Ruitenburg's Theorem mechanized and contextualized

Tadeusz Litak
FAU Erlangen-Nürnberg
tadeusz.litak@fau.de

In 1984, Wim Ruitenburg published a surprising result about periodic sequences in intuitionistic propositional calculus (IPC). The property established by Ruitenburg naturally generalizes local finiteness (intuitionistic logic is not locally finite, even in a single variable). However, one of the two main goals of this note is to illustrate that most "natural" non-classical logics failing local finiteness also do not enjoy the periodic sequence property; IPC is quite unique in separating these properties. The other goal of this note is to present a Coq formalization of Ruitenburg's heavily syntactic proof. Apart from ensuring its correctness, the formalization allows extraction of a program providing a certified implementation of Ruitenburg's algorithm.

1 Introduction

In 1984, Wim Ruitenburg [39] published a surprising result about periodic sequences in intuitionistic propositional calculus (IPC). For quite a while, the result seemed relatively neglected. One of the few researchers making extensive use of it was the late Sergey Mardaev [29, 26, 28, 30, 27]. It also got mentioned in Humberstone's monograph on logical connectives [23]. Recently, however, there has been renewed interest [15, 18, 16, 17], although as we are going see, Ruitenburg's discovery appears to deserve still broader attention.

Consider a propositional formula A. Fix a propositional variable p, which can be thought of as representing the context hole or the argument of A taken as a polynomial (other propositional variables being additional constants). Given any other formula B, write $A(B)$ for the result of substituting B for p. Also, write $A \equiv_L B$ for $\vdash_L A \leftrightarrow B$, where L is a chosen system of propositional logic. Now define the natural iterated substitution operation

$$A^0(p) := p, \quad A^{n+1}(p) := A(A^n(p)).$$

Such a sequence turns almost immediately into a cycle modulo $\equiv_{\mathsf{CPC}}$, i.e., the equivalence relation of Classical Propositional Calculus:

Lemma 1 ([39], Lemma 1.1)**.** *For any A,* $\quad A(p) \equiv_{\mathsf{CPC}} A^3(p)$.

The above observation can be reformulated as asserting that CPC has *uniformly globally periodic sequences* (ugps). A logic L has this property if there exist b, $c > 0$ s.t. for any formula A, $A^b(p) \equiv_L A^{b+c}(p)$. However, ugps has still a rather strong logical form: two existential quantifiers preceding an universal one. Hence one can consider changing the order of quantifiers to weaken the property:

(eventually) periodic sequences:

	globally			locally		
uniformly	$\exists b.$	$\exists c > 0.$	$\forall A.$	$\exists c > 0.$	$\forall A.$	$\exists b.$
parametrically	$\exists b.$	$\forall A.$	$\exists c > 0.$	$\forall A.$	$\exists b.$	$\exists c > 0.$

$A^b(p) \equiv_L A^{b+c}(p)$

So, do standard non-classical propositional calculi, IPC in particular, have at least *plps* (*parametrically locally periodic sequences*)?[1] If not, is there something special about CPC which makes results such as Lemma 1 possible?

One of many peculiarities of CPC as seen from the general perspective of *abstract algebraic logic* [4, 12] is that it is *finite*, i.e., determined by a single finite algebra of truth values. There are some other natural examples of finite logics (mostly some fuzzy logics of finite chains, such as Łukasiewicz three-valued logic), but in general, this property is rather rare among logics of importance in today's Computer Science. A somewhat more general property is *local finiteness*: a logic is locally finite if given a finite set of propositional variables, one can form only finitely many non-equivalent formulas. The modal system S5 is a typical example of a logic which is locally finite without being finite.

Lemma 2. *Any locally finite logic has plps.*

Sketch. Any sequence $p, A(p), A^2(p), A^3(p)\ldots$ disproving the plps property would also disprove local finiteness. □

It is, however, well-known that IPC is not locally finite: even in one propositional variable, there are infinitely many nonequivalent formulas. The exact description of the infinite algebra of formulas in one propositional variable is provided by the Rieger-Nishimura Theorem (see [38, 32] and [6, Ch. 7] for references, also Appendix D herein). As we are going to see in Section 2, most "natural" propositional logics which fail to be locally finite, fail to have (even parametrically locally) periodic sequences. IPC turns out to fare better. One can indeed show that (uniformly or parametrically) globally periodic sequences would be too much to expect, at least when formulas are allowed to contain other variables than p itself [39, §2]. But we do have

Theorem 3 ([39], Theorem 1.9)**.** IPC *has the ulps property: for any A, there exists b s.t.* $A^b(p) \equiv_{\mathsf{IPC}} A^{b+2}(p)$. *Moreover, b is linear in the size of A.*

In fact, Ruitenburg's theorem is effective: the proof provides an algorithm to compute b in question; cf. Section 3. Moreover, as the periodic sequence property (in all its incarnations) transfers from sublogics to extensions in the same signature (just like local finiteness and unlike uniform interpolation), we also get that all superintuitionistic logics (*si-logics*) have ulps. This shows that unlike local finiteness, ulps does *not* guarantee the finite model property (*fmp*), or even Kripke completeness.

On the other hand, there is an obvious connection with fixpoint definability[2]: If $A(p)$ is a monotone formula, then $\{A^n(p)\}_{n\in\omega}$ stabilizes when reaching a cycle. In particular, substituting $\bot$ for p in $A^b(p)$ produces the least fixpoint, while substituting $\top$ produces the greatest fixpoints. This is why Mardaev [29, 26, 28, 30, 27] quotes Ruitenburg when investigating the issue of fixpoint definability in non-classical logics. Periodic sequences, however, are by no means the only way of ensuring that a logic has definable fixpoints: there are examples of systems having the latter property without the former. In fact, a combination of Pitts' [34] uniform interpolation with what modal logicians would describe as a

[1] I am not aware whether terminology and distinctions used in the present paper have been systematically introduced before. The original work of Ruitenburg uses the term *finite order*.

[2] In fact, the present author got interested in Ruitenburg's result for similar reasons, in the context of ongoing joint work with Albert Visser on definability of fixpoint in intuitionistic modal logics involving (a strong version of) the Löb axiom. I would like to thank Albert Visser for attracting my attention to Ruitenburg's work, for his support and comments on early drafts. Thanks are also due to Wim Ruitenburg for providing his recollections of how the theorem was proved. Furthermore, George Metcalfe kindly corrected my misunderstandings concerning the status of RM ("R with Mingle") and provided me with several additional references. Finally, I would like to thank the referees of this and earlier incarnations of this paper, Alexis Saurin, Lutz Schröder and the participants of Oberseminar of our group at FAU for discussions and suggestions.

(definable) *master modality*[3] plus some trivial additional restrictions would be sufficient. Ghilardi et al. [15, 16] discuss further the issue of computing fixpoints and fixpoint definability, and compare the two approaches. It is worth mentioning here that Ruitenburg in the final part of his paper suggests a potential connection with uniform interpolation, despite preceding Pitts' [34] by several years.

Finally, as suggested by a referee, a clarification might be in order. The reader might be aware of results on definability of fixpoints in modal logics (classical or intuitionistic ones) containing some form of the Löb axiom. Nevertheless, as pointed out in Corollaries 5 and 6, such logics generally fail the plps property. Such definability results concern *guarded* or *modalized* fixpoints, i.e., those where the fixpoint variable occurs only in the scope of a modal operator. Van Benthem [1] and Visser [44] illustrate how to use definability of such fixpoints to derive definability of ordinary, monotone fixpoints: Precisely here one can use the periodic sequence property for the underlying modality-free (*extensional*) reduct of the logic in question.

The purpose of this note is twofold. In Section 2, I discuss the status of periodic sequences in other non-classical logics, illustrating just how special the situation of IPC is. In Section 3 and appendices, I present a mechanization of Ruitenburg's result in the Coq proof assistant.[4]

1.1 Related work

In 2015–2016, when the mechanization described in Section 3 was produced, Ruitenburg's original and poorly understood syntactic proof[5] was the only available one. This in fact motivated the present author to take on the challenge of mechanizing the proof, despite a relatively limited experience with proof assistants at the time. In the meantime, Ghilardi and Santocanale [18, 17] provided a semantic proof using the apparatus developed in the Ghilardi and Zawadowski monograph [14], involving games for bounded bisimulations and (pre)sheaves over the category of finite rooted posets with bounded morphisms. Nevertheless, as Ghilardi and Santocanale admit, their semantic proof does not provide tight bounds and computational information provided by Ruitenburg's proof, and extracted by the Coq mechanization described in Section 3. Furthermore, the hope expressed in their final remark

> While we can expect that periodicity phenomena of substitutions do not arise for the basic modal logic K, they surely do for locally tabular [i.e., locally finite] modal logics. Considering also the numerous results on definability of fixpoints ... these phenomena are likely to appear in other subsystems of modal logics. As far as we know, investigation of periodicity phenomena in modal logics is a research direction which has not yet been explored and where the bounded bisimulation methods might prove their strength once more.

in the light of Section 2 herein requires qualification: outside of locally finite modal logics, there seems to be little hope and scope for periodicity. Open Problem 1 below isolates one potential intuitionistic modal logic for which a generalization of Ruitenburg result might be possible. In fact, the mechanization described here can be used in investigating the problem; cf. Remark 10 in Appendix C.

[3] This is only needed if the logic in question contains additional "modal" connectives or lacks some structural rules. For intutionistic propositional logic itself, the requirement of having a "master modality" (global deduction theorem, equationally definable principal congruences...) is trivially satisfied, just like for standard relevance logics. On the other hand, this criterion is not generally met by substructural logics such as those covered by Theorem 8. They generally fail to satisfy axioms ensuring EDPC [13, Theorem 3.55]. Same problems arise with non-transitive modalities, even in the classical unimodal setting.

[4] The content of both sections is based on the work done in years 2015–2017, which for various reasons remained unpublished and presented only in the form of a talk at TACL 2017.

[5] It is worth mentioning here that Wim Ruitenburg himself (p.c.) claims that his original proof was utilizing Kripke semantics. Difficulties in explaining it to his colleagues, in particular Albert Visser, and Visser's additional insights finally recorded as Lemma 1.7 in Ruitenburg's paper, convinced him to cast the argument into a purely syntactic setting, which at the time proved clear enough to both Ruitenburg and Visser.

Of all Coq formalizations of non-trivial results concerning various propositional calculi, the recent work of Férée and van Gool [11] is probably closest to our interests here. It deals with Pitt's syntactic proof of uniform interpolation for IPC, which as discussed above provides another route towards fixpoint definability, and it also allows extraction of executable code, actually computing propositional quantifiers[6]. As Pitts' proof was cast in the setting of the terminating sequent calculus G4ip [22, 9] (Ruitenburg, by contrast, works with a slightly idiosyncratic and purely Hilbert-style setting, as discussed in Section 3 and Appendix A), Férée and van Gool [11] mechanizes some metatheory of that calculus, in particular admissibility of a restricted form of cut and other structural rules. The present mechanization simply assumes decidability of IPC and does not attempt to provide either a syntactic proof via cut elimination or a Kripke-based semantic one, although such developments are available elsewhere.

Finally, there is an entire recent body of work mechanizing in Coq G4ip-style calculi for several propositional logics (over classical and intuitionistic base) developed by Shillito and coauthors [42, 41, 21]. It would seem of interest to turn the present formalization into a part of a larger library, integrating the developments described above and possibly other scattered contributions, such as the Coq development supporting the discussion of modal negative translations in Litak et al. [25].

2 Periodic sequences in nonclassical logics

In order to understand properly the special status of Ruitenburg's result, let us compare the situation in IPC with that in other non-classical logics, e.g., substructural or modal ones. It turns out that in almost all standard cases, plps (and even more so, ulps) implies local finiteness; IPC seems rather exotic in having the first property without the second one.

2.1 Modal logics over CPC

For modal logics over the boolean propositional base, the reader can refer to, e.g., Chagrov and Zakharyaschev [6] for notation, syntax and semantics; one difference is that I am using here a superscript $\cdot^{\mathsf{cl}}$ to make the CPC propositional base clear. For transitive modal logics, having periodic sequences is indistinguishable from local finiteness, i.e., the converse of Lemma 2 holds:

Theorem 4. *A normal extension of* $\mathsf{K4}^{\mathsf{cl}}$ *has plps iff it is locally finite.*

Proof. It is known [6, Theorem 12.21] that a normal modal logic extending K4 is locally finite iff it is of *infinite depth*, i.e., admits Kripke frames of arbitrary finite depths. Consider $A_{\mathsf{K4}}(p) := q \vee \Box(q \to \Box p)$. A straightforward modification of the argument proving the above equivalence [6, Theorem 12.21] shows the failure of plps (in the proof, the valuation for q should be defined in the same way as the valuation for p): the sequence $\{A^n_{\mathsf{K4}}(p)\}_{n\in\omega}$ never stabilizes. □

Corollary 5. *All extensions of* K^{cl} *contained in either* $\mathsf{S4Grz.3}^{\mathsf{cl}}$ *(such as* $\mathsf{K4}^{\mathsf{cl}}$*,* $\mathsf{S4}^{\mathsf{cl}}$*,* T^{cl}*) or* $\mathsf{GL.3}^{\mathsf{cl}}$ *(in particular* $\mathsf{GL}^{\mathsf{cl}}$*) fail to have locally periodic sequences.*

For subsystems of $\mathsf{GL.3}^{\mathsf{cl}}$, this can be proved via a simpler alternative technique that remains useful when the propositional base is weakened to IPC; see Theorem 7.

Moreover, even without transitivity it does not appear easy to find examples of logics with plps which are not locally finite. Shapirovsky [40] has provided an example of a normal modal logic which

[6] To make the relationship even stronger, the apparatus developed in the Ghilardi and Zawadowski monograph [14] provides a model-theoretic proof of both Pitts' result and Ruitenburg's result. In fact, the starting point for that monograph was their earlier article [19], explicitly motivated as "language-free" or categorical analysis of uniform interpolation. Visser [45] follows a similar approach based on bounded bisimulations, cast in somewhat less categorical terms.

has finitely many formulas in one variable, but fails local finiteness. Unfortunately, the technique used in the proof of Theorem 4 can be also applied to his example with a minor modification: namely, use

$$A_{\mathsf{Sha}}(p) := q \vee \Box(q \to \Box(p \vee r)),$$

where r is to be evaluated as $\{\omega\}$ in Shapirovsky's frame.

2.2 Intuitionistic modal logics

The reader is referred to the extensive literature [43, 47, 46, 24] for basic information about intuitionistic modal logics. Just for clarity, we are only discussing here intuitionistic modal logics with a single modality $\Box$ (no $\Diamond$), which is reflected in the notation. Theorem 4 immediately extends to intuitionistic modal logics being counterparts of standard extensions of K^{cl} (see Simpson's [43] Requirement 3):

Corollary 6. *All extensions of* $\mathsf{K}^{\mathsf{int}}_{\Box}$ *contained in either* $\mathsf{S4Grz.3}^{\mathsf{cl}}$ (*such as* $\mathsf{K4}^{\mathsf{int}}_{\Box}$, $\mathsf{T}^{\mathsf{int}}_{\Box}$,$\mathsf{S4}^{\mathsf{int}}_{\Box}$, $\mathsf{S4Grz.3}^{\mathsf{int}}_{\Box}$ *or* $\mathsf{S4Grz.3}^{\mathsf{int}}_{\Box}$) *or* $\mathsf{GL.3}^{\mathsf{cl}}$ (*in particular* $\mathsf{GL}^{\mathsf{int}}_{\Box}$ *or* $\mathsf{GL.3}^{\mathsf{int}}_{\Box}$) *fail to have locally periodic sequences.*

Some intuitionistic modal logics of computational interest have "degenerate" classical counterparts (see [24] for a discussion) and hence Corollary 6 cannot be used to disprove that they have periodic sequences. This includes $\mathsf{S}^{\mathsf{int}}_{\Box} := \mathsf{K}^{\mathsf{int}}_{\Box} \oplus A \to \Box A$, i.e., the Curry-Howard logic of *applicative functors*, also known as *idioms* [31]. Its classical counterpart S^{cl} and all its two consistent proper extensions are finite logics enjoying ulps. In fact, S^{cl} has exactly two proper consistent extensions, one denoted as Triv and the other denoted as Ver [6]. In contrast, not only does $\mathsf{S}^{\mathsf{int}}_{\Box}$ have uncountably many propositional extensions, but the failure of plps remains a common phenomenon among them. To show this, one can use a proof technique applicable to (subsystems of) logics with Löb-style axioms, either intuitionistic or classical ones:

Theorem 7. *No sublogic of* $\mathsf{KM.3}^{\mathsf{int}}_{\Box}$, *also denoted as* $\mathsf{KM}_{\mathsf{lin}}$ [7] *has parametrically locally periodic sequences; this in particular applies to* $\mathsf{SL.3}^{\mathsf{int}}_{\Box} := \mathsf{S}^{\mathsf{int}}_{\Box} \oplus \mathsf{GL.3}^{\mathsf{int}}_{\Box}$, $\mathsf{SL}^{\mathsf{int}}_{\Box} := \mathsf{S}^{\mathsf{int}}_{\Box} \oplus \mathsf{GL}^{\mathsf{int}}_{\Box}$ *or* $\mathsf{S}^{\mathsf{int}}_{\Box}$.

Proof. The logic $\mathsf{KM.3}^{\mathsf{int}}_{\Box}$ (or $\mathsf{KM}_{\mathsf{lin}}$) is the logic of the Kripke frame where the modal and the intuitionistic order are, respectively, irreflexive and reflexive variant of the reverse order on natural numbers. Consider $A_{\mathsf{KM}}(p) := \Box p$ and the valuation sending p to $\emptyset$; the denotation of $A^n_{\mathsf{KM}}(p)$ is the set of natural numbers smaller or equal to n. Hence, the sequence $\{A^n_{\mathsf{KM}}(p)\}_{n\in\omega}$ never stabilizes and plps fails in every logic sound in this frame. □

To contrast this with Theorem 4, note that $\mathsf{KM.3}^{\mathsf{int}}_{\Box}$, the propositional fragment of the logic of the Mitchell-Bénabou logic of the *topos of trees* [3, 7, 24], is *prefinite* or *pretabular*: all its extensions are finite, each determined by a finite chain. Interestingly, neither the proof Theorem 4 nor the proof of Theorem 7 apply to the Propositional Lax Logic $\mathsf{PLL}^{\mathsf{int}}_{\Box}$ [10], i.e., the Curry-Howard counterpart to (the type system of) Moggi's monadic metalanguage [2].

Open Problem 1. Does $\mathsf{PLL}^{\mathsf{int}}_{\Box}$ have locally periodic sequences?

2.3 Substructural logics

Arguments analogous to those above establish that in the realm of substructural logics [13], plps as a rule coincides with local finiteness. Consider $A_{\otimes}(p) := p \cdot p$, where $\cdot$ is the substructural *fusion* connective [13, §2.1.2], also known by linear logicians as *tensor* or *multiplicative conjunction* $\otimes$ and the realm of the Logic of Bunched Implications BI and separation logic as *spatial*, *separating* or *independent* conjunction $*$ [37].

Theorem 8. *The product logic* Π*, the infinite valued Łukasiewicz logic* $Ł_\infty$ *or the logic of the* heap model *of* BBI (*boolean logic of bunched implications* [5, 33, 35, 37])*fail to have plps. Consequently, the property fails in all their sublogics, including* $(\mathsf{In}-)\mathsf{FL}_{(\mathsf{ew})}$*, multiplicative-additive fragment of linear logic* MALL (*and its intuitionistic fragment* IMALL) *and fuzzy logics such as* BL *or* MTL.[7]

Proof. This is shown by evaluating the sequence $\{A^n_\otimes(p)\}_{n\in\omega}$ defined above in the heap model or the $[0,1]$-interval with corresponding ℓ-norms. □

Thus, in order to find a natural substructural logic L enjoying the plps without local finiteness, one should look at those where the sequence $\{A^n_\otimes(p)\}_{n\in\omega}$ stabilizes modulo $\equiv_L$. It can be naturally achieved by stipulating that fusion is idempotent (both square-increasing and square-decreasing). This, however, is a very restrictive condition. When L satisfies the weakening rule, it collapses fusion to ordinary additive conjunction $\wedge$ and substructural implication to Heyting implication. Idempotent systems where $\cdot$ does not entirely collapse to $\wedge$ are sometimes considered by relevance logicians, with perhaps the most famous example being RM ("R with Mingle") . However, this system has been long known to be locally finite anyway [8]. See more recent references [36, 20] on limits of local finiteness results for idempotent structural logics.

Open Problem 2. Are there natural non-Heyting examples of (idempotent? square-increasing?) substructural logics with the plps property failing local finiteness?

3 Coq formalization

The formalization is available as a git repository

https://git8.cs.fau.de/software/ruitenburg1984.

It consists of less than 4000 lines of Coq code, split into 6 files. The code allows working program extraction to OCaml or Haskell; it can be also used directly for computation using Coq's core functional programming language (Gallina). More on that can be found in Appendix E.

Naturally, the formalization involves a *deep* rather than *shallow* embedding of IPC. The syntax of IPC is formalized from first principles. Also, all semantic discussion (i.e., any mention of Kripke models) from the original paper is omitted. The semantic counterexamples given by Ruitenburg are unproblematic and easy to understand. The important part is purely syntactic.

As the formalization was developed in 2015–2016, and it was an exercise for the author in understanding Ruitenburg's paper and improving his own skills, it does not involve most modern or complex Coq libraries and features. After relatively minor changes, it has proved possible to compile under recent versions of Coq (8.17 and 8.18 at the time of this writing), but the development itself is not using in an essential way anything that was not already available in versions 8.4pl6 and 8.5. Some libraries being used are already getting obsolete, but a proper overhaul would constitute a separate project, focusing directly on the theorem-proving community. Section 1.1 and the work of Férée and van Gool [11] or Shillito and coauthors [42, 41, 21] suggest how such an overhaul could potentially look like. A more detailed discussion is delegated to appendices, which are going to remain available in the online version of this paper (an arXiv preprint/technical report).

3.1 Conclusions

The routes for future development have been already suggested in the paper. I find the question whether there are other natural non-locally-finite logics with llps particularly intriguing (Open Problems 1 and

[7] See Galatos et al. [13] for substructural systems mentioned in the statement of this theorem.

2). Combining the present formalization with some standard proof of decidability of IPC and using it, e.g., to eliminate altogether the meta-level Excluded Middle (Appendix B) or to compute optimal size of a bound for any input (Appendix E) also seems a natural challenge for future work.

References

[1] Johan van Benthem. "Modal Frame Correspondences and Fixed-Points". In: *Studia Logica* 83.1-3 (2006), pp. 133–155.

[2] P. N. Benton, Gavin M. Bierman, and Valeria de Paiva. "Computational Types from a Logical Perspective". In: *J. Funct. Program.* 8.2 (1998), pp. 177–193.

[3] Lars Birkedal et al. "First Steps in Synthetic Guarded Domain Theory: Step-Indexing in the Topos of Trees". In: *LMCS* 8 (4 2012), pp. 1–45.

[4] W. J. Blok and D. Pigozzi. *Algebraizable logics*. Vol. 77 (396). Memoirs AMS. AMS, 1989.

[5] James Brotherston and Max Kanovich. "Undecidability of Propositional Separation Logic and Its Neighbours". In: *J. ACM* 61.2 (Apr. 2014), 14:1–14:43. ISSN: 0004-5411. DOI: 10.1145/2542667. URL: http://doi.acm.org/10.1145/2542667.

[6] A. Chagrov and M. Zakharyaschev. *Modal Logic*. Oxford Logic Guides 35. Oxford: Clarendon Press, 1997.

[7] Ranald Clouston and Rajeev Goré. "Sequent Calculus in the Topos of Trees". In: *Proceedings of FoSSaCS 2015*. Ed. by Andrew M. Pitts. Vol. 9034. LNCS. Springer, 2015, pp. 133–147. ISBN: 978-3-662-46677-3. DOI: 10.1007/978-3-662-46678-0_9. URL: http://dx.doi.org/10.1007/978-3-662-46678-0_9.

[8] J. Michael Dunn. "Algebraic Completeness Results for R-Mingle and Its Extensions". In: *The Journal of Symbolic Logic* 35.1 (1970), pp. 1–13. ISSN: 00224812. URL: http://www.jstor.org/stable/2271149 (visited on 02/01/2024).

[9] Roy Dyckhoff. "Contraction-free sequent calculi for intuitionistic logic". In: *Journal of Symbolic Logic* 57 (1992), pp. 795–807.

[10] Matt Fairtlough and Michael Mendler. "Propositional Lax Logic". In: *Inf. Comput.* 137.1 (1997), pp. 1–33.

[11] Hugo Férée and Sam van Gool. "Formalizing and Computing Propositional Quantifiers". In: *Proceedings of CPP 2023*. Ed. by Robbert Krebbers et al. ACM, 2023, pp. 148–158. DOI: 10.1145/3573105.3575668. URL: https://doi.org/10.1145/3573105.3575668.

[12] Josep Maria Font, Ramon Jansana, and Don Pigozzi. "A Survey of Abstract Algebraic Logic". In: *Studia Logica* 74.1-2 (2003), pp. 13–97.

[13] Nikolaos Galatos et al. *Residuated Lattices: An Algebraic Glimpse at Substructural Logics*. Studies in Logic and the Foundations of Mathematics 151. Elsevier, 2007. ISBN: 0444521410, 9780444521415.

[14] S. Ghilardi and M. Zawadowski. *Sheaves, Games and Model Completions*. Vol. 14. Trends In Logic, Studia Logica Library. Dordrecht: Kluwer, 2002.

[15] Silvio Ghilardi, Maria João Gouveia, and Luigi Santocanale. "Fixed-Point Elimination in the Intuitionistic Propositional Calculus". In: *Proceedings of FoSSaCS 2016*. Ed. by Bart Jacobs and Christof Löding. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 126–141. ISBN: 978-3-662-49630-5. DOI: 10.1007/978-3-662-49630-5_8. URL: http://dx.doi.org/10.1007/978-3-662-49630-5_8.

[16] Silvio Ghilardi, Maria João Gouveia, and Luigi Santocanale. "Fixed-point Elimination in the Intuitionistic Propositional Calculus". In: *ACM Trans. Comput. Log.* 21.1 (2020), 4:1–4:37. DOI: 10.1145/3359669. URL: https://doi.org/10.1145/3359669.

[17] Silvio Ghilardi and Luigi Santocanale. "Free Heyting algebra endomorphisms: Ruitenburg's Theorem and beyond". In: *Math. Struct. Comput. Sci.* 30.6 (2020), pp. 572–596. DOI: 10.1017/S0960129519000203. URL: https://doi.org/10.1017/S0960129519000203.

[18] Silvio Ghilardi and Luigi Santocanale. "Ruitenburg's Theorem via Duality and Bounded Bisimulations". In: *Proceedings of AiML 2018*. Ed. by Guram Bezhanishvili et al. College Publications, 2018, pp. 277–290. URL: http://www.aiml.net/volumes/volume12/Ghilardi-Santocanale.pdf.

[19] Silvio Ghilardi and Marek Zawadowski. "A sheaf representation and duality for finitely presented Heyting algebras". In: *Journal of Symbolic Logic* 60 (1995), pp. 911–939.

[20] José Gil-Férez, Peter Jipsen, and George Metcalfe. "Structure theorems for idempotent residuated lattices". In: *Algebra universalis* 81.2 (2020), p. 28. DOI: 10.1007/s00012-020-00659-5. URL: https://doi.org/10.1007/s00012-020-00659-5.

[21] Rajeev Goré, Revantha Ramanayake, and Ian Shillito. "Cut-Elimination for Provability Logic by Terminating Proof-Search: Formalised and Deconstructed Using Coq". In: *Proceedings of TABLEAUX 2021*. Ed. by Anupam Das and Sara Negri. Vol. 12842. Lecture Notes in Computer Science. Springer, 2021, pp. 299–313. DOI: 10.1007/978-3-030-86059-2_18. URL: https://doi.org/10.1007/978-3-030-86059-2%5C_18.

[22] Jörg Hudelmaier. *Bounds for cut elimination in intuitionistic propositional logic*. Ph.D. Thesis. Tübingen: University of Tübingen, 1989.

[23] Lloyd Humberstone. *The Connectives*. MIT Press, 2011.

[24] Tadeusz Litak. "Constructive modalities with provability smack". In: *Leo Esakia on duality in modal and intuitionistic logics*. Ed. by Guram Bezhanishvili. Vol. 4. Outstanding Contributions to Logic. https://arxiv.org/abs/1708.05607. Springer, 2014. DOI: 10.1007/978-94-017-8860-1_7. URL: https://arxiv.org/abs/1708.05607.

[25] Tadeusz Litak, Miriam Polzer, and Ulrich Rabenstein. "Negative Translations and Normal Modality". In: *2nd International Conference on Formal Structures for Computation and Deduction (FSCD 2017)*. Ed. by Dale Miller. Vol. 84. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, 27:1–27:18. ISBN: 978-3-95977-047-7. DOI: 10.4230/LIPIcs.FSCD.2017.27. URL: http://drops.dagstuhl.de/opus/volltexte/2017/7741.

[26] Sergey I. Mardaev. "Convergence of positive schemes in S4 and Int". In: *Algebra and Logic* 33.2 (1994), pp. 95–101. ISSN: 1573-8302. DOI: 10.1007/BF00739995. URL: http://dx.doi.org/10.1007/BF00739995.

[27] Sergey I. Mardaev. "Definable fixed points in modal and temporal logics—a survey". In: *Journal of Applied Non-Classical Logics* 17.3 (2007), pp. 317–346.

[28] Sergey I. Mardaev. "Fixed points of modal negative operators". In: *Bull. Sect. Log., Univ. Lodz, Dep. Log* 26 (1997), pp. 135–138.

[29] Sergey I. Mardaev. "Least fixed points in Grzegorczyk's logic and in the intuitionistic propositional logic". In: *Algebra and Logic* 32.5 (1993), pp. 279–288. ISSN: 1573-8302. DOI: 10.1007/BF02261708. URL: http://dx.doi.org/10.1007/BF02261708.

[30] Sergey I. Mardaev. "Negative modal schemes". In: *Algebra and Logic* 37.3 (1998), pp. 187–191. ISSN: 1573-8302. DOI: 10.1007/BF02671590. URL: http://dx.doi.org/10.1007/BF02671590.

[31] Conor McBride and Ross Paterson. "Applicative programming with effects". In: *J. Funct. Program.* 18.1 (2008), pp. 1–13.

[32] Iwao Nishimura. "On Formulas of One Variable in Intuitionistic Propositional Calculus". In: *The Journal of Symbolic Logic* 25.4 (1960), pp. 327–331. ISSN: 00224812. URL: http://www.jstor.org/stable/2963526.

[33] Peter W. O'Hearn and David J. Pym. "The Logic of Bunched Implications". English. In: *The Bulletin of Symbolic Logic* 5.2 (1999), pp. 215–244. ISSN: 10798986. URL: http://www.jstor.org/stable/421090.

[34] Andrew M. Pitts. "On an interpretation of second order quantification in first order intuitionistic propositional logic". In: *The Journal of Symbolic Logic* 57 (01 Mar. 1992), pp. 33–52. ISSN: 1943-5886. DOI: 10.2307/2275175. URL: http://journals.cambridge.org/article_S0022481200023161.

[35] David J. Pym, Peter W. O'Hearn, and Hongseok Yang. "Possible worlds and resources: the semantics of BI". In: *Theoretical Computer Science* 315.1 (2004). Mathematical Foundations of Programming Semantics, pp. 257–305. ISSN: 0304-3975. DOI: http://dx.doi.org/10.1016/j.tcs.2003.11.020. URL: http://www.sciencedirect.com/science/article/pii/S0304397503006248.

[36] James Raftery. "Representable idempotent commutative residuated lattices". In: *Trans. Am. Math. Soc.* 359 (Oct. 2007). DOI: 10.1090/S0002-9947-07-04235-3.

[37] John C. Reynolds. "Separation Logic: A Logic for Shared Mutable Data Structures". In: *Proceedings of LiCS 2002*. IEEE Computer Society, 2002, pp. 55–74. ISBN: 0-7695-1483-9. DOI: 10.1109/LICS.2002.1029817. URL: http://dx.doi.org/10.1109/LICS.2002.1029817.

[38] Ladislav Rieger. *On the lattice theory of Brouwerian propositional logic*. Acta Facultatis Rerum Naturalium Universitatis Carolinae 189. F. Řivnáč, Prague, 1949.

[39] W. Ruitenburg. "On the period of sequences $(A^n(p))$ in intuitionistic propositional calculus". In: *Journal of Symbolic Logic* 49 (1984), pp. 892–899.

[40] Ilya B Shapirovsky. "Glivenko's theorem, finite height, and local finiteness". In: *Advances in Modal Logic 2018*. arXiv preprint arXiv:1806.06899. 2018.

[41] Ian Shillito and Rajeev Goré. "Direct elimination of additive-cuts in GL4ip: verified and extracted". In: *Proceedings of AiML 2022*. Ed. by David Fernández-Duque, Alessandra Palmigiano, and Sophie Pinchinat. College Publications, 2022, pp. 429–450.

[42] Ian Shillito et al. "A New Calculus for Intuitionistic Strong Löb Logic: Strong Termination and Cut-Elimination, Formalised". In: *Proceedings of TABLEAUX 2023*. Ed. by Revantha Ramanayake and Josef Urban. Vol. 14278. Lecture Notes in Computer Science. Springer, 2023, pp. 73–93. DOI: 10.1007/978-3-031-43513-3_5. URL: https://doi.org/10.1007/978-3-031-43513-3%5C_5.

[43] Alex K. Simpson. "The Proof Theory and Semantics of Intuitionistic Modal Logic". PhD thesis. University of Edinburgh, 1994. URL: http://homepages.inf.ed.ac.uk/als/Research/thesis.ps.gz.

[44] Albert Visser. "Löb's Logic Meets the μ-calculus". In: *Processes, Terms and Cycles: Steps on the Road to Infinity, Essays Dedicated to Jan Willem Klop, on the Occasion of His 60th Birthday*. Ed. by Aart Middeldorp et al. Vol. 3838. Lecture Notes in Computer Science. Springer, 2005, pp. 14–25. ISBN: 3-540-30911-X.

[45] Albert Visser. "Uniform Interpolation and Layered Bisimulation". In: *Gödel '96, Logical Foundations of Mathematics, Computer Science and Physics —Kurt Gödel's Legacy*. Ed. by P. Hájek. reprinted as Lecture Notes in Logic 6, Association of Symbolic Logic. Berlin: Springer, 1996, pp. 139–164.

[46] Frank Wolter and Michael Zakharyaschev. "Intuitionistic Modal Logics as fragments of Classical Modal Logics". In: *Logic at Work, Essays in honour of Helena Rasiowa*. Ed. by Ewa Orlowska. Springer–Verlag, 1998, pp. 168–186.

[47] Frank Wolter and Michael Zakharyaschev. "On the relation between intuitionistic and classical modal logics". In: *Algebra and Logic* 36 (1997), pp. 121–125.

A Setup and basic lemmas

The language of IPC is defined as usual:

```
Inductive form :=
  | var : nat → form
  | imp : form → form → form
  | and : form → form → form
  | or : form → form → form
  | tt : form
  | ff : form.

Notation "A '&' B " := (and A B) (at level 40, left associativity).
Notation "A '\v/' B " := (or A B) (at level 45, left associativity).
Notation "A '-»' B" := (imp A B) (at level 49, right associativity).

Definition p := var 0.
Definition q := var 1.
Definition r := var 2.
```

Variable p will be used as a distinguished variable of the formula: the input or the argument of a polynomial. It is convenient to make it the variable with index 0 (consider the use of `destruct` in proofs where the distinguished variable should get a special treatment). We also explicitly add equivalence:

```
Notation "A '«->» B" := ((A -» B) & (B -» A)) (at level 58).
```

Ruitenburg's paper uses a formulation of IPC in terms of syntactic consequence (turnstile) relation between (finite) sets of formulas and formulas themselves. This approach is natural from the point of view of abstract algebraic logic [4, 12]. It would be natural to replace this turnstile-Hilbert-style axiomatization by a Gentzen-style formalism, either sequent calculus or natural deduction. We will return to this point in § B. The chosen formalization of IPC, however, is convenient for our purposes and stays as close to the development in Ruitenburg's article as possible (Ruitenburg, in fact, did not write the exact axiomatization he was using or give an explicit reference for it, but it is easy to reconstruct).

Rather unsurprisingly, in the Coq version of the axiomatization I replaced finite sets of formulas with finite lists. The standard Hilbert-style presentation of IPC can be found in numerous references. In my setup, it looks as follows:

```
Reserved Notation "G '|–' A" (at level 63).

Notation context := (list form).

Inductive hil : context → form → Prop :=
 | hilst : ∀ G A, In A G → G |- A
 | hilK : ∀ G A B, G |- A -» B -» A
 | hilS : ∀ G A B C, G |- (A -» B -» C) -» (A -» B) -» A -» C
 | hilMP : ∀ G A B, G |- (A -» B) → (G |- A) → (G |- B)
 | hilC1 : ∀ G A B, G |- (A -» B -» A & B)
 | hilC2 : ∀ G A B, G |- (A & B -» A)
 | hilC3 : ∀ G A B, G |- (A & B -» B)
 | hilA1 : ∀ G A B, G |- (A -» A \v/ B)
 | hilA2 : ∀ G A B, G |- (B -» A \v/ B)
 | hilA3 : ∀ G A B C, G |- (A -» C) -» (B -» C) -» (A \v/ B -» C)
 | hiltt : ∀ G, G |- tt
 | hilff : ∀ G A, G |- ff -» A
                        where "G '|–' A" := (hil G A).
```

A sequence of lemmas follows in `HilbertIPCsetup.v`, establishing basic properties of the turnstile relation. There are also some easy tactics for use in later proofs. They should be all rather self-explanatory. Again, in Ruitenburg's paper trivial lemmas of this kind are used tacitly or nearly tacitly. Several, though by no means all of the basic lemmas in this part were added to the `Hint` database and so were, e.g., constructors of hil. This has in some cases slowed down working of some tactics, in particular `eauto`, but I believe overall the database has not been unduly swollen and `eauto`, `auto` and their cousins remain useful.

One also needs a standard notion of substitution; as the focus is entirely on a propositional language with no notions of—and no problems of—binding, α-conversion etc., I decided to use a straightforward, own formalization, with a minimal number of tailored tactics to make the development smoother. Base substitutions are simply functions from variables to formulas; we can thus reduce them to functions from natural numbers to formulas. They are extended inductively to arbitrary formulas and then to arbitrary contexts:

```
Fixpoint sub (s: nat → form) (A : form) : form :=
  match A with
    | var i ⇒ s i
```

```
  | A -» B ⇒ (sub s A) -» (sub s B)
  | A & B ⇒ (sub s A) & (sub s B)
  | A \v/ B ⇒ (sub s A) \v/ (sub s B)
  | tt ⇒ tt
  | ff ⇒ ff
end.

Fixpoint ssub (s : nat → form) (G: context) : context :=
  match G with
    | nil ⇒ nil
    | A :: G' ⇒ (sub s A) :: (ssub s G')
  end.
```

Typical substitutions arise inductively from a base substitution replacing a single chosen variable by a formula and leaving all other variables unchanged:

```
Definition s_n (n:nat) (A: form) : (nat → form) :=
  fun m ⇒ match (eq_nat_dec n m) with
              | left _ ⇒ A
              | right _ ⇒ (var m)
            end.

Definition s_p := s_n 0.

Notation "A '{' B '}/' n " := (sub (s_n n B) A) (at level 29).
Notation "A '{' B '}/p'" := (sub (s_p B) A) (at level 29).
Notation "'ssubp' B G" := (ssub (s_p B) G) (at level 29).
```

As only the more narrow substitution s_p for the chosen variable is needed on most occasions, it does pay off to state suitable lemmas in two versions, one for s_n and one for s_n, the former usually with postfix _gen. There are also several notions of freshness, for formulas and for lists, both as a predicate and as a boolean-valued recursive function (all distinguished by corresponding suffixes). Finally, we can finalize the notion of iterated substitution:

```
Fixpoint f_p (A : form) (n : nat) : form :=
  match n with
    | 0 ⇒ var 0
    | S n' ⇒ sub (s_p (f_p A n')) A
  end.
```

Remark 9. The definitions so far were fairly straightforward. There are, however, two basic lemmas that are worth singling out and contrasting.

```
Lemma hil_ded: ∀ G (A B : form), A :: G |- B → G |- A -» B.
Lemma ded_subst_gen : ∀ A G B C n, G |- B «-» C →
                                   G |- (sub (s_n n B) A) «-» (sub (s_n n C) A).
```

As a consequence of the last lemma we have

```
Lemma ded_subst : ∀ A G B C, G |- B «-» C →
                             G |- (sub (s_p B) A) «-» (sub (s_p C) A).
```

For most non-classical logics, it is by no means common to have both of these metatheorems at the same time. In the modal logic setting, for example, a turnstile relation enjoying this deduction theorem (i.e., the one of the form $\Gamma \cup \{A\} \vdash B$ implies $\Gamma \vdash A \to B$) would be the one known as the *local consequence relation*. However, this notion of consequence does not enjoy the other metatheorem. The

global consequence relation, which incorporates the Rule of Necessitation, satisfies in turn the second theorem, but not the first. Similar problems would arise in the realm of substructural logics. Yet both these metatheorems are heavily used in Ruitenburg's work (implicitly) and the present formalization (explicitly), which indicates some reasons why the rarity of periodic sequences property outside the realm of locally finite logics may be not a coincidence. As far as substructural logics are concerned (at least those not enjoying the weakening rule), other examples of incompatible metatheorems heavily used in the present developement would include:

```
Lemma hil_weaken: ∀ G (A B : form), G |- A → B :: G |- A.
Lemma hil_weaken_gen : ∀ G G' A, G |- A → G' ++ G |- A.
Lemma hil_weaken_incl : ∀ G G' A, G |- A → incl G G' → G' |- A.
```

B Decidability of the turnstile relation

Ruitenburg's proof of `Theorem rui_1_4` in several places does case analysis, splitting cases between $\Gamma \vdash A$ and $\Gamma \nvdash A$. While in classical metatheory this does not require further justification, constructively of course it amounts to decidability of the turnstile relation. Simpler syntactic notions such as equality between variables or between formulas are trivially decidable (and, as one may guess, useful in formal development):

```
Lemma dceq_v : ∀ n n0, {var n = var n0} + {var n ≠ var n0}.
Lemma dceq_f: ∀ (A B: form), {A = B} + {A ≠ B}.
```

But a constructive proof of decidability of turnstile relation would require incorporating an actual decidability proof for IPC. As the present development is purely syntactic (and extending it with some standard Kripke completeness proof for IPC would provide little insight into Ruitenburg's proof), the only route worth considering would be the one mentioned above: give up the Hilbert-style approach of Ruitenburg's paper and use a cut-free sequent system together with an actual proof of cut elimination, then use it to prove decidability of turnstile. This is a viable idea for future development, in particular if extended with proof term assignment to extract the computational content of Ruitenburg's result. Still, it seems orthogonal to the actual goal of verifying the original proof. The route taken in the present paper looks as follows:

```
Module DecidEquiv.

Require Import Coq.Logic.Classical_Prop.

Lemma decid_equiv : ∀ G A, (G |- A) ∨ ~(G |- A).
  intros. apply classic.
Qed.

End DecidEquiv.
```

That is, excluded middle was imported only inside a module in order not to contaminate the rest of development. The reader can verify that it is only used in two places in the proof of `Theorem rui_1_4`.

C Proving Ruitenburg's auxiliary lemmas

So far, we were discussing metatheorems basic to the point of being used implicitly in Ruitenburg's paper. In this section, we are finally going to start formalizing the actual development in the paper itself, corresponding Lemma 1.2 and Lemmas 1.6–1.8 in the original paper. Lemma 1.2 is the very last part of `HilbertIPCSetup`:

```
Lemma rui_1_2_i : ∀ A i k, [f_p A i ; sub (s_p tt) A] |- f_p A (i + k).
Lemma rui_1_2_ii : ∀ A i n, [ f_p A i; sub (s_p tt) A ] |-
                                (f_p A (S n) -» f_p A n) -» f_p A n.
```

Lemmas 1.6–1.8 form the entirety of `Ruitenburg1984Aux`:

```
Lemma rui_1_6 : ∀ A m,
                  (∀ i, [f_p A i ; sub (s_p tt) A] |- f_p A m) →
                                [sub (s_p tt) A] |- f_p A m «-» f_p A (m+1).
Lemma rui_1_7_i : ∀ A m n,
                   [ sub (s_p tt) (f_p A (2×m + 1))] |- sub (s_p tt) (f_p A n).
Lemma rui_1_7_ii : ∀ A m n,
                 [ sub (s_p tt) (f_p A (2×m + 2))] |- sub (s_p tt) (f_p A (2 ×n)).
Lemma rui_1_8 : ∀ A m,
                  [sub (s_p tt) A] |- (f_p A m) «-» (f_p A (m + 1)) →
                                [] |- f_p A (m+1) «-» f_p A (m + 3).
```

As one can see, these are rather nontrivial metatheorems about IPC. Still, it was a rather pleasant part of the paper to formalize. While the lemmas in question are non-trivial observations about IPC, the formalization itself posed no significant, Coq-specific problems. The automated proofs follow closely proofs in the paper. I would even claim that the Coq proofs are at times easier to understand than in the original version and clarify some remarks that were not always entirely transparent—e.g., the repeated instruction to "use (iterated) substitution"—but this is ultimately a question of individual preferences.

Remark 10. It is worth noting that unlike the main theorem itself (`Theorem rui_1_9_Ens`) and its key ingredient (`Theorem rui_1_4`), all the lemmas in question would hold for any extension of IPC which meets the restrictions discussed in Remark 9, which as we discussed applies to intuitionistic modalities satisfying the axiom $\mathsf{S}^{\mathsf{int}}_{\Box}$, i.e., Curry-Howard counterparts of applicative functors. In fact, the repository includes a fork `feature/extended_formalisms` with a subfolder `applicative_development` to illustrate that all the results of `HilbertIPCSetup` and `Ruitenburg1984Aux` (i.e., Lemmas 1.2 and 1.6–1.8 discussed in this section) would work for any logic of the form $\mathsf{S}^{\mathsf{int}}_{\Box} \oplus A$, for any given A in the unimodal propositional language with $\Box$. Whether or not there are interesting non-locally-finite logics of this form for which the rest of the development can be carried, i.e., for which llps holds is not clear; cf. Open Problem 1.

D Bounds as Ensembles: proving the main result

`Theorem rui_1_4` relies on a notion of a *bound* of formula A over a context (a set, or in our case a list of formulas). It is a finite set of formulas, each of which is equivalent to a substituted implicational subformula of A (more precise definition below), and the proof of `Theorem rui_1_4` proceeds by induction over its cardinality. Ensemble, an old weapon in Coq's arsenal, seems particularly well-suited for such proofs. The disadvantage, of course, is that being a predicate, i.e., a `Prop`-valued function, it does not allow the use of program extraction or Coq's programming capabilities; we will see a solution in § E. I believe, however, that it was beneficial to keep the logical and computational uses of bounds apart. If the code is refined in future, it could be beneficial to explicitly use the notion of *reflection* here.

After giving the obvious definition of Subformulas, we identify their special subclass BoundSubformulas: those which are either implicational subformulas or propositional variables. Then one can proceed to defining what bounds are:

```
Definition Bound (G: context) (A : form) (b : Ensemble form) :=
  ∀ C: form, In (BoundSubformulas A) C →
                  ∃ B , In b B ∧ G |- sub (s_p tt) C «-» B.
```

There is a minor discrepancy between the Bound used here and the *bounds* as used by Ruitenburg: the latter does not impose that b already contains formulas (equivalent to) BoundSubformulas of A elements p substituted with tt; in Ruitenburg's version, the part following the turnstile would be sub (s_p tt) C «-» (s_p tt) B. Of course, if b is a bound in his sense, then $b\{tt/p\}$ is a bound both in the present sense and in his sense; hence, the present definition is narrower, but the difference does not matter from a practical point of view. On the other hand, Ruitenburg's definition insist that a bound contains tt, whereas it may well happen that a Bound contains nothing equivalent to it; consider, for example, a bound of q -» r over []. The difference can be handled easily, as we will see in the statement of Theorem rui_1_4 below; moreover, Ruitenburg's convention will be followed when bounds are treated as lists rather than Ensembles (see § E).

```
Definition ExactBound (G: context) (A : form) (b : Ensemble form) :=
  (∀ B: form, In b B →
                  ∃ C , In (BoundSubformulas A) C ∧
                        G |- sub (s_p tt) C «-» B) ∧
  Bound G A b
```

even though the Bound captures only inclusion in one direction. The rest of `BoundsSubformulas.v` is devoted to various auxiliary lemmas about notions involved.

We can move on now to the contents of `Ruitenburg1984KeyTheorem`, which contains the actual proof of the central syntactic result in the paper. The first larger theorem proved in this file, before actual Theorem rui_1_4, deals with a remark in the base case of the proof, referring to the Rieger-Nishimura theorem mentioned in Section 1. Recall that this very theorem pinpoints why IPC does not have local finiteness by describing the infinite poset of all IPC-formulas in one free variable, quotiented by provable equivalence and ordered by provable implication. While the Rieger-Nishimura lattice has been thoroughly understood and reconstructed on several occasions, using differing techniques[8], it could be of interest to formalize it fully. Fortunately, as it turns out, we only need a corollary of this result:

```
Lemma Rieger_Nishimura_corollary :∀ G B v,
            fresh_l_p v (B :: G) →
            ( ∀ C : form, (BoundSubformulas B) C → G |- sub (s_p tt) C ) → ∀ C, (Subformulas B)
C → G |- (sub (s_p (var v)) C) -» ff ∨
                                  G |- sub (s_p (var v)) C «-» (var v) ∨
                                  G |- sub (s_p (var v)) C.
```

With this last issue out of the way, one can finally state and prove

```
Theorem rui_1_4: ∀ n G A B,
  Included (BoundSubformulas B) (BoundSubformulas A) →
  ∀ i v,
  let G' := (f_p A i) :: sub (s_p tt) A :: G in
  (∃ b, let b' := (App tt b) in Bound G' A b' ∧ cardinal b n) →
  fresh_l_p v (p::B::A::G) →
      ((f_p A (2×n)) -» (var v)) :: G' |-
        (sub (s_p (var v)) B «-» sub (s_p tt) B) & (sub (s_p tt) B -» (var v)) ∨
```

[8] In fact, its very name refers to the rediscovery of Rieger's result [38] by Nishimura [32]. More information and further references can be found in standard monographs (see, e.g., [6, Ch. 7]).

```
((f_p A (2×n)) -» (var v)) :: G' |- sub (s_p (var v)) B «-» (var v) ∨
((f_p A (2×n)) -» (var v)) :: G' |- sub (s_p (var v)) B.
```

A comparison reveals inessential differences with the statement of this theorem in the original paper. Instead of assuming that BoundSubformulas of B are contained in those of A, a premise of the result stated by Ruitenburg is the existence of a bound (of size n) of $A \,\&\, B$ over G'. Then, at the very beginning of the proof, an observation is made that one can assume that *B is a subformula of A by replacing A by the equivalent formula* $A \,\&\, (B \setminus \mathrm{v}/\,\mathrm{tt})$. The assumption made here, i.e., that (BoundSubformulas B) are included in (BoundSubformulas A) seems an optimal solution. Another, very minor difference is that the supposed bound b is immediately tweaked to b' containing tt. This has to do with the difference in the definition of our Bound mentioned above.

The proof is by induction on n, i.e., the cardinality of a bound for A (and, consequently, also for B). The inductive step, furthermore, involves an induction over B. This in turn involves a consideration of numerous cases and subcases of these subcases, almost each of which involves some actual propositional reasoning in IPC; it is rather unlike, e.g., many cut-elimination proofs, where most cases would be more or less automatic and only some critical ones would require attention. For this reason, I believe this is an example of a situation where one should *not* strive for very strong tactics, crushing automatically most of these cases with their subcases behind the scenes; a separate question is to what extent this would have been even doable. While the present formalization involves a fair number of tailored tactics to deal with most mundane tasks (and can be conceivably still automated further), an actual insight into the proof can be only obtained by following simultaneously Ruitenburg's development on paper and walking through the present formalization in an IDE. The original proof of Theorem 1.4, despite its compactness, gives hints for almost all cases worthy of attention; still, what was more or less one page on paper turns into more than 700 lines of code, and this despite all the spadework done before.

Once this central technical piece is proved, it yields corollaries (already in the file `Ruitenburg1984Main.v`):

```
Corollary rui_1_4': ∀ A b n, (Bound [] A b) → cardinal b n →
  ∀ i v, let v' := S v in fresh_f_p v' A →
     let G' := [f_p A i ; sub (s_p tt) A] in
     let G'' := (f_p A (2×n) -» (var v')) :: G' in
       G'' |- (sub (s_p (var v')) A «-» sub (s_p tt) A) &
              (sub (s_p tt) A -» (var v')) ∨
       G'' |- sub (s_p (var v')) A «-» (var v') ∨
       G'' |- sub (s_p (var v')) A.
```

In other words, this corollary is simply stating what rui_1_4 means for a single formula A rather than for a pair of formulas A, B (note one needs to ensure here that the fresh variable in question is not p itself; it might happen that A does not contain it).

```
Corollary rui_1_5: ∀ A b i n, (Bound [] A b) → cardinal b n →
     let m:= (2 × n + 1) in
       [sub (s_p tt) A; f_p A i] |- f_p A m.
```

While it may seem surprising, proving this much simpler-looking corollary is the only goal of the intimidating `Theorem` rui_1_4, and the proof of this corollary is not even using the theorem itself, but rather `Corollary` rui_1_4' above. Note also that in the original paper, the statement of the theorem does not involve the connection between m and a bound, although it is clear in the proof.

This corollary is now combined with Lemmas 1.6–1.8 discussed in § C to yield the main result:

```
Theorem rui_1_9_Ens:
  ∀ A b m, (Bound [] A b) →
```

```
cardinal b m →
[] |- f_p A (2 × m + 2) «-» f_p A (2 × m + 4).
```

This finally explains the name Bound: the size of such a bound for A determines (linearly) how many iterated substitutions (at worst) it takes before it enters the cycle. Clearly, there is always a bound linear in the size of A: simply take all the implicational subformulas (i.e., BoundSubformulas) of A and substitute tt for p removing possible duplicates. Is it the best that one can do? And is it important to care about such minor adjustments? This is the last part of our considerations.

E Bounds as lists: computations and program extraction

As discussed in § D, treatment of bounds as Ensembles, very convenient for the proof of main result, is not very useful computationally, starting from the fact that `Prop` gets erased during program extraction. For this reason, one can consider a natural reformulation of the notion of a bound in terms of lists. These, in turn, would be awkward in proofs discussed in § D, but are bread-and-butter from a functional programming point of view. All that is needed to verify the programs obtained in this way is to provide bridge theorems between Ensemble- and list-counterparts of the same notion, and this is much easier than developing everything from scratch in terms of lists. These are the contents of `BoundsLists.v`. As suggested above, a more structured approach would probably involve systematic use of reflection.

A suitable counterpart of Bound is

```
Definition bound (b: list form) (A : form) (G : context) :=
  Forall (fun C ⇒ Exists (fun B ⇒ G |- B «-» C ∧ [] |- B «-» sub (s_p tt) B) b) (mb_red A).
```

Using a natural function converting contexts (i.e., lists) to Ensembles, one can easily show

```
Lemma bound_is_Bound : ∀ b A G, bound b A G → Bound G A (context_to_set b).
```

And the corresponding version of the main theorem is

```
Theorem rui_1_9_list: ∀ A b, (bound b A []) →
  ∃ m, m ≤ length b ∧ [] |- f_p A (2 × m + 2) «-» f_p A (2 × m + 4).
```

The most naïve way to produce a bound for A over [] (and hence over any G, see bound_for_bound_upward) is by

```
Fixpoint mb_red (A : form) : list form :=
match A with
    | var i ⇒ [sub (s_p tt) (var i) ; tt]
    | B -» C ⇒ sub (s_p tt) (B -» C) :: (mb_red B ++ mb_red C)
    | B & C ⇒ (mb_red B ++ mb_red C)
    | B \v/ C ⇒ (mb_red B ++ mb_red C)
    | tt ⇒ [tt]
    | ff ⇒ [tt]
end.
```

Note that this time we are following Ruitenburg's convention and explicitly including tt.

However, this is obviously suboptimal. To begin with, the output of mb_red is almost guaranteed to contain duplicates, but this is easy to deal with (using dup_rem). More importantly, such a list is also likely to contain equivalent formulas, which are also, given the definition, redundant. It gets particularly dramatic when the formula in question contains no other variables than p; cf. Proposition 2.3 and Theorem 2.4 in Ruitenburg's paper [39]; within the one-variable fragment, IPC has strictly globally periodic

sequences, just like the classical logic. But improvements are possible also when a formula contains more than one variable. The present development is restricted to a simple optimizer `t_optimize`, which is essentially removing redundant occurrences of `tt`. The function `optimized_bound` combines duplicate removal from `dup_rem` with iterating `t_optimize` as many times as the formula depth of A requires. Still further improvements are certainly possible. Given that IPC is decidable, the ultimate option would be the one discussed in § B: integrating a decision procedure for IPC and testing pairwise elements of a given `bound`, removing the elements equivalent to those found earlier in the list.

At any rate, even the present development can be used for actual computation, either using Coq's programming language capabilities or, if one prefers, program extraction. Combo functions available at the end of `Ruitenburg1984Main.v`, i.e., *optimized_cycle* or *cycle_formula_length* produce the value after which the sequence is going to enter a cycle for a given $A(p)$ and the size of corresponding $A^{2m+2}(p)$. They can be directly extracted to any typical target language such as Haskell or OCaml. In fact, Coq's `Compute` itself does a satisfying job in computing these values. However, even simple experiments indicate one should be rather careful as a blow-up can occur very quickly. The fact that m itself is linear in the size of A surely enough does not mean that $A^{2m+2}(p)$ is and rather simple examples can make it painfully clear. Consider, e.g., a formula from `BoundsLists.v`:

```
Definition exform1 := (q -» p -»r) & ((p -» r) -» p \v/ r).
```

for which the length of the value of `optimized_bound` is just $m := 4$. But now set $A(p)$ to be `exform1` and, as an exercise, estimate the size of $A^{2m+2}(p)$.[9]

[9]This is, in fact, a mistake I made myself while experimenting with the code. I ran Coq on this input without a prior pen-and-paper or simply commonsensical estimate of the size of output. To Coq's credit, it was able to return with the actual formula after several minutes. A curious reader can see it in `f_p_exform1_10_output.txt`.