set of applicable methods small. This can be achieved by subsuming specialised methods by more general ones. For example, as soon as the system has learnt repeated application of simplify in group theory (rep-simplify = simplify*), we can remove the proof method simplify.

when a larger number of primitive methods is replaced by the learnt methods.

As expected, the *proof length* is reduced by using learnt methods.

On average, the *time* it took to prove *simple* theorems of group theory was approximately 100% longer than without the learnt methods. Notice that this does not include the case of *complex* theorems, when the proof planner timed out without finding the proof plans of the given theorems. The reason for bad timing in the case of *simple* theorems is that the learnt methods are small and simple, and the proof search contains the overhead due to the current implementation for the reuse of the learnt methods (see Section 3.3).

On the other hand, in the case of *complex* group theory examples, the advantage of having learnt methods in the search space is evident from the fact, that when our learnt methods are not available to the planner, then it cannot prove some complex theorems. When trying to apply methods such as associativity left or right, for which the planner has no control knowledge about their application, then it cannot find a proof plan within the given resources (i.e., four hours of CPU time). Our learnt methods, however, encapsulate typical patterns of reasoning about these theorems, hence they provide control over the way the methods are applied in the proof and lead to successful proof plans.

4.2 Residue class domain

In the domain of residue classes, we gave our learning mechanism examples from the residue class $\mathbb{Z}_3$ domain such that it learnt two new methods: tryanderror (as demonstrated in our examples in Section 3.2), and choose.

We applied the standard set of methods and the set enhanced with the two learnt methods to randomly chosen theorems regarding the residue class sets $\mathbb{Z}_3$, $\mathbb{Z}_6$ and $\mathbb{Z}_9$. We subdivided the results in Table 1 according to whether only one of the learnt methods or both of them were applicable in the proof. The number of method matchings is also represented in FIGURE 2 and the length of proofs in FIGURE 3. The labels in these figures denote the class of theorems, for example, "choose L" stands for theorems where the learnt method choose was applicable and proved by a strategy containing the learnt methods, while "choose S" stands for the same class of theorems but now proved with the standard strategy (i.e., without the learnt methods).

There is an improvement in each residue class set when learnt methods are available. Since choose replaces only small subproofs, whereas tryanderror can prove the whole theorem in one step, the latter has clearly better results for *proof length* and *matchings*. The benefit in the search for proofs where both learnt methods are applicable lies between them.

In addition to comparing the absolute values for our measures within the different sub-domains of residue class theorems (i.e., $\mathbb{Z}_3$, $\mathbb{Z}_6$ and $\mathbb{Z}_9$) in Table 1, we also compare the relative improvement between the different sub-domains. This can be done by examining the ratio between the number of *matchings* in the standard (S) and the enhanced (L) sets of methods (and the same for *proof length*), and then comparing the ratios for each type of theorems across sub-domains. Table 1 states these values. For example, the ratio for *proof length* in the case of theorems that use tryanderror method in $\mathbb{Z}_3$ is 19.30. This means that the proofs when only standard methods are available are 19.30 times longer than when learnt methods are available as well.

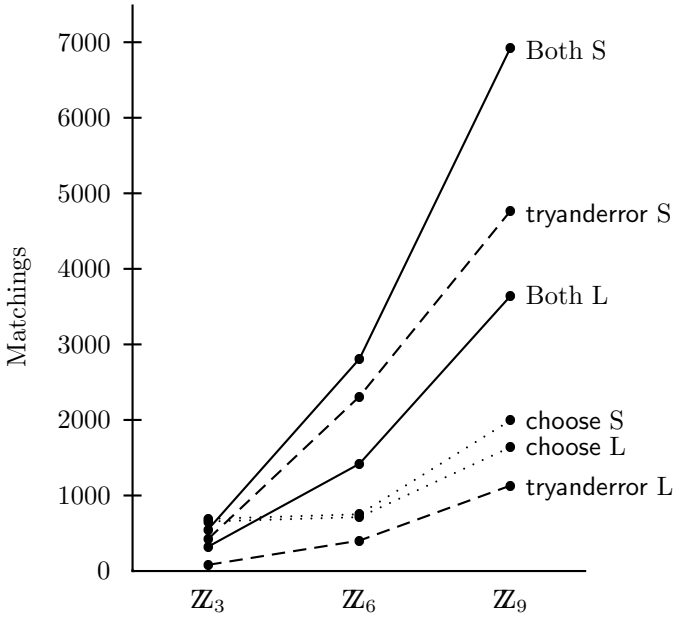


FIG. 2. Method matchings for residue classes.

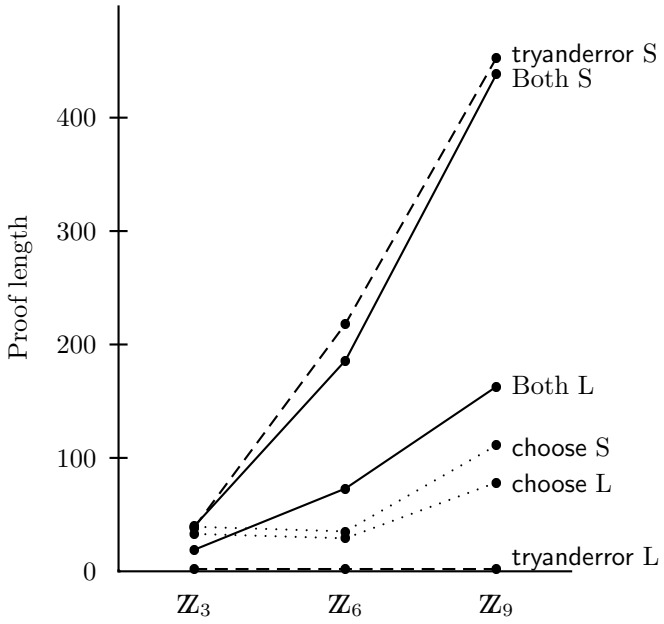


FIG. 3. Proof length for residue classes.

Table 1 clearly shows that the ratios for *proof length* increase across sub-domains (e.g., in case both learnt methods are used, the ratio increases from 2.09 to 2.55 and 2.69 across sub-domains). This indicates that the more complex the theorem (higher residue classes have longer and more complex proofs), the better the improvement when learnt methods are available to the planner.

In general, the same trend can be observed for the *matchings* ratios. An exception are the ratios for the type of theorems that can be proved using tryanderror method, which only marginally decrease across sub-domains (but we would expect them to increase as in the case for theorems that are proved using choose method). This can be explained by the fact that the theorems were randomly chosen across sub-domains, rather than using the theorems for the same properties but different residue classes. Namely, the random differences in the complexity of theorems in different sub-domains may be significant, e.g., the properties randomly chosen in $\mathbb{Z}_6$ may be more complex to prove than the ones chosen in $\mathbb{Z}_3$.

On average, the *time* it took to prove theorems of residue classes with the newly learnt methods was 50% shorter for proofs containing tryanderror than without such methods, 25% longer for both methods and 80% longer for choose. The time corresponds to the measured *matchings* but suffers from the overhead of the current implementation, especially for the smaller choose method. Since the learnt methods are tried before the standard set of methods, this effect increases for longer proofs.

4.3 Set theory domain

The examples in this domain were selected to test the cost of learnt methods in the search process when they are not applicable. To this end, we added to the set of available methods the method learnt-set (see Section 3.2) that was learnt from theorems containing *three* variables. Note that since all the theorems in the development set have three variables, the universal quantification in learnt-set is eliminated (introduced in backward reasoning) exactly three times. Our development set is chosen deliberately in this restricted way in order to test the effect of a learnt method in situations where it is not applicable or applicable only in combination with other methods. In this way, we wanted to find out to which degree can learnt methods have a negative effect on the search space. Note that if we chose 'better examples' for learning, e.g., theorems that have one, three *and* five variables, then our learnt method learnt-set would be more powerful and applicable to all three types of theorems.

In order to test the restricted method learnt-set we added to our test set two types of theorems, namely, with one and with five variables. As expected, learnt-set is not applicable in the proofs of theorems with one variable. In the proofs of theorems with five variables learnt-set is applicable after two methods of the standard set are applied.

For theorems with three variables the proof search performs best, that is, the number of *matchings* is reduced by a factor of three when the learnt method is available. *proof length* is reduced by more than five times. The results for theorems with five variables are still better than without the learnt method, but as expected, not as good as with three variables. For theorems with one variable, where learnt-set is not applicable at all, the proof search clearly suffers from the additionally available learnt method, and hence the number of *matchings* is increased. Of course, *proof length* is not affected in this case.

The benefits and drawbacks of the availability of learnt methods can be seen very clearly in these evaluation results for the set theory examples. Namely, when a learnt method is applicable, then its availability improves the performance of the proof planner. However, when a learnt method is not applicable then the proof planner has

to test a larger set of methods, and this will harm its performance.

On average, the *time* it took to prove or disprove conjectures in set theory with the newly learnt methods was about 40% faster for theorems with three variables, approximately 5% faster for theorems with five variables, and nearly 20% slower for theorems with one variable.

4.4 Analysis of results

As it is evident from the discussion above, in general, the availability of newly learnt methods that capture general patterns of reasoning improves the performance of the proof planner. In particular, the number of *matchings* (which are the most expensive part of the proof search) is reduced across domains, as indicated in Table 1. Furthermore, as expected, learnt methods cause proofs to be shorter, since they encapsulate a number of other methods. Also, the *time* is in general reduced when using learnt methods. There are some overheads, and in some cases these are bigger than the improvements. Since the *time* should be related to the reduced number of *matchings*, but it is not in all our cases (group theory), this indicates that our implementation of the execution of learnt methods, as described in Section 3.3, is not as efficient as that of the ΩMEGA proof planner.

In our experiments, the *coverage* when using learnt methods is increased, which is also indicated by the fact that using learnt methods, ΩMEGA can prove theorems that it cannot prove otherwise. Since in our experiments proof plans were either found relatively quickly or not at all, we did not notice a possible effect where some proof plans that were found with the standard set of methods, now could no longer be found, because the learnt methods misled the proof search and increased planning time beyond the four hour limit.

The reason for the improvements described above is due to the fact that our learnt methods provide a structure according to which the existing methods can be applied, and hence they direct search. This structure also gives better explanations why certain methods are best applied in particular combinations. For example, the simplification method for group theory examples indicates how the methods of associativity, inverse and identity should be combined together, rather than applied blindly in any possible combination.

A general performance problem of using learnt methods arises when a learnt method is not applicable. A learnt method is not applicable when there is no instantiation of the learnt sequence so that the methods of this instantiation are applicable. This means that every possible instantiation has to be tested and refuted. In the presented experiments, the learnt methods nearly always outperform the standard set of primitive methods. But there could be worst case scenarios where the learnt method is very general (contains many star operations) and a large part of the learnt sequence is applicable but the whole sequence is not. This has not happened in our experiments.

4.5 Analysis of general approach

The additional hierarchical structure of proofs constructed with learnt methods can also be beneficial for proof verbalisation and proof explanation tools like P.REX [8]. The information hidden within our learnt methods can now likewise be hidden in

verbalisations, and expanded if appropriate or requested by the user. Namely, learnt methods encapsulate bigger and more abstract steps in proofs than smaller methods that make up our learnt methods. Hence, learnt methods provide a higher-level explanation of what is going on in the proof plan, and therefore they help to reflect the main idea of a proof by masking and grouping details in the proof. In combination, for instance, with the proof verbalisation tool P.REX it enables the proof planner ΩMEGA to automatically produce better explanations of the proofs which can be as high-level or as low-level as needed.

The preconditions of learnt methods are currently generated by the search engine for the reuse of methods described in Section 3.3. The engine searches for the instantiation of the method outline which is applicable in a given proof situation. This means that a small amount of search, which is guided by the method outline, needs to be carried out in the applicability test of the learnt method. Note that in the standard set of methods, i.e., not the learnt ones, the applicability test is carried out by checking if the *explicitly* and *declaratively* stated preconditions for the method hold or not in a given proof situation – similarly to the case with our learnt methods, this may also require search. The fact that the preconditions of the standard set of methods are declaratively stated, but the preconditions of our learnt methods need to be computed, does not change how proof methods are treated in the planning process. All methods, whether learnt or not, form part of the search space that the proof planner traverses in the process of finding a proof plan. Indeed, one of our motivations stated at the start of this paper was to devise a mechanism which is able to learn new primitives of the search space, rather than control the search within a fixed set of primitives. In the framework of proof planning the primitives of the search space are proof methods which we can now learn automatically. Even when some search needs to be carried out in order to compute the applicability condition of our learnt methods, this still is much better, that is, search is much pruned, than when such methods are not available to the planner. This is supported by the results of our evaluation demonstrated above in this section.

It is obvious that a new learnt method does not, in general, make some of the lower level methods obsolete while sustaining some notion of completeness. On the other hand, we cannot rule out this possibility completely for special cases. Determining such situations is challenging and requires proof theoretic methods based on derivability and admissibility criteria. This task can clearly not be addressed automatically in our approach.

We can see our approach as a mechanism that learns how to hierarchically structure the search through the search space. We built new methods that encapsulate guided search over some more primitive methods, and then add these new elements as a kind of *chunks of structured search* to our system. This contrasts with the idea of having only one global control layer in proof planning, since our learnt methods themselves can be seen as little planning processes consisting of a set of internal methods and control information on how to search with them.

The mechanism for reusing learnt methods described in Section 3.3 is specific to ΩMEGA proof methods. On the other hand, the learning algorithm presented in Section 3.2 is general and can be used for learning in other automated reasoning systems, not just the ΩMEGA proof planner (see Section 6). The learning algorithm learns method outlines which with some enrichment could be used in other systems as infer-

ence rules, in ΩMEGA as proof methods, in $\lambda Clam$ [27] as methodical expressions [26], etc. In fact, in some systems, like $\lambda Clam$, method outlines are exactly methodical expressions that the planner can use directly, so no enriching of the method outline representation is required. In other systems, enriched method outlines are just inference rules. This may give rise to the question of what is the difference between methods, methodical expressions and tactics. It seems that our method outlines offer a unified view of all these structures that are used in different automated reasoning system, e.g., inferencing systems, tactical theorems provers, and proof planners. Depending on the system, a different primitive of the search space is needed (e.g., inference rules, tactics, proof methods, methodical expressions). Hence, the enriching of the learnt method outline representation so that the new primitive can be used in the given system has to be carried out differently, or may indeed need no enriching at all. Studying how this process varies for different systems may give us some clues about the similarities and differences between such structures, but this is left for future work.

5 Related Work

Some work has been done in the past on applying machine learning techniques to theorem proving. Unfortunately, not much work has concentrated on high-level learning of structures of proofs and extending the reasoning primitives within an automated theorem prover.

For example, Schulz's work in [29], which is a continuation of previous work such as by Fuchs and Fuchs [9] and Denzinger and Schulz [6], investigates learning of heuristic control knowledge in the context of machine oriented theorem proving, more precisely, equational or superposition based theorem proving. Knowledge gained from the analysis of the inference process is used to learn important search decisions, which are represented as abstract clause patterns. These are employed in heuristic evaluation functions to better guide the search when attacking new proof problems. The selection of heuristic evaluation functions for a new problem at hand is guided by meta-data. The main difference with our work is that the learnt information in Schulz's work is not becoming a reasoning primitive, such as our learnt methods. It rather guides the search amongst the existing primitives at the global search layer instead of building up new, structured chunks of encapsulated search processes.

Silver [30] and Desimone [7] used precondition analysis which learns new inference schemas by evaluating the pre- and postconditions of each inference step used in the proof. A dependency chart between these pre- and postconditions is created, and constitutes the pre- and postconditions of the newly learnt inference schema. These schemas are syntactically complete proof steps, whereas the ΩMEGA methods contain arbitrary function calls which cannot be determined by just evaluating the syntax of the inference steps.

Kolbe, Walther, Brauburger, Melis and Whittle have done related work on the use of analogy [21] and proof reuse [17, 16], that is, a sort of learning to solve new problems by using a similar existing problem. Their systems require a lot of reasoning with one example to reconstruct the features which can then be used to prove a new example. The reconstruction effort needs to be spent in every new example for which the old proof is to be reused. In contrast, we use several examples to learn a reasoning

pattern from them, and then with a simple application, without any reconstruction or additional reasoning, reuse the learnt proof method in any number of relevant theorems. Ireland [12] extends the applicability of the proof planning approach by patching failed proof plans by so-called proof critics.

A piece of related work in cognitive science is Furse's Mathematics Understander [10], MU, which stores mathematical domain and procedural knowledge in a contextual memory system, and tries to simulate how students learn mathematics from textbooks. MU builds up a uniform low-level data structure, while we build high-level hierarchical proof planning methods. Having explicit methods allows us to check proofs for their correctness, while in MU incorrect proof steps cannot be distinguished from correct ones. The hierarchical character of our methods also allows for a user-adaptive proof presentation.

In the field of machine learning there is a huge amount of relevant work and we mention only some that we deem most related to our work. In terms of a learning mechanism, more recent work on learning regular expressions, grammar inference and sequence learning by Sun and Giles [31] is related. Learning regular expressions is equivalent to learning finite state automata, which are also recognisers for regular grammars. Muggleton has done related work on grammatical inference methods [22] which automatically construct finite-state structures from trace information. His method IM1 is a general one and can describe all other existing grammatical inference methods. IM1 consists of first, generating a prefix tree from example traces, second, merging of states to get canonical acceptor states (which still describe only the example traces), and third, merging states which essentially does the generalisation of the structure. The generalisation, i.e., merging, is determined by a particular chosen heuristic measure. The existing state automata learning techniques differ depending on the heuristic that they employ for generalisation. The main difference to our work is that these techniques typically require a large number of examples in order to make a reliable generalisation, or supervision or an oracle which confirms when new examples are representative of the inferred generalisation. Furthermore, the heuristics described by Muggleton do not seem to be sufficient for generalisation in our case, as none of the states describing our proof traces would be merged. It is unclear what other heuristic could be employed to suffice the generalisation of our examples. Moreover, these techniques learn only sequences, i.e., regular expressions. However, our language is larger than regular grammars as it includes constant repetitions of expressions and expressions represented as trees.

There have been various approaches to incorporate learning in planning. In the PRODIGY system [32] a number of techniques for learning are available. The goal of the learning process is either to get control knowledge, that is, rules that describe which goal to tackle next and which method to prefer at the decision points of the planning algorithm, or learn planning operators from the change of planning states by observing an expert agent. The learning mechanism of LEARNΩMATIC differs in both aspects as its goal is to learn new operators that are learnt from other operators and could be compared to learning of macro operators of chunks [28]. Another difference is that learning from an analysis of the domain theory, in our case the set of methods, without the generation of examples appears to be difficult, since proof planning methods are complex and the postconditions are only available when a method is applied in a concrete proof situation. The abstraction from the proof to method names that is the

input for the learning mechanism of LEARNΩMATIC is rather radical compared with abstractions in other planning systems, see [15]. There, a hierarchy of abstractions can be established by analysing the predicates of the domain theory. Some ideas for abstractions in method learning that retain possibly useful information are discussed in the next section.

Related is also the work on pattern matching in DNA sequences [2], as in the GENOME project, and some ideas underlying our learning mechanism have been inspired by this work.

6 Future Work

There are several aspects of our learning framework which need to be addressed in the future. With respect to the representation formalism, we have mainly considered sequential rewriting proofs. Other styles (different directions of reasoning) should also be considered.

Furthermore, we would like to apply our learning approach to other proof planners, such as λ*Clam* [27]. Since proof methods have a different structure in different proof planners, this task would require using the same learning mechanism, but probably, instead of our applicability test, a different reuse of methods approach than in the case of ΩMEGA.

The expressiveness of our language L for method outlines (see Section 3.1) could be studied further in order to determine if it should be extended. In particular, we could look into what type of ΩMEGA methods cannot be expressed using the current language L, and what other language constructs we would need. Moreover, we could examine if our language is sufficient to express primitives of the search space in other automated reasoning systems, like methodical expressions in λ*Clam* or inference rules in other theorem provers.

Regarding the learning algorithm itself, we need to examine what are good heuristics for our generalisation and how suboptimal solutions can be improved. While the learning mechanism is not efficient, we argue that we do not need a highly complicated and efficient technique for learning patterns, as in the GENOME project, for example. If we moved to larger example sets we could use a divide and conquer heuristic. Our learning algorithm without heuristics is sufficient for small patterns (e.g., less than 50 steps). We did not encounter larger patterns in our examples and do not expect very large ones for our application domain, since we assume well-chosen examples for the learning part. Our approach reflects the view that human mathematicians learn complex structures not in one single step but compose them step by step in a hierarchical way.

An interesting aspect that could be addressed in the future is whether a system could automatically learn the information that is abstracted from the proof traces and that has to be reconstructed by search performed in the applicability test when reusing learnt methods. What could this additional information that describes learnt methods more specifically be? When we take a look at the examples in group theory, it seems obvious that the *simplification* using associativity, inverse and identity methods are meant to act on the same subformula. This information is lost during abstraction, and hence, during the applicability test of the learnt method, associativity is applied at every possible place. So, the question is, could the smallest subterm of an expression

to which the newly learnt method should be applied, i.e., the focus for the method, be learnt automatically and how? Future investigations could address such questions as well as identify additional pieces of information that describe learnt proof methods more specifically. In order to reduce search with the newly learnt methods it would also be good to learn meta-level control knowledge for them.

Another interesting, but difficult idea for future work is to characterise well-chosen examples more precisely, so that these could be selected automatically, rather than depend on the user. It would be desirable to identify automatically the subparts of proof traces in several examples of proofs that contain the same reasoning pattern. In our framework, this has to be done by the user of the system. Techniques from data mining or algorithmic learning theory could perhaps be useful to tackle this difficult problem, however, they usually require very large data sets, which in proof planning we typically do not have. An idea is to apply our approach to mechanised theorem provers (rather than proof planners), for which we have large proof corpuses (e.g., MIZAR, ISABELLE, OTTER), and then use data mining techniques in order to get good examples from them.

The extraction of method sequences from proofs is currently implemented with respect to the chronological order of method applications during proof planning. There could be other orderings, e.g., the different linearisations of the proof tree, some of them could even result in more adequate learnt method outlines. For example, in a situation where the planner has more than one different subgoal that can be closed by the same sequence of method applications $[m_1, m_2]$, it depends on the search behaviour of the proof planner whether the proofs will have a trace like $[m_1, \ldots, m_1, m_2, \ldots, m_2]$ or $[m_1, m_2, \ldots, m_1, m_2]$. The learning mechanism will produce $[m_1^*, m_2^*]$ in the first case and $[m_1, m_2]^*$ in the second case. The latter will have a better search behaviour in the applicability test of the learnt method because only one instantiation for the star operator has to be found.

Finally, an idea for more long-term future research is to model the powerful human learning capability in theorem proving more adequately. For this, it would be necessary to model how humans introduce new vocabulary for new (emerging) concepts (e.g., representing associative expressions as lists of terms in the expressions, annotations in rippling [5, 11]). With our approach, we cannot do that, however. It is a very challenging question left for future projects.

7 Conclusion

In this paper we described a hybrid system LEARNΩMATIC, which is based on the ΩMEGA proof planning system enhanced by automatic learning of new proof methods. This is an important advance in addressing such a difficult problem, since it makes significant steps in the direction of enabling systems to better their own reasoning power. Proof methods can be either engineered or learnt. Engineering is expensive, since every single new method has to be freshly engineered. Hence, it is better to learn, whereby we have a general methodology that enables the system to automatically learn new methods. The hope is that ultimately, as the learning becomes more complex, the system will be able to find better or new proofs of theorems across a number of problem domains.

A demonstration of LEARNΩMATIC implementation can be found on the following

web page: `http://www.cs.bham.ac.uk/~mmk/demos/LearnOmatic/`.

Acknowledgements

The main part of this work was done when the authors were working for the School of Computer Science of The University of Birmingham. Many people were generous with their time and advice and their help was invaluable to us. In particular, we would like to thank Alan Bundy, Predrag Janičić, Achim Jung, Stephen Muggleton, and Julian Richardson for their helpful advice on our work, and Andreas Meier, and Volker Sorge for their help with some of the implementation in ΩMEGA. This work was supported by an EPSRC Advanced Research Fellowship GR/R76783, EPSRC grants GR/M22031 and GR/M99644, SFB 378 Project grant, and European Commission IHP Calculemus Project grant HPRN-CT-2000-00102.

References

[1] C. Benzmüller, L. Cheikhrouhou, D. Fehrer, A. Fiedler, X. Huang, M. Kerber, M. Kohlhase, E. Melis, A. Meier, W. Schaarschmidt, J. Siekmann, and V. Sorge. ΩMEGA: Towards a mathematical assistant. In W. McCune, editor, *14th Conference on Automated Deduction*, number 1249 in Lecture Notes in Artificial Intelligence, pages 252–255. Springer Verlag, 1997.

[2] A. Brazma. Learning regular expressions by pattern matching. Technical Report TCU/CS/1994/1, Institute of Mathematics and Computer Science, University of Latvia, 1994.

[3] A. Bundy. The use of explicit plans to guide inductive proofs. In E. Lusk and R. Overbeek, editors, *9th Conference on Automated Deduction*, number 310 in Lecture Notes in Computer Science, pages 111–120. Springer Verlag, 1988. Longer version available from Edinburgh as DAI Research Paper No. 349.

[4] A. Bundy. A critique of proof planning. In Antonis C. Kakas and Fariba Sadri, editors, *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski*, number 2408 in Lecture Notes in Computer Science, pages 160–177. Springer Verlag, 2002.

[5] A. Bundy, A. Stevens, F. van Harmelen, A. Ireland, and A. Smaill. Rippling: A heuristic for guiding inductive proofs. *Artificial Intelligence*, 62:185–253, 1993. Also available from Edinburgh as DAI Research Paper No. 567.

[6] J. Denzinger and S. Schulz. Learning domain knowledge to improve theorem proving. In M.A. McRobbie and J.K. Slaney, editors, *13th Conference on Automated Deduction*, number 1104 in Lecture Notes in Artificial Intelligence, pages 62–76. Springer Verlag, 1996.

[7] R.V. Desimone. Learning control knowledge within an explanation-based learning framework. In I. Bratko and N. Lavrač, editors, *Progress in Machine Learning – Proceedings of 2nd European Working Session on Learning, EWSL-87*, Wilmslow, UK, 1987. Sigma Press. Also available from Edinburgh as DAI Research Paper 321.

[8] A. Fiedler. *P.rex*: An interactive proof explainer. In R. Goré, A. Leitsch, and T. Nipkow, editors, *Automated Reasoning – Proceedings of the First International Joint Conference, IJCAR-01*, number 2083 in Lecture Notes in Artificial Intelligence, pages 416–420. Springer Verlag, 2001.

[9] M. Fuchs and M. Fuchs. Feature-based learning of search-guiding heuristics for theorem proving. *AI Communications*, 11:175–189, 1998.

[10] E. Furse. Learning university mathematics. In C.S. Mellish, editor, *Proceedings of the 14th IJCAI*, volume 2, pages 2057–2058. International Joint Conference on Artificial Intelligence, Morgan Kaufmann, 1995.

[11] D. Hutter. Guiding inductive proofs. In M.E. Stickel, editor, *10th Conference on Automated Deduction*, number 449 in Lecture Notes in Artificial Intelligence, pages 147–161. Springer Verlag, 1990.

[12] A. Ireland. The use of planning critics in mechanizing inductive Proofs. In A. Voronkov, editor, *International Conference on Logic Programming and Automated Reasoning, LPAR-92*,

number 624 in Lecture Notes in Artificial Intelligence, pages 178–189. Springer Verlag, 1992. Also available from Edinburgh as DAI Research Paper 592.

[13] M. Jamnik, M. Kerber, and M. Pollet. Automatic learning in proof planning. In F. van Harmelen, editor, *Proceedings of 15th ECAI*, pages 282–286. European Conference on Artificial Intelligence, IOS Press, 2002.

[14] M. Jamnik, M. Kerber, and M. Pollet. LEARNΩMATIC: System description. In A. Voronkov, editor, *18th Conference on Automated Deduction*, number 2392 in Lecture Notes in Artificial Intelligence, pages 150–155. Springer Verlag, 2002.

[15] C.A. Knoblock. An analysis of ABSTRIPS. In James Hendler, editor, *Artificial Intelligence Planning Systems: Proceedings of the First International Conference (AIPS-92)*, pages 126–135. Morgan Kaufmann, 1992.

[16] T. Kolbe and J. Brauburger. PLAGIATOR — A Learning Prover. In W. McCune, editor, *14th Conference on Automated Deduction*, number 1249 in Lecture Notes in Artificial Intelligence, pages 256–259. Springer Verlag, 1997.

[17] T. Kolbe and C. Walther. Reusing Proofs. In A. Cohn, editor, *Proceedings of the 11th ECAI*, pages 80–84. Wiley, New York, 1994.

[18] A. Meier and V. Sorge. Exploring properties of residue classes. In M. Kerber and M. Kohlhase, editors, *Symbolic Calculation and Automated Reasoning: The Calculemus 2000 Symposium*, pages 175–190, Natick, MA, 2001. A K Peters.

[19] E. Melis and A. Meier. Proof planning with multiple strategies. In J. Loyd, V. Dahl, U. Furbach, M. Kerber, K. Lau, C. Palamidessi, L.M. Pereira, Y. Sagivand, and P. Stuckey, editors, *First International Conference on Computational Logic*, volume 1861 of *Lecture Notes in Artificial Intelligence*, pages 644–659. Springer Verlag, 2000.

[20] E. Melis and J.H. Siekmann. Knowledge-based proof planning. *Artificial Intelligence*, 115(1):65–105, 1999.

[21] E. Melis and J. Whittle. Analogy in inductive theorem proving. *Journal of Automated Reasoning*, 22(2), 1998.

[22] S. Muggleton. *Inductive Acquisition of Expert Knowledge*. Addison-Wesley, Reading, MA, 1990.

[23] G. Plotkin. A note on inductive generalization. In D. Michie and B. Meltzer, editors, *Machine Intelligence 5*, pages 153–164. Edinburgh University Press, Edinburgh, UK, 1969.

[24] G. Plotkin. A further note on inductive generalization. In D. Michie and B. Meltzer, editors, *Machine Intelligence 6*, pages 101–126. Edinburgh University Press, Edinburgh, UK, 1971.

[25] G. Pólya. *How to solve it*. Princeton University Press, Princeton, NJ, 1945.

[26] J.D.C. Richardson and A. Smaill. Continuations of proof strategies. In R. Gore, A. Leitsch, and T. Nipkov, editors, *Short Papers of International Joint Conference on Automated Reasoning, IJCAR-01*, pages 130–139, 2001.

[27] J.D.C. Richardson, A. Smaill, and I. Green. System description: proof planning in higher-order logic with lambdaclam. In C. Kirchner and H. Kirchner, editors, *15th Conference on Automated Deduction*, number 1421 in Lecture Notes in Artificial Intelligence, pages 129–133. Springer Verlag, 1998.

[28] P.S. Rosenbloom, J.E. Laird, and A. Newell. *The Soar Papers: Readings on Integrated Intelligence*. MIT Press, 1993.

[29] S. Schulz. *Learning Search Control Knowledge for Equational Deduction*. Number 230 in DISKI. Akademische Verlagsgesellschaft Aka GmbH Berlin, 2000.

[30] B. Silver. Precondition analysis: Learning control information. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning 2*, Palo Alto, CA, 1984. Tioga Press.

[31] R. Sun and L. Giles, editors. *Sequence Learning: Paradigms, Algorithms, and Applications*, number 1828 in Lecture Notes in Artificial Intelligence. Springer Verlag, 2000.

[32] M. Veloso, J. Carbonell, A. Pérez, D. Borrajo, E. Fink, and J. Blythe. Integrating planning and learning: The PRODIGY architecture. *Journal of Experimental and Theoretical Artificial Intelligence*, 7(1):81–120, 1995.

Received November 2003

Theory-Contraction is NP-Complete

NEIL TENNANT, *Department of Philosophy, 230 North Oval Mall, The Ohio State University, Columbus, Ohio 43210, E-mail: tennant.9@osu.edu*

Abstract

I investigate the problem of contracting a dependency-network with respect to any of its nodes. The resulting contraction must not contain the node in question, but must also be a *minimal mutilation* of the original network. Identifying successful and minimally mutilating contractions of dependency-networks is non-trivial, especially when non-well-founded networks are to be taken into account. I prove that the contraction problem is NP-complete.[1]

1 Caveat

The reader needs to know at the very outset what this paper sets out to do, and what it does not. I treat here of finite belief-systems. My interest is in how a finite belief-system can be contracted with respect to any belief in it—how, that is to say, one can give the belief up while at the same time inflicting minimal mutilation on the original belief-system. More precisely, I am interested in determining the computational complexity of the following decision problem: given a finite belief-system T, and any belief p therein, is there a contraction of T with respect to p of such-and-such finite size (less than that of T)?

This paper has only one theoretical aim. I aim to show that the foregoing decision problem is NP-complete. To that end, I need to provide the formal definitions that are required both for the formulation of the problem and for rigorous proof of the claim that the problem is NP-complete. I have no other theoretical aims here than this one. The aim is important enough on its own. It is well known that there are thousands of NP-complete decision problems. But that fact does not entitle one to dismiss the main result of this paper as 'just another NP-completeness theorem'. That would be to underestimate the important of this particular decision problem for the logic of theory dynamics and for epistemology at large. This anticipatory

[1]The main ideas of this paper were first presented to the Interdisciplinary Research Seminar on Mechanization of Inference at The Ohio State University in Autumn Term 1996. More recent versions have been presented to the Ohio State Re-usable Software Research Group; the University of Colorado Colloquium on History and Philosophy of Science; and the Fourth In-House Workshop of the Center for Philosophy of Science at the University of Pittsburgh. Thanks are due to the members of those audiences for comments that have helped clarify the presentation. The Seminar on Mechanization of Inference was funded by a generous grant from the OSU Office of Research, and by grants from the Department of Philosophy and the Center for Cognitive Science. OSU has given me further support in the form of a Special Research Assignment for Spring Term 2003. I am grateful to Victor Marek and Harvey Friedman for helpful discussions in connection with this topic. I would also like to thank Sam Buss, for a careful reading, helpful observations, and an important simplification of my original proof of Theorem 1; and Anish Arora, for the central insight in the proof of Theorem 2. It was Anish who first encouraged me to isolate just the result on NP-completeness for publication as a theoretical piece. In the event, the paper had to expand beyond its original formal limits, in order to meet referees' demands for more informal explanation of the motivation behind my modelling. Hence the current expository texture of the paper, which I hope the general reader will find helpful. Any defects that remain are entirely the author's responsibility.

defence depends, of course, on the formal modelling being accepted as a good first approximation of the epistemologically important phenomena. But arguing in detail that it should be so accepted lies beyond the scope of this paper. So too does the proof of a certain metatheorem, due to Harvey Friedman, that ensures that one incurs no loss of theoretical generality by restricting oneself to *finite* belief-systems.

Despite my focus on proving the main theoretical result about NP-completeness, I am very concerned to make the ideas behind my modelling, and that result, easily accessible to the reader who is not a specialist in the area of belief-revision. For the sake of clarity, therefore, I provide illustrations of defined notions along the way. I also make some comments, which I keep to a minimum, about the motivation behind my modelling of belief-systems and behind the formal definitions that give it expression. These comments are expository only, intended to help the reader who is unfamiliar with the general area. They are not intended as a remotely adequate substitute for the kind of discussion that would be required in order both to justify the motivation behind my chosen form of modelling and to show that it has been given ideal expression in the formal definitions chosen. Such a discussion would need to deal with a host of philosophical, epistemological, logical and mathematical issues. These issues simply lie beyond the scope of this paper. I have already, to some extent, discussed these issues elsewhere (see [16], [17] and [18]) and I intend to discuss them in even greater detail in future publications.

I ought also to give the reader yet further advice as to what this paper is not. It is not the kind of paper that specifies any algorithms. Nor is it the kind of paper that reports on implementations of algorithms. Both the specification and the implementation of algorithms for contracting belief-systems are discussed in detail in other papers that I have already written, and which I intended to submit for publication only after this purely theoretical paper had been accepted. For one needs to know that the computational problem is NP-complete in order to be able to assess any proposed algorithms for solving it. The actual algorithm that I do propose is a development (in successively less greedy versions) of what I called the 'staining algorithm' in [16].

Nor am I, in this paper, concerned to provide detailed critiques of rival accounts of belief-revision and/or contraction, such as *AGM*-theory or JTMS-theory. That, too, has already been undertaken elsewhere (see [16] and [18]), and is to be developed further in future papers. In this paper I confine myself to making the briefest comments possible on both *AGM*-theory and JTMS-theory, simply so that the reader can appreciate how the present approach differs from those.

2 Introduction

The task of belief-revision can be stated as follows. Suppose one is an ideally rational agent, and one's current system of belief implies the proposition p. Suppose that p has just been falsified. Thus one wishes to believe not-p rather than p, and to mutilate one's belief-system as little as possible in coming to do so. First, one contracts the belief-system with respect to p—giving up p and anything in the system that either implies, or wholly depends on, p. Then one adds not-p to the contracted result.

There should be a rigorous account of this operation of contraction. Epistemologists and philosophers of science talk of 'minimally mutilating changes in the web of belief', but give no constructive account of how to effect them. Hence we face the challenge

of giving a philosophically sound, mathematically precise account of contraction, programmable on a computer. The account will be normative, not descriptive: it will reveal what ought to be done by an ideally rational agent confronted with the need to change its mind. Automation of the task would bring widespread applications.

The aim in this paper is to set out an abstract and completely general framework for the discussion of systems of belief and their contractions; to explicate what exactly is meant by minimal mutilation; to characterize the contraction-problem precisely; and to show that this problem is NP-complete.

3 Historical Background

There are two main approaches to the problem of theory-contraction. One is from mathematical logic; the other is from artificial intelligence. My approach is distinct from both of these, seeking to avoid different difficulties inherent in each. I devote a subsection to each of these other approaches, in order to highlight the respects in which my approach differs.

3.1 AGM-*theory*

The approach from mathematical logic, so-called AGM-theory, is due to Alchourrón, Gärdenfors and Makinson. (See Gärdenfors [11] for an account of AGM-theory.) AGM-theory treats belief-systems as theories, that is, as logically closed (hence infinite) sets of sentences. When a theory T is contracted with respect to one of its member-sentences p, the resulting contraction $(T - p)$ is required to satisfy certain conditions, which are set out as the postulates of AGM-theory. The AGM-theorist's aim is then to show that certain functions defined on T and p will produce outputs $(T - p)$ satisfying these postulates. These functions are not, in general, computable, because of their infinitistic character and the undecidability of the underlying relations of logical consequence, reference to which is essential for the theoretical constructions involved. Hence AGM-theory does not offer a convincing prospect for computer-implementation of its methods of contraction.

One of the most important postulates of AGM-theory is the postulate of recovery. Recovery is the claim that if one contracts a theory T with respect to p, and subsequently reinstates p, then one will recover all the original consequences of T. This principle can be counterexemplified in situations for which one's pre-theoretical intuitions are compelling. In this connection see Niederée [14], at pp. 323–4; Levi [13], at pp. 134–5; Hansson [12], at pp. 252–3; and Tennant [16], at pp. 870–4. Recovery violates the following basic intuition: If one believes the proposition b only because one believes the proposition a, and one is giving up a, then one will give up b as well.

My account of theory-contraction differs from AGM-theory on three important points: implementability (on which I insist); the postulate of recovery (which I abandon); and the requirement of minimal mutilation (which I ensure). Moreover, the first difference, on implementability, arises because I am not concerned with (necessarily infinite) logical closures; instead, all the objects and structures with which I operate are finite, and all relations among them are effectively decidable. Moreover, there is a principled philosophical and metalogical justification for restricting our treatment to the finite and computable. This rests on a result proved by Harvey Friedman,

to the effect that there are at most finitely many logically minimal sets of axioms implying any theorem, provided only that the theory in question is axiomatized by means of finitely many axioms and finitely many 'normal' axiom-schemes. (Every extant axiomatic theory in mathematics meets this condition.)

3.2 Truth-maintenance systems

My account of theory-contraction also differs significantly from the prevailing paradigm, among AI-theorists, of so-called truth-maintenance systems (TMSs), reason-maintenance systems (RMSs), and justification-based truth-maintenance systems (JTMSs). (These originate from the work of Doyle [7]. For an integrated overview, see Forbus and de Kleer [9].) My account differs from the various kinds of TMS-theorizing in several ways.

First, unlike the JTMS-theorist (see Forbus and de Kleer [9], ch. 7), I take seriously the possibility of coherentist epistemologies according to which beliefs may be held without their being grounded in beliefs of any privileged class (such as observation reports or sense-datum reports). That is to say, I permit the general possibility of a belief within a belief-system having no well-founded pedigree of support terminating on what the JTMS-theorist calls 'enabled assumptions'. I can still accommodate well-founded belief-systems as a special (and obviously very important) case; but my formal theory accommodates non-well-founded belief-systems as well.

Accommodating the non-well-founded case is sufficient to distinguish my approach from JTMS-theory, *even if* some coherentist epistemologists were to quibble that 'abandoning one belief may mean the remaining beliefs no longer cohere and a radically different set of beliefs need to be adopted'. If the coherentist's complaint here is that I am giving her short shrift, I can simply point out that JTMS-theory gives her even shorter shrift. Moreover, the coherentist has yet to provide a really precise explication of exactly what coherentism consists in, in so far as 'local' relations of justificatory support among beliefs are concerned. There is the very real prospect that at least one systematic explication of coherentism would allow for local relations of justificatory support, and would allow, further, for the general possibility described above—namely, that a belief within a belief-system might have no well-founded pedigree of support terminating on initial beliefs that are themselves in no need of any further support. Moreover, on my approach there is also the prospect of providing convincing evidence for the very claim quoted from my imaginary coherentist critic a few lines back.

Secondly, my primary theoretical focus is on inferential *steps* (identified by their immediate premises and conclusions) rather than on *nodes* (which correspond to sentences, be they premises or conclusions, or both). Any computational treatment of a subject-matter is greatly facilitated by the appropriate choice of data-structure for theoretical manipulation and reasoning. I contend that the most appropriate data-structure is that of the step, not the (labelled) node. I shall not immediately expound on this contention, but shall rather let the ensuing development of ideas and techniques bear it out.

The final and most important difference between my approach and all forms of TMS-theory, including JTMS-theory, is that I tackle the most general problem of how to contract a belief-system with respect to *any* belief that the [J]TMS-theorizer

regards as 'in'. I do not confine the operation of contraction to be simply a matter of 'retracting assumptions', to use the terminology of JTMS-theory. The procedure of 'assumption retraction' is the most complex procedure on offer from current JTMS-theorizing. This procedure is fully deterministic. Given a belief-system $\mathcal{B}$, and any one of its enabled assumptions a, the result of retracting a is *uniquely* determined by $\mathcal{B}$ and a. By contrast, the procedure of 'retracting' a *distant consequence* c of the enabled assumptions within a belief-system (equivalently: contracting the beief-system with respect to c) is in general highly non-deterministic. It is this non-deterministic problem that we address here.

4 The modelling: informal explanation of general features

The formal definitions in the following section will be easy to understand if the reader bears in mind the intended modelling.

4.1 Nodes

Nodes—featureless and unstructured—are to represent particular beliefs within the belief-system of a rational agent. They represent, as it were, specific propositions which the agent might or might not believe. Nodes are the basic objects of which we treat. Everything else that I shall be talking about will be built out of nodes by forming hereditarily finite sets.

4.2 Steps

The agent's beliefs will be represented as standing in relations of support made explicit in what I shall call (justificatory) *steps* involving nodes. A step is fully specified by specifying its finitely many *premises* and its *conclusion*.

I impose a simple rational constraint at the outset. Note that no rational agent would ever seek to justify a belief by appealing both to that belief *and yet other beliefs*. This rational reluctance can be subsumed as a special case of the following more general constraint. *No rational belief system may contain two distinct steps with the same conclusion, one of whose premises are premises of the other step.* Put another way: *steps in a rational belief system require all their premises.* If an agent comes to realize, of a given step in her system, that she can reach its conclusion by using some but not all of the premises of that step, then she will discard the weaker step in favor of the stronger step that uses the weaker (reduced) set of premises.

From now on I assume that all belief systems under discussion satisfy the foregoing rational constraint.

A step whose conclusion is distinct from each of its premises is called *transitional*. Although nodes are basic, in that steps are built up out of them, we shall find that steps themselves are the most convenient data-type with which to work. Note that an agent can know of a step without being committed to any of its nodes, as beliefs. But if an agent knows of a step all of whose premise-nodes she believes, then I take her to be committed to believing the conclusion of that step as well.

Given the rational constraint above, any step with more than one premise will be transitional. For steps with exactly one premise, there are two possibilities: those

where the conclusion is distinct from the premise and hence are transitional; and those where the conclusion is identical to the premise and hence are non-transitional.

In the latter case, we acquire a convenient way of representing an *initial* belief: one that is believed outright, in the sense that as far as the agent is concerned, it requires no further support from any other beliefs that the agent happens to hold. I say in such a case that the belief p in question can be thought of as 'depending on itself'. This does *not* mean that the belief p is *a priori*, and therefore unable ever to be repealed. It means only that the agent believes p outright, in the sense that, as far as she is concerned, she would be justified in believing p even if p were to enjoy no justificatory support from any other beliefs within her scheme. This is of course compatible with p also enjoying such support in certain cases. To make vivid the point that initial beliefs are not necessarily *a priori*, I should point out that my modelling of finite belief-systems could confine itself to just the *contingent* beliefs of the agent. In my representation of the agent's beliefs, a belief's 'depending on itself' (i.e., being believed outright) will be registered by means of a single-premise step, whose premise is the very same node as its conclusion. By adopting this convention we attain a certain smoothness in our mathematical treatment. Note, however, that it is compatible with p's being an initial belief that p stand also as the conclusion of some other, transitional, step, all of whose premises the agent believes. Such would be the case, for example, if p were a simple observational belief that the agent had acquired in the usual way, but for which the agent also had theoretical support—by way of, say, prediction within a higher-level theory on the basis of yet other observational beliefs.

Any rational agent who has a system of beliefs might treat some of these only as initial, that is to say, might believe them outright, without offering yet other beliefs in support of them. In all other cases, a belief will have support within the agent's system. That is, the belief in question will stand as the conclusion of at least one transitional step that involves at least one premise, and all of whose premises are in turn believed. Naturally, a belief can be 'over-determined', by enjoying support from many distinct premise-sets, each of which is *sufficient*, from the agent's point of view, to justify the belief in question. This is what makes the contraction problem, in general, so interesting. For, in giving up such a belief p, the agent will have to give up at least one premise within the premise-set of *each* step that has p as its conclusion.

4.3 Closure

In my modelling, a transitional step is understood as being something of which the agent is aware, or which the agent has acknowledged, as rationally compelling. Hence it is a normative requirement that the agent's system of beliefs should be 'closed' with respect to all such steps. One is, in effect, saying to the agent 'Since you yourself acknowledge the step from $x_1, \ldots, x_n$ to y as compelling, and since you believe each of $x_1, \ldots, x_n$, you should also believe y'. Note that such obligations are *agent-relative*, since the nodes are featureless. I am not adverting to the logical structures of the sentences expressing the beliefs $x_1, \ldots, x_n$ and y, and insisting that the agent should recognize the step as logically valid. Rather, it is taken as a given that the agent herself, for whatever reason, has already acknowledged the step in question as compelling. Her obligation is now only that of *taking the step* to y should she ever

come to believe all of $x_1, \ldots, x_n$.

Let us represent the step from $x_1, \ldots, x_n$ to y by means of the notation

$$x_1, \ldots, x_n \mid y.$$

The reader is advised not to confuse my use of '|' here with other widespread uses of that symbol. The latter include the use on which it is read as 'such that' within a set-abstract like '$\{x \mid F(x)\}$'; as well as its use in logic programming, where it means 'given', as in procedural rules of the form 'goal | sub-goal$_1$, ..., sub-goal$_n$'. My use of '|' might look like a single turnstile shorn of its horizontal stroke: to be sure, premises occur to its left, and a conclusion occurs to its right. But what is designated overall is a justificatory *step* (for the agent). No claim of logical deducibility is made or implied by the use of '|'. For, first, '$x_1, \ldots, x_n \mid y$' is a complex *noun phrase*, meaning 'the step whose premises are the nodes $x_1, \ldots, x_n$ and whose conclusion is the node y'. By contrast, '$\varphi_1, \ldots, \varphi_n \vdash \psi$' is a *statement*, to the effect that there is a proof of the sentential conclusion ψ from the sentential premises $\varphi_1, \ldots, \varphi_n$. Secondly, the nodes are *featureless*, and devoid of logical structure. That is not to say, however, that an agent might not know of a certain step because of an actual deductive proof in her possession, leading from the sentential premises corresponding to the premise-nodes of the step in question to the sentential conclusion corresponding to its conclusion-node. It is just to say that when such a step is represented as a transition, not among sentences, but among featureless nodes, such deductive provenance can be, and is, suppressed. This is in the interests of revealing only such justificatory 'macrostructure' within a belief-system as is relevant for the purposes of contraction. This structure resides at a grosser and more abstract level than that of the detailed logical forms of sentences themselves.

In the course of the agent's investigations, she can come to accept or acknowledge many transitional steps of the form just discussed—

$x_1, \ldots, x_n \mid y$

—while yet failing to believe all of $x_1, \ldots, x_n$. The belief y might fail to enjoy any support, within the agent's belief-system, from sufficient sets of beliefs (from the agent's point of view). In such situations the known steps with y as conclusion are not useless. For, if ever the agent were to come to believe all the premises of any such step, its conclusion y would thereby acquire rational support (from her point of view). This could happen, for example, if the agent at first believed all of $x_1, \ldots, x_{n-1}$ but did not believe x_n; and subsequently came to believe x_n. At that point, since the step

$x_1, \ldots, x_n \mid y$

is known to the agent, she would be rationally obliged to adopt the belief y. That is, she would have to close her beliefs under the step in question.

Clearly, such obligatory operations are iterable, with respect to all known steps all of whose premises have come to be believed. But the iterations will only ever be finite in number. This is because only finitely many steps are known to the agent, and each step involves only finitely many premises. The need for these iterable operations arises only when the agent adopts one or more new beliefs.

Now it is important not to fall prey to the thought that since the agent may know of schematic rules which correspond to an infinity of steps, her belief-system must be infinite. I reiterate the point that my nodes are featureless, and their labels are not to be thought of as endowed with any essentially logical structure. Each node represents a particular belief, as given by a particular proposition, expressible by a particular sentence. No agent will ever have justified to herself more than finitely many such beliefs. Nor will she ever have performed, in her thinking, more than finitely many steps. Indeed, at no time in the future of humankind will the *communal* belief-system, even assuming perfect memory, ever contain more than finitely many beliefs, or involve more than finitely many steps. I call this the *finitary predicament*. Fortunately, it is what makes a computational account of contraction and revision possible! To put the matter succinctly: for the purposes of belief-representation, *schemata are irrelevant; only their instances count*. And at any point in time, there will have been only finitely many instantiations.

I shall call the outcome of the iterable operations described above the *universal closure* of the agent's belief-system, within the set of steps known to the agent, and with respect to the agent's set of newly-adopted beliefs. I am assuming that in this context there is no possibility of confusing this kind of universal closure with the logician's syntactic notion of the universal closure of a formula with free variables, which closure is obtained by prefixing the formula with universal quantifiers to bind its free variables. Universal Closure is called 'propagate inness' by JTMS-theorists; see Forbus and de Kleer [9].

The situation can be schematized as follows. Let $\mathcal{C}$ be the finite set of all transitional steps known to the agent. Every transitional step in the agent's current belief-system $\mathcal{B}$ (which itself is finite) will be in $\mathcal{C}$; but in general there could be (transitional) steps in $\mathcal{C}$ that are not in $\mathcal{B}$. Moreover, $\mathcal{B}$ can contain *initial* steps, none of which is in $\mathcal{C}$. Note that in the belief-system $\mathcal{B}$, every node of every step is believed, whether it be a premise or a conclusion of the step in question.

By way of illustration, take the current belief-system $\mathcal{B}$ to consist of the steps

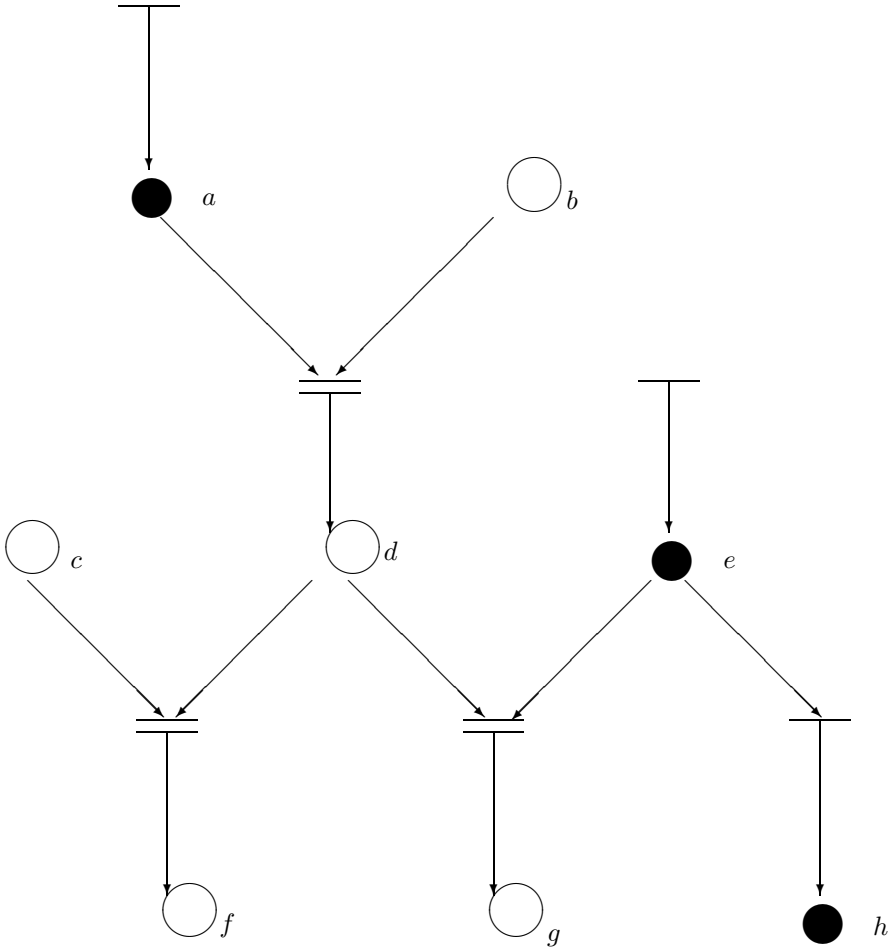
$a \,|\, a$ (initial)
$e \,|\, e$ (initial)
$e \,|\, h$ (transitional)

and let the set $\mathcal{C}$ of transitional steps known to the agent be

$e \,|\, h$ (note: this is also in $\mathcal{B}$)
$a, b \,|\, d$
$c, d \,|\, f$
$d, e \,|\, g$

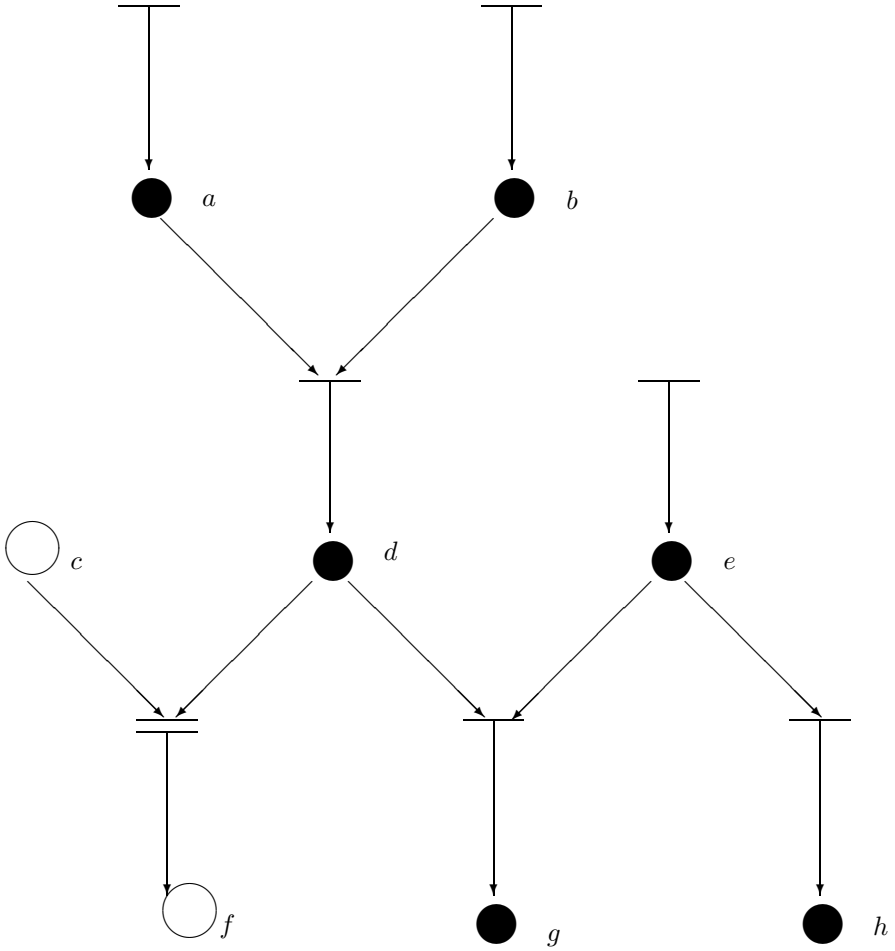
The following diagram represents the combination of $\mathcal{B}$ and $\mathcal{C}$. The black nodes represent the actual beliefs of the agent. These are conclusions of steps in $\mathcal{B}$. The white nodes represent propositions that are not believed, but which feature in transitional steps in $\mathcal{C} \setminus \mathcal{B}$. The initial beliefs (which are all in $\mathcal{B}$) are each furnished with an 'initial arrow' emanating from a horizontal stroke that itself receives no arrows. Horizontal strokes represent the inferential transitions made in steps. The inferential direction is downwards. In general, more than one arrow can come down to a single inference stroke, from the premises of the step in question; while only one arrow descends from

the stroke, to the conclusion of the step in question. If a step has all its premise-nodes black, then its horizontal stroke is also 'black', this being shown by a single black line. Such a step will be in $\mathcal{B}$. But if a step has at least one white premise-node, then its horizontal stroke is accordingly 'white', this being shown by two black lines with white space between them. Such a step is in $\mathcal{C}$, but not in $\mathcal{B}$.



Let $\mathcal{P}$ now be a finite set of nodes not in $\mathcal{B}$. The nodes in $\mathcal{P}$ are to be understood as 'newly adopted' initial beliefs. If, upon adopting all the new beliefs in $\mathcal{P}$, the agent finds that she now believes all the premises of some step in $\mathcal{C} \setminus \mathcal{B}$, then she is obliged to close her beliefs under that step, thereby possibly acquiring a further new belief (namely, the conclusion of the step in question). The eventual outcome of iterating the operation will be the universal closure discussed above. In symbols, it is the *universal closure of* $\mathcal{B}$ *within* $\mathcal{C}$ *with respect to* $\mathcal{P}$—abbreviated $[\mathcal{B}; \mathcal{P}]_{\mathcal{C}}$.

Let us illustrate the process of universal closure with the diagram above. Suppose we take $\mathcal{P}$ to be $\{b\}$. Then the universal closure $[\mathcal{B}; \mathcal{P}]_{\mathcal{C}}$ is easily seen to be the 'black' part of the following diagram:



By talking of nodes and of steps involving them, I am dealing with ideas familiar to workers in AI and computer science. What I have been calling 'belief-systems' modelled by means of nodes and steps are better known to workers in these areas as *dependency-networks*. Intuitively, a dependency-network is a model of a rational agent's belief-system, whose nodes represent the sentences or propositions believed.

Dependency-networks can in general contain *sub*networks. The task of contracting a network T with respect to one of its nodes p will be the task of finding a suitable subnetwork: one which does not contain the node p, but which satisfies the rational closure requirement with respect to the transitional steps in T, and which is, in a suitable sense to be explicated below, *maximal*. That is to say, the subnetwork R in question must be the outcome of a *minimally mutilating* process of contraction of the original network T. And the subnetwork R must itself be closed with respect to the transitional steps in T.

Why require such closure? Because the subnetwork R is supposed to represented a new rational 'reflective equilibrium' for the agent. Such an equilibrium will have been achieved by the agent after carrying out the contraction-process in question only if she avails herself of all the justificatory transitions *that she knows about*. The

justificatory transitions in question are exactly the transitional steps in T. We must remember that in 'giving up' a step, the agent is not losing, or eschewing, or no longer claiming to be aware of, the *justificatory transition* that it represents; rather, she is simply reaching a state in which she no longer believes the conclusion of that step, and no longer believes all its premises. The step itself, however, is still there for her to use; and use it she ought, when rationally closing her new system of beliefs, which is a *subsystem* of her original system T. Put graphically: in contractions, what can change is not the overall structure of the diagram, but only the colors of certain of its nodes (from black to white, or *vice versa*).

I shall explicate the notion of maximality below. It is important to realize that in general a network T might contain more than one maximal closed subnetwork not containing p. That is, the operation of contracting T with respect to p can in general yield more than one result. This will be reflected in the non-deterministic character of any algorithms that one might devise for performing contractions.

I have encountered the complaint, from theorists in the AGM-tradition, that any 'procedure for contraction' should tell one *exactly which* contraction to adopt. But this is not, in general, possible! Contraction is not *functional*. That is, contraction is not an operation with uniquely defined values. No matter what extra information one avails oneself of, in the way of relative entrenchment of beliefs (say), there is always the logical and mathematical possibility that *there is more than one equally good contraction of a given system of beliefs with respect to any chosen one of the beliefs therein*. My account has the virtue of accepting this possibility (which tends to be the rule rather than the exception), and turning to the task of how one accounts for *all* the eligible contractions in the multiple-valued case. Indeed, the contraction algorithm(s) to be described in another paper systematically generate all the eligible contractions; and in the implementation that I favor, in Prolog, this is achieved by back-tracking from one good solution to another, until all the equally good solutions have been enumerated.

5 The modelling: formal treatment

5.1 Nodes and steps

Definition 5.1 A pair (U, x) is called a *step* just in case
(1) U is a non-empty finite set of nodes and x is a node, and
(2) x is in U if and only if $U = \{x\}$.
U is called the *premise set* of the step (U, x); x is called the *conclusion* of the step (U, x).

Some terminological clarification is in order here. What I am here calling steps are called 'justifications' in the JTMS-literature. I prefer the shorter, neutral designation because not every step is a justification in the usual epistemological sense. In general, justifications (in the usual sense) would consist of more than one step. Note also that 'premises', in the JTMS-literature, are taken to be assumptions that cannot be retracted. This too is a non-standard use of the logician's word 'premise'. The logician speaks of premises for a step of inference, without necessarily endorsing those premises as true.

If U=$\{y_1, \ldots, y_n\}$ then the step (U, x) can be written symbolically as

$y_1, \ldots, y_n \,|\, x$.

It is easy to see that for a given x there are only two possible forms of step with x as conclusion:
$y_1, \ldots, y_n \,|\, x$ where each y_i is distinct from x $(1 \leq i \leq n)$; and
$x \,|\, x$.
I shall call these *transitional* and *initial* steps respectively.

5.2 Universal closure of belief-systems

The informal explanation of universal closure given in the previous section should enable the reader to understand the following statement, in pseudocode, of the algorithm for universal closure.

Algorithm 5.1: UNIVERSALCLOSURE($\mathcal{C}, \mathcal{B}, \mathcal{P}$)

Steps $\leftarrow \mathcal{C}$
Closure $\leftarrow \mathcal{B} \cup$ the set of initial steps involving members of $\mathcal{P}$
while some step in *Steps* but not in *Closure* has each of its premises standing as the conclusion of a step in *Closure*
do {
  NewSteps $\leftarrow$ the set of all such steps
  Closure $\leftarrow$ *Closure* $\cup$ *NewSteps*
  Steps $\leftarrow$ *Steps* $\setminus$ *NewSteps*
}
return (*Closure*)

The algorithm must terminate, since $\mathcal{C}$ is finite. I say that it determines the *universal* closure (of $\mathcal{B}$ within $\mathcal{C}$ with respect to $\mathcal{P}$) because of the word 'each' in the **while** condition.

Universal closure is exceptionally simple. Indeed, even the prima-facie more difficult problem of deciding whether a given step can be derived using finitely many given steps is solvable in linear time. In this connection, see Dowling and Gallier [6]. The problem is exactly that of querying a Prolog program with finitely many *propositional* clauses. A conditional Prolog clause such as '*a* :- *b*, *c*.' is precisely the transitional step $b, c \,|\, a$. An unconditional Prolog clause '*a*.' is precisely the initial step $a \,|\, a$. To find whether, say, the step $p, q, r \mid s$ follows from finitely many other steps, simply form the Prolog program consisting of the clauses corresponding to the latter steps, adjoin the Prolog clauses '*p*.', '*q*.' and '*r*.' to the program, and query the expanded program with *s*.

5.3 Dependency-networks

Definition 5.2 Let T be a set of steps, called *T-steps*. For any step (U, x) in T, U is said to be *x-generating* or *x-justifying* (in T).

Definition 5.3 Again, let T be a set of steps. T is a *dependency-network* if and only if T is finite and

(1) for every (U, x) in T and for every y in U, there is some (V, y) in T; and
(2) if (U, x) and (W, x) are in T, then W is not a proper subset of U.

Condition (1) in the definition of a dependency-network ensures that every node involved in any way in T is justified in T. Any conclusion x of a T-step (U, x) is justified simply because of the premise set U of that step.

Condition (2) is the rational constraint discussed earlier. It ensures that a dependency-network only ever uses the most succinct justifications available. There is no point in having the step (U, x) in a dependency-network if for some proper subset W of U we have (W, x).

I shall write '$x\,|\,x$' more economically as $|\,x$.

Remark 1: It is evident from the definition that a dependency-network need not have a well-founded justificatory structure. Note, however, that the non-well-foundedness is confined to looping, and does not arise from infinite descending chains. This is because networks are finite.

Remark 2: In general there can be more than one T-step of the form (U, x) for any given node x. Condition (2) in the definition of a dependency-network tells us that each such U is a 'minimal generating set' for x: that is, U implies x but (as far as the agent is aware) no proper subset of U implies x. In future I shall often suppress the adjective 'minimal'; but it should be borne in mind that it always applies.

Definition 5.4 $|\,T\,|$, the *core* of T, is the set of nodes x such that for some U, T contains the step (U, x).

An agent whose beliefs are modelled by a dependency-network T believes exactly what is in $|\,T\,|$, and for the reasons articulated by the steps in T. An initial node in T will be a node p for which T contains the initial step $(\{p\}, p)$.

Definition 5.5 A node q is said to be 'in' a dependency-network T (and conversely: T 'contains' q) just in case some step (U, q) with conclusion q is a member of T. (*Cf.* Doyle [7], p. 234.) Recall that a dependency-network is defined as a set of steps. Note that every member of any such premise set U will also be 'in' T, because of condition (1) in the definition of a dependency-network. Thus any node involved in any T-step is 'in' T. With this understanding of the preposition 'in', we can say (loosely) that a node is in $(T \setminus R)$ when what is meant, strictly, is that it is in $(|\,T\,| \setminus |\,R\,|)$.

Definition 5.6 R is a *subnetwork* of T just in case every step of R is a step of T and R is a dependency-network. I also say that the network T *extends* the subnetwork R.

Remark 3: Note that a subnetwork of T, since it is a dependency-network in its own right, is 'closed upwards' modulo T. That is to say, a subnetwork of T will not have any nodes of T as initial nodes if they are not also initial in T. Instead, the subnetwork will supply for each of its non-initial nodes at least one of the patterns of

justification with which it is furnished within the containing network T.

Example to illustrate Remark 3. Let T be the dependency-network

$$|p,\ \ |q,\ \ p|r,\ \ q|r.$$

The node r is the only non-initial node. Any subnetwork of T that contains r will have to contain also one of its patterns of justification within T. As it happens, there are two distinct such patterns: one leading from the initial node p, the other leading from the initial node q. Thus the only possible proper, non-empty subnetworks of T are

$$|p,\ \ p|r \qquad \text{and} \qquad |q,\ \ q|r.$$

Both these subnetworks terminate their justifications for r on initial nodes. When closing upwards modulo T, however, one does not in general always terminate in this fashion on initial nodes. For example, consider the non-well-founded network

$$p|q,\ \ q|p,\ \ p|r,\ \ q|r.$$

The only subnetwork of T that contains the node r is T itself. For, as we close upwards from r modulo T, we must include either p or q as a node. Each of these, in turn, requires the other, in order to have any T-justification at all. But neither of them is initial.

Remark 4. Not every subset of a dependency-network is a subnetwork.

Example to illustrate Remark 4. $\{(\{b\}, a)\}$ is a subset but not a subnetwork of the dependency-network $\{(\{b\}, a), (\{a\}, b)\}$. $\{(\{b\}, a)\}$ fails to be a dependency-network because b enjoys no justification within it.

Now let R be an arbitrary subnetwork of T.

Definition 5.7 R is *closed downwards* modulo T if and only if for every step (U, x) in T, if $U \subseteq |R|$, then (U, x) is in R.

Definition 5.8 The *T-closure* of R is the least subnetwork S of T satisfying the following:
(1) R is a subnetwork of S; and
(2) S is closed downwards modulo T.

This definition has as an immediate consequence that the T-closure of R is closed upwards modulo T. (See Remark 3 above.)

I shall write $[R]_T$ for the T-closure of R, and sometimes suppress mention of T when T is clear from the context. An intuitive understanding of T-closure is as follows. The subnetwork R of T, before such closure, will, as a dependency-network in its own right, contain justifications for every one of its nodes (as a conclusion). But it might also contain all the premises (*i.e.* members of U) of some T-step (U, x) without containing the conclusion x itself. The requirement of T-closure is that R should then be expanded by adding (U, x) to R and thereby justifying x 'within' R.

It does not follow that every conclusion of T itself would in due course be added in this way. The closure process could be carried out to its fullest possible extent while yet the T-closure fall properly short of T itself. This would happen when $|R|$ failed to encompass enough of the members of $|T|$ for the T-steps to effect a full enough expansion during closure. Clearly also, a proper subnetwork R of T can be properly contained in its own T-closure. The following remark summarizes these observations.

Remark 5. In general we have that R is a (possibly proper) subset of its T-closure $[R]_T$, which in turn is a (possibly proper) subset of T.

Example to illustrate Remark 5. $R = \{(\{a\}, a), (\{b\}, b)\}$ is a proper subnetwork of
$T = \{(\{a\}, a), (\{b\}, b), (\{a, b\}, c), (\{d\}, d)\}$
but the T-closure of R is $\{(\{a\}, a), (\{b\}, b), (\{a, b\}, c)\}$. So here we have R properly contained in its own T-closure, which in turn is a proper subnetwork of T.

Remark 6: $[R]_T$ can be computed from R and T in polynomial time.

Definition 5.9 A subnetwork R of T is *T-closed* if and only if R is its own T-closure, that is, just in case $R = [R]_T$.

5.4 Contraction formally defined

I shall impose three basic adequacy requirements on any contraction R of a dependency-network T with respect to any of its nodes p.

Definition 5.10 The following conditions on a subnetwork R of T are individually necessary and jointly sufficient for R to be a *contraction of T with respect to p* :

1. (Honesty) R is a T-closed subnetwork of T.
2. (Success) $|R|$ does not contain p.
3. (Minimal Mutilation) R is inclusion-maximal with regard to (1) and (2).

Remember that T-closure involves only the application of steps that are never called into question during the contraction process.

The Honesty condition says that one should not be able to 'close' $T{-}p$ any further by means of T-steps. Thus a contraction contains all nodes known to be consequences of any nodes that it contains.

The Success condition says that $T-p$ should not contain any step of the form (U, p). Put another way, p should not be in $|T{-}p|{=}|[T{-}p]_T|$.

The Minimal Mutilation condition ensures that the contraction is as economical as possible, in the sense that one gives up as little as possible in the way of T-steps when passing from T to $T-p$. The condition seeks to conserve the outputs of past computational effort in developing T. As de Kleer [5], p. 129, put it: 'Thus inferences, once made, need not be repeated ...'. I am therefore requiring the contraction to be a maximally non-(p generating), T-closed subnetwork of T. So a contraction $T{-}p$ will contain all those T-steps compatible with the requirement that p not be in $|T{-}p|$.

6 Results

6.1 The complexity of contraction

First I show that the contraction problem (understood as a decision problem) is in NP. To be precise, the decision problem in question is this:

> Given T and given an integer k, decide whether there is a T-closed subnetwork R not containing p, which is of size at least k.

Theorem 6.1 The contraction problem is in NP.

PROOF. The NP algorithm guesses R of size k and decides, in polynomial time, whether it is T-closed and is a subnetwork and does not contain p. ∎

Theorem 6.2 The contraction problem is NP-complete.

PROOF. By Theorem 6.1, the contraction problem is in NP. It remains to show that some NP-complete problem can be transformed in polynomial time into the contraction problem. To this end, consider the class of networks all of whose transitional steps have exactly two self-justifying premises, and conclusion p. Clearly any contraction of such a network (with respect to p) that is of size k will be determined by a non-p-generating subset, of size k, of the set $\mathcal{P}$ of self-justifying premises. Thus the problem of finding a contraction of size k with respect to p boils down to the problem of finding a subset of $\text{card}(\mathcal{P}) - k$, whose members are to be removed from the network. (The length of input for the latter problem is no less than some fixed fraction of the length of input for the contraction problem; such a linear factor will not affect the resulting complexity class for the contraction problem.) But this is isomorphic to the NP-complete Vertex Cover (VC) Problem, which is to find a subset (of a given size) of the set of vertices of a graph that contains, for each edge in the graph, at least one of the two vertices incident on that edge. (See Garey and Johnson [10], p. 46.) In the case of contraction I have described, we are looking, analogously, for a subset (of a given size) of the set of premises involved in two-premise steps of the network that contains, for each two-premise step in the network, at least one of its premises. (Naturally, we seek to minimize this subset in order to maximize the resulting contraction.) The isomorphism with graphs is easy to see. Construe nodes in the network as vertices in a graph. The (unordered) pair-sets of premises of the steps can then be construed as edges of the graph, which are (unordered) pair-sets of vertices. One's choice of a set of premise-nodes to be removed (at least one from each pair-set) in order for the common conclusion of all the two-premise steps to have no justification then corresponds to a choice of a set of vertices in the graph that covers each edge, by containing at least one vertex from each edge, i.e. from each pair-set of vertices. ∎

A good contraction algorithm would aim to enumerate all the contractions of a given network T with respect to any of its nodes p, without any particular interest in maximum-*sized* contractions. A contraction, by definition, is inclusion-maximal in a certain regard, and there will in general be many a contraction that is not of maximum size. The method of proof of Theorem 6.2, however, shows that the problem of searching for a contraction is NP-hard.

7 Discussion

7.1 Significance of our results for extensions of JTMS-theory

As far as *giving up* beliefs is concerned, extant JTMS-theory provides only the algorithmic operation of 'retracting an enabled assumption'. In the terminology of this paper, this is the restricted operation of contracting with respect to an *initial* node. (See Forbus and de Kleer [9], ch. 7.) This restricted operation is deterministic, and it is easy to provide for it a polynomial-time algorithm.

By contrast, I have shown here that the general, non-deterministic operation of contracting with respect to *any* node, whether initial or non-initial, is intractable. My contraction operation includes the JTMS-theorist's operation of assumption-retraction as a special case, and generalizes it appropriately for the retraction of nodes in general, in what is obviously the canonical way. Retracting a node p (equivalently: contracting with respect to a node p) is a general operation of obvious importance. The upshot of our investigations is therefore: JTMS-theory cannot be fully general in its treatment of important operations on dependency-networks except at the cost of intractability.

7.2 Logical problems are usually intractable

In any area of logic, it would be naive to expect our algorithms (when we have them) to be tractable, in the technical sense of being computable in polynomial time. After all, the simplest kind of logical problem—that of determining, of a given sentence of a propositional language, whether it is has a satisfying truth-value-assignment—is the original and most famous NP-complete problem (Cook [4]). So the problem of deciding theoremhood in classical propositional logic is co-NP-complete. Changing 'classical' to 'intuitionistic' makes matters prima facie worse: theoremhood in intuitionistic propositional logic is P-space complete (Statman [15]). In well-known systems of *relevant* propositional logic, the decision problem can be even worse. Indeed, theoremhood in the Anderson-Belnap system R of relevant logic (Anderson and Belnap [1]) is not even decidable (Urquhart [20]). Its well-known decidable fragment LR (Thistlethwaite *et al.* [19]) obtained by dropping the distributivity axiom, is at best ESPACE-hard, at worst space-hard in a function that is primitive recursive in the generalised Ackerman exponential (Urquhart, [20]).

Adding the first-order quantifiers 'some' and 'all' to the logical system brings undecidability almost everywhere. Classical first-order logic is decidable when only monadic predicates are involved, but is undecidable as soon as one adds a single two-place predicate (Church [3], [2]). Intuitionistic first-order logic is even worse: it becomes undecidable as soon as there are two one-place predicates. (This is a result of Saul Kripke. See Dummett [8], pp. 209-213.)

One should not be at all surprised or dejected, then, to find that the problem of belief-contraction is 'intractable' in the sense of being NP-complete or possibly worse. The interesting challenge was how to conceive of the problem in a manner sufficiently amenable to complexity analysis in the first place. It is in this spirit that we have conducted the foregoing investigations. The result has been encouraging. We have discovered that theory-contraction, conceived in this way, is of the lowest complexity that one could have expected it to be. Moreover, theorists of contraction who object

(misguidedly, in my view) to representing belief-systems in my finitary fashion now have to contend with the prospect that their own 'contraction decision-problems' (in so far as they might be able to define any, on their approaches) have to contend with NP-completeness as a *lower* bound of computational complexity. If NP-completeness is bad news for the dependency-network theorist, then it will be just as bad, if not worse, for *AGM*-theorists—if ever they produce any computational implementations of their contraction functions.

7.3 Future work

In planned sequels to this paper, which are already written, I specify various successively less greedy versions of a contraction algorithm; provide Prolog programs that implement them; and demonstrate how these programs produce intuitively satisfying results on all extant contraction problems canvassed in the literature.

There are also two main ways in which the simple modelling set out above could be extended, so as to take into account other features of belief-systems considered important by the epistemologist. The first extension would enable one to take into account, when performing contractions, a relation of *relative entrenchment* among nodes. This would be a partial ordering in general, providing information of the form 'if either a or b has to be given up, then it should be a rather than b'.

The second extension of the simple modelling would enable one to represent the agent as willing and able, in the course of a contraction, to give up one of the *steps* within the dependency-network. The present modelling treats the steps themselves as immune to revision, and allows only nodes to be given up. Allowing steps themselves to be vulnerable in contractions, however, can have dramatic knock-on effects, especially when one considers that a given step might be an instance of a more general rule which would itself be impugned should the step be given up. A fully detailed study of this problem is well under way, but its detailed results, both computational and philosophical, will have to await another occasion. Suffice it to say here that the modelling in this paper can be extended so as to allow for the retraction of steps as well as nodes. This is a possibility which, once explored, reveals why thinkers are rightly loathe to tamper with their logic—for the knock-on effects are enormous. It is also a possibility whose closer scrutiny commends an inviolable core of logic, namely the logic needed in order to underpin the contraction-process itself. Finally, as far as retracting steps as well as nodes is concerned: for the theoretical purposes of the present paper, it has been important *not* to countenance this possibility. That the contraction problem is NP-complete when *only nodes* are given up is all the more arresting.

In closing, I should point out that I have opted for a considerable degree of abstraction. I have abstracted away from details such as the internal logical structure of the beliefs represented by nodes, and what underlying logic might be employed to justify steps within a belief system. I have sought instead to provide a representation of what is essentially involved in the process of contraction. It remains to be investigated whether indeed my chosen level of theoretical representation might have some 'psychological reality'. Perhaps it is (or could be) part of the design of a good cognitive system that its 'contraction and revision module' operate at the level of abstraction employed here. The linguistic system furnishes complex grammatical structures for

the expression of one's beliefs; and one's logical system generates various (derived) steps, for the deductive development of one's beliefs. For the purposes of orderly, rational and efficient revision or updating, however, the only structure that is required is that captured by the kind of dependency-networks employed above. Between the actual sentences of one's language, and the nodes in such networks, there need only be a system of 'pointers'. Contraction and revision can proceed at the more abstract level of nodes and steps, prescinding altogether from distracting questions about the internal logical structure of the nodes themselves, and the kind of logic that justifies the steps.

References

[1] Alan Ross Anderson and Nuel D. Belnap, Jr. *Entailment: The Logic of Relevance and Necessity. Vol. I.* Princeton University Press, 1975.

[2] Alonzo Church. Correction. *Journal of Symbolic Logic*, 1:101–102, 1936.

[3] Alonzo Church. A note on the Entscheidungsproblem. *Journal of Symbolic Logic*, 1:40–41, 1936.

[4] S. A. Cook. The complexity of theorem-proving procedures. *Proc. 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[5] J. de Kleer. An assumption-based TMS. *Artificial Intelligence*, 28:127–162, 1986.

[6] W. F. Dowling and J. H. Gallier. Linear-time alrgorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming*, 1:267–284, 1984.

[7] John Doyle. A truth maintenance system. *Artificial Intelligence*, 12:231–272, 1979.

[8] Michael Dummett. *Elements of Intuitionism.* Clarendon Press, Oxford, 1977.

[9] K. Forbus and J. de Kleer. *Building Problem-Solvers.* MIT Press, Cambridge, MA, 1993.

[10] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Bell Laboratories, Murray Hill, NJ, 1979.

[11] Peter Gärdenfors. *Knowledge in Flux.* MIT Press, Cambridge, MA, 1988.

[12] S. O. Hansson. Belief contraction without recovery. *Studia Logica*, 50:251–260, 1991.

[13] Isaac Levi. *The Fixation of Belief and Its Undoing.* Cambridge University Press, 1991.

[14] R. Niederée. Multiple contraction. a further case against Gärdenfors' Principle of Recovery. In A. Fuhrmann and M. Morreau, editors, *The Logic of Theory Change. Lecture Notes in Artifical Intelligence 465.* Springer Verlag, Berlin, Heidelberg, New York, 1987.

[15] Richard Statman. Intuitionistic propositional logic is polynomial space complete. *Theoretical Computer Science*, 9:67–72, 1979.

[16] Neil Tennant. Changing the theory of theory change: Towards a computational approach. *British Journal for Philosophy of Science*, 45:865–897, 1994.

[17] Neil Tennant. Changing the theory of theory-change: reply to my critics. *British Journal for Philosophy of Science*, 48:569–586, 1997.

[18] Neil Tennant. On having bad contractions: or, no room for recovery. *Journal of Applied Non-Classical Logics*, 7:241–266, 1997.

[19] Paul Thistlewaite, Michael A. MacRobbie, and Robert K. Meyer. *Automated Theorem-Proving in Non-Classical Logic. Research Notes in Theoretical Computer Science.* Pitman, London and Wiley, New York, 1987.

[20] Alasdair Urquhart. The undecidability of entailment and relevant implication. *Journal of Symbolic Logic*, 49:1059–1073, 1984.

Received September 2003

Contents for Volume 11

Issue No. 1

Issue No. 2

Issue No. 3

Issue No. 4

Issue No. 5

Issue No. 6

Interest Group in Pure and Applied Logics (IGPL)

The Interest Group in Pure and Applied Logics (IGPL) is sponsored by The European Association for Logic, Language and Information (FoLLI), and currently has a membership of over a thousand researchers in various aspects of logic (symbolic, mathematical, computational, philosophical, etc.) from all over the world (currently, more than 50 countries). Our main activity is that of a research and information clearing house.

Our activities include:

- Exchanging information about research problems, references and common interest among group members, and among different communities in pure and applied logic.
- Helping to obtain photocopies of papers to colleagues (under the appropriate copyright restrictions), especially where there may be difficulties of access.
- Supplying review copies of books through the journals on which some of us are editors.
- Helping to organise exchange visits and workshops among members.
- Advising on papers for publication.
- Editing and distributing a Newsletter and a Journal (the first scientific journal on logic which is FULLY electronic: submission, refereeing, revising, typesetting, publishing, distribution; first issue: July 1993): the Logic Journal of the Interest Group on Pure and Applied Logics. (For more information on the Logic Journal of the IGPL, see the Web homepage: http://www.oup.co.uk/igpl)
- Keeping a public archive of papers, abstracts, etc., accessible via ftp.
- Wherever possible, obtaining reductions on group (6 or more) purchases of logic books from publishers.

If you are interested, please send your details (name, postal address, phone, fax, e-mail address, research interests) to:

IGPL Headquarters
c/o Prof. Dov Gabbay
King's College, Dept of Computer Science
Strand
London WC2R 2LS
United Kingdom
e-mail: dg@dcs.kcl.ac.uk

For the organisation: Dov Gabbay, Ruy de Queiroz and Hans Jürgen Ohlbach.