Journal of Automated Reasoning
https://doi.org/10.1007/s10817-020-09555-y

# Constructive Decision via Redundancy-Free Proof-Search

**Dominique Larchey-Wendling**[1]

Received: 17 April 2020 / Accepted: 20 April 2020


**Abstract**
We give a constructive account of Kripke–Curry's method which was used to establish the decidability of implicational relevance logic ($\mathbf{R}_{\to}$). To sustain our approach, we mechanize this method in axiom-free Coq, abstracting away from the specific features of $\mathbf{R}_{\to}$ to keep only the essential ingredients of the technique. In particular we show how to replace Kripke/Dickson's lemma by a constructive form of Ramsey's theorem based on the notion of almost full relation. We also explain how to replace König's lemma with an inductive form of Brouwer's Fan theorem. We instantiate our abstract proof to get a constructive decision procedure for $\mathbf{R}_{\to}$ and discuss potential applications to other logical decidability problems.

**Keywords** Constructive decidability · Relevance logic · Sequent calculi · Contraction rule · Redundancy-free search · Almost full relations · Mechanization in Coq

**Mathematics Subject Classification** 03F03 · 03B35 · 03B47

## 1 Introduction

In this paper, we give a fully constructive/inductive account of Kripke's decidability proof of implicational relevance logic $\mathbf{R}_{\to}$, fulfilling the program outlined by Riche [17]. The result is known as Kripke's but it crucially relies on Curry's lemma [18] which states that if a sequent $S_2$ is redundant over a sequent $S_1$ and $S_2$ has a proof, then $S_1$ has a shorter proof. Our account of Kripke–Curry's method is backed by an axiom-free mechanized proof of the result in the Coq proof assistant.[1] However, their method and our constructivized implementation is in no way limited to that particular logic. As explained in [19], "Kripke's procedure for deciding $\mathbf{R}_{\to}$ can be seen as a precursor for many later algorithms that rely on the existence of a well quasi ordering (WQO)." From a logical perspective, Kripke–Curry's method has been adapted to *implicational ticket entailment* [2] and *the multiplicative and exponential*

[1] https://coq.inria.fr.

Work partially supported by the joint project ANR-FWF TICAMORE (No. ANR-16-CE91-0002).

✉ Dominique Larchey-Wendling
dominique.larchey-wendling@loria.fr

[1] Université de Lorraine, CNRS, LORIA, Campus Scientifique, BP 239, 54 506 Vandœuvre-lès-Nancy, France

*fragment of linear logic* [1]. However, both of these recent papers are now contested inside the community because of flaws in the arguments; see [9, footnote 1], [20, footnote 4], [6, pp 360-362] and [22]. This illustrates that the beauty of Kripke–Curry's method should not hide its subtlety and justifies all the more the need to machine-check such proofs.

From a complexity perspective, Schmitz [19] recently gave a 2- ExpTime complexity characterization of the entailment problem for $\mathbf{R}_{\to}$, implying a decision procedure. However, both theoretically and practically, the existence of a complexity characterization does not automatically imply a constructive proof of decidability. Indeed, the decision procedure itself or its termination proof might involve non-constructive arguments. In the case of $\mathbf{R}_{\to}$, the result of [19] "relies crucially" on the 2- ExpTime-completeness of the coverability problem in branching vector addition systems (BVASS) [7]. Checking the constructive acceptability of such chains of results implies checking that property for every link in the chain, an intimidating task, all the more problematic when considering mechanization.[2]

Our interest in the entailment problem for $\mathbf{R}_{\to}$ lies in the inherent simplicity and generality of Kripke–Curry's argumentation. It is centered around the notion of redundancy avoidance. But compared to e.g. intuitionistic logic (**IL**), the case of $\mathbf{R}_{\to}$ is specific because *redundancies are not limited to repetitions*: the redundancy relation is not the identity. The case of repetition is not so interesting: Curry's lemma is trivial for repetition; and the sub-formula property and the pigeon hole principle ensure that Gentzen's system **LJ** has a finite search space.

In the case of $\mathbf{R}_{\to}$, the sequent $S_2$ is redundant over $S_1$ if they are cognate[3] and $S_1$ is included into $S_2$ for multiset inclusion [18]. In [17], Dickson's lemma is identified as the main difficulty for transforming Kripke–Curry's method into a constructive proof. Dickson's lemma[4] is a consequence of Ramsey's theorem which, stated positively, can be viewed as the following result [24]: the intersection of two WQOs is a WQO. The closure of the class of WQOs under direct products follows trivially and so does Dickson's lemma. We think that the use of König's lemma in Kripke's proof is also a potential difficulty w.r.t. constructivity. Admittedly, there are many variants of this lemma and indeed, we will use one which is well suited in a constructive argumentation.

Let us now present the content of this paper. In Sect. 2, we propose an overview of Kripke–Curry's argumentation focusing on the two issues of Dickson's lemma and König's lemma. To constructivize that proof, we approached the problem posed by Dickson's lemma by using Coquand's [3] direct intuitionistic proof of Ramsey's theorem through an intuitionistic formulation of WQOs as *almost full relations* (AF) [25]. Starting in Sect. 3, we switch to the language of inductive type theory. We recall some basic notions and notations, introduce formal (finite and indexed) trees and their branches, informative finiteness predicates, and then the AF and bar inductive predicates which we prove equivalent. We recall the constructive Ramsey's and Fan theorems and we conclude with a constructive version of König's lemma tailored for AF (redundancy) relations. We explain how to use it for constructive and terminating search in a finitely branching and irredundant search space.

In Sect. 4, we give a detailed formal account of what could be called the central ingredients of Kripke–Curry's proof by outlining the essential steps of our constructive mechanization in Coq. Our Theorem `proof_decider` of Fig. 3 on page 17 abstracts away from the particular case of $\mathbf{R}_{\to}$ by isolating the essential ingredient: an almost full redundancy relation which satisfies Curry's lemma.

[2] As for coverability in BVASS, it seems that the arguments developed in [7] cannot easily be converted to constructive ones (private communication with S. Demri).

[3] i.e. they are identical when ignoring repetitions and permutations.

[4] Dickson's lemma states that under product order, $\mathbb{N}^k$ is a WQO for any $k \in \mathbb{N}$.

In Sect. 5, we instantiate the `proof_decider` into a constructive decision procedure for the specific case of $\mathbf{R}_{\to}$. For this, we describe how we implement the equivalence between the Hilbert's style proof system for $\mathbf{R}_{\to}$ and three sequent proof systems for $\mathbf{R}_{\to}$: the Gentzen system $\mathbf{LR1}_{\to}$ with an explicit contraction rule, both with a cut rule and cut-free, and the contraction absorbing and cut-free $\mathbf{LR2}_{\to}$ that is used for the decision procedure. We describe the implementation of soundness and completeness results, cut-admissibility for $\mathbf{LR1}_{\to}$ using a semantic argument, and Curry's and Kripke's lemmas for $\mathbf{LR2}_{\to}$. We conclude with the constructive decider for $\mathbf{R}_{\to}$.

The technical aspects of our proofs are sustained by a Coq v8.8+ mechanization which is available under a Free Software license at: https://github.com/DmxLarchey/Relevant-decidability. The size of this development is significant—around 15 000 lines of code,—but most of the code is devoted to libraries and the implementations of the proof systems $\mathbf{R}_{\to}$, $\mathbf{LR1}_{\to}$ and $\mathbf{LR2}_{\to}$ and the links between them: soundness/completeness results, cut-elimination, sub-formula property, finitely branching proof-search, etc. The case of implicational intuitionistic logic $\mathbf{J}_{\to}$ is treated as well in this mechanization but we do not discuss this less interesting example here. The core of our constructivization of Kripke–Curry's proof can be found in the file proof.v and is only around 800 lines long (including comments).

This paper is a revised and completed version of the conference paper [13]. In comparison, we modify Sect. 2 in marginal ways only. In Sect. 3 however, we add a short introduction to the type-theoretical tools that we use, more complete explanations with some supplementary intermediate results and outlines of proofs about almost full relations and bar inductive predicates. Moreover, we give a high-level overview on how the constructive König's lemma can be used to bound a finitely branching and redundancy-free search space. We enrich Sect. 4 with short descriptions of the essential proof steps for the main results. We contribute a completely new Sect. 5 where we give some details above how sequent systems and translations between them are implemented in Coq. These results correspond to most of the source code as measured in lines of code. Generally we also provide more frequent pointers to the Coq source, the pdf version of this completed paper being hyperlinked with the above GitHub repository.

In preparation for this paper, we have updated the source code corresponding to the conference paper, but only in marginal ways. However, to maintain synchronization between paper revisions and source code revisions, we have *tagged* the above Git repository with tag v1.0 for the older conference source tree and with tag v2.0 for the later source tree matching the present paper.

## 2 Constructive Issues in Kripke's Decidability Proof

In this section, we recall the main aspects of Kripke's decidability proof for the *implicational fragment of relevance logic* $\mathbf{R}_{\to}$, described with Hilbert style proof rules in Fig. 1. We sum up the description of [18] while focusing on the aspects of the arguments that were challenging from a constructive perspective. Among the many research directions later suggested by Riche [17] for solving the missing link—a constructive proof of IDP or of Dickson's lemma,—the use of Coquand's approach to *Bar induction* [4] turned out as a solution.

Notice that in the notation $\mathbf{R}_{\to}$, the symbol $\to$ represents the logic-level implication to stay coherent with [17–19]. But in this paper, we rather use $\supset$ to denote object-level logical

$$\vdash_{\mathrm{h}} A \supset A \qquad \vdash_{\mathrm{h}} (A \supset B) \supset (C \supset A) \supset (C \supset B) \qquad \frac{\vdash_{\mathrm{h}} A \supset B \quad \vdash_{\mathrm{h}} A}{\vdash_{\mathrm{h}} B}$$
$$\vdash_{\mathrm{h}} (A \supset A \supset B) \supset (A \supset B) \qquad \vdash_{\mathrm{h}} (A \supset B \supset C) \supset (B \supset A \supset C)$$

**Fig. 1** Hilbert's style proof system for implicational relevance logic $\mathbf{R}_{\to}$

implications to avoid conflicting with the Coq notation for meta-level function types $T_1 \to T_2$; see e.g. the below definition of `HR_proof`.[5]

### 2.1 What is a Constructive Proof of Relevant Decidability?

Let us formalize the high-level question that we solve in this paper. Before we give a mechanized constructive proof of decidability for $\mathbf{R}_{\to}$, we need to formally define the language and provability/proofs, at least for $\mathbf{R}_{\to}$. The type $\mathbb{F}$ of formulæ of $\mathbf{R}_{\to}$ is inductively defined by $A, B : \mathbb{F} ::= p \mid A \supset B$ where $p$ ranges over a fixed enumerable type, e.g. a copy of the type of natural numbers $\mathbb{N}$. The type `HR_proof` $A$ of Hilbert-style proofs of $A : \mathbb{F}$ is denoted by $\vdash_{\mathrm{h}} A$ and can straightforwardly be defined in Coq using the (informative) inductive predicate:

```
Inductive HR_proof : 𝔽 → Set :=
 | id   : ∀A,     ⊢h A ⊃ A
 | pfx  : ∀A B C, ⊢h (A ⊃ B) ⊃ (C ⊃ A) ⊃ (C ⊃ B)
 | comm : ∀A B C, ⊢h (A ⊃ B ⊃ C) ⊃ (B ⊃ A ⊃ C)
 | cntr : ∀A B,   ⊢h (A ⊃ A ⊃ B) ⊃ (A ⊃ B)
 | mp   : ∀A B,   ⊢h A ⊃ B → ⊢h A → ⊢h B
where "⊢h A" := (HR_proof A).
```

which reflects the rules of the Hilbert system for $\mathbf{R}_{\to}$ of Fig. 1. A *constructive decidability proof* for $\mathbf{R}_{\to}$ would then be given by a term `HR_decidability` of type:

$$\texttt{HR\_decidability} : \forall A : \mathbb{F},\ \{\texttt{inhabited}(\vdash_{\mathrm{h}} A)\} + \{\neg\texttt{inhabited}(\vdash_{\mathrm{h}} A)\}$$

i.e. a total computable function which maps every formula $A$ to a `boolean` value which if `true`, ensures that there is a proof of $A$, and if `false` ensures that there is no proof of $A$. A *constructive decider* is a stronger result of type:

$$\texttt{HR\_decider} : \forall A : \mathbb{F},\ (\vdash_{\mathrm{h}} A) + (\vdash_{\mathrm{h}} A \to \texttt{False})$$

that is a total computable function that maps every formula $A$ to either a proof of $A$ or else ensures that no such proof can exist. This absence of a proof for $A$ is witnessed by a function mapping proofs of $A$ to (proofs of) an absurdity. In other words, a constructive decider is an (always terminating) constructive proof-search algorithm. Obviously, adding axioms to Coq might hinder the computability of its terms (ensured by the normalization property of Coq). Hence, we allow no axiom and we aim at defining `HR_decidability` or `HR_decider` in axiom-free Coq.

### 2.2 Sequent Calculi for $\mathbf{R}_{\to}$

Hilbert's style $\mathbf{R}_{\to}$ formulation is (unsurprisingly) not really suited to designing decision procedures based on proof-search. A standard approach is to convert Hilbert's systems into

[5] Moreover, in type theory, the function type subsumes logical implication via the BHK interpretation where proofs of $A \to B$ are viewed as functions mapping proofs of $A$ to proofs of $B$.

$$\frac{}{A \vdash A}\ \langle \mathrm{AX} \rangle \qquad \frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B}\ \langle {}^{\star}\mathrm{W} \rangle \qquad \frac{\Gamma \vdash A \quad A, \Delta \vdash B}{\Gamma, \Delta \vdash B}\ \langle \mathrm{CUT} \rangle$$

$$\frac{\Gamma \vdash A \quad B, \Delta \vdash C}{\Gamma, \Delta, A \supset B \vdash C}\ \langle {}^{\star}{\supset} \rangle \qquad \frac{A, \Gamma \vdash B}{\Gamma \vdash A \supset B}\ \langle {\supset}^{\star} \rangle$$

**Fig. 2** The $\mathbf{LR1}_{\to}$ sequent calculus rules for implicational relevance logic $\mathbf{R}_{\to}$

*sequent rules* such as those of $\mathbf{LR1}_{\to}$ in Fig. 2 (see also [18]). In this particular system, a sequent $\Gamma \vdash A$ is composed of a multiset $\Gamma$ of formulæ on the left of the $\vdash$ symbol and exactly one formula $A$ on the right of the $\vdash$ symbol. There are three structural rules: $\langle \mathrm{AX} \rangle$, $\langle {}^{\star}\mathrm{W} \rangle$ and $\langle \mathrm{CUT} \rangle$, and two logical rules: $\langle {}^{\star}{\supset} \rangle$ and $\langle {\supset}^{\star} \rangle$. The soundness/completeness of this conversion to sequent calculus is ensured by the following result: a formula $A$ has a Hilbert proof $\vdash_{\mathrm{h}} A$ if and only if the sequent $\emptyset \vdash A$ has a proof in $\mathbf{LR1}_{\to}$ (with $\emptyset$ as the empty multiset); see Sect. 5.2 or file relevant_equiv.v for the mechanized proof.

Although designed for proof-search, the sequent system $\mathbf{LR1}_{\to}$ still suffers two major problems when considering fully automated procedures: one is the $\langle \mathrm{CUT} \rangle$ rule and the other is the more problematic contraction rule $\langle {}^{\star}\mathrm{W} \rangle$. *Cut-elimination* is one of the central questions of proof-theory, partly because $\langle \mathrm{CUT} \rangle$ makes proof-search infinitely branching. Fortunately, the $\langle \mathrm{CUT} \rangle$ rule is admissible in $\mathbf{LR1}_{\to}$ so we can safely remove that rule from $\mathbf{LR1}_{\to}$; see Sect. 5.2 or file sem_cut_adm.v.

On the other hand $\langle {}^{\star}\mathrm{W} \rangle$ needs to be handled much more carefully. The trick of Curry, which is well described in [18] is to absorb several instances of $\langle {}^{\star}\mathrm{W} \rangle$ in the rule $\langle {}^{\star}{\supset} \rangle$ but in a tightly controlled way.[6] By replacing both rules $\langle {}^{\star}\mathrm{W} \rangle$ and $\langle {}^{\star}{\supset} \rangle$ with the single rule $\langle {}^{\star}{\supset_2} \rangle$

$$\frac{\Gamma \vdash A \quad B, \Delta \vdash C}{\Theta, A \supset B \vdash C}\ [{}^{\star}{\supset_2}] \qquad \text{when } \mathtt{LR2c}(A \supset B, \Gamma, \Delta, \Theta)$$

he obtains the system $\mathbf{LR2}_{\to}$ composed of the three rules $\langle \mathrm{AX} \rangle$, $\langle {\supset}^{\star} \rangle$ and $\langle {}^{\star}{\supset_2} \rangle$. The side condition $\mathtt{LR2c}(A \supset B, \Gamma, \Delta, \Theta)$ is a bit complicated to express formally so we will informally sum up its central idea: while applying $\langle {}^{\star}{\supset_2} \rangle$ top-down, some controlled/bounded form of contraction is allowed on every formula: the principal formula $A \supset B$ can be contracted at most twice while side formulæ in $\Gamma$, $\Delta$ can be contracted at most once. See the definition of $\mathtt{LR2c}$ in Sect. 5.2 for a working and precise characterization.

### 2.3 Irredundant Proofs in $\mathbf{LR2}_{\to}$

Before using $\mathbf{LR2}_{\to}$ for deciding $\mathbf{R}_{\to}$, $\mathbf{LR2}_{\to}$ must of course be proved *equivalent* to $\mathbf{LR1}_{\to}$ and this is not a trivial task; Sect. 5 is specifically devoted to those technical details. The cornerstone of the equivalence between $\mathbf{LR2}_{\to}$ and $\mathbf{LR1}_{\to}$ lies in a critical property of $\mathbf{LR2}_{\to}$ called *Curry's lemma*. It ensures both:

- the *admissibility of the contraction rule* $\langle {}^{\star}\mathrm{W} \rangle$ *in* $\mathbf{LR2}_{\to}$;
- the *completeness of irredundant proof-search* in $\mathbf{LR2}_{\to}$.

We say that a sequent $\Gamma \vdash A$ *is redundant over* a sequent $\Delta \vdash B$ and we denote $\Delta \vdash B \prec_{\mathrm{r}} \Gamma \vdash A$ if $\Delta \vdash B$ can be obtained from $\Gamma \vdash A$ by repeated top-down applications of the contraction rule $\langle {}^{\star}\mathrm{W} \rangle$. It is also convenient to characterize redundancy using the *number of*

[6] Unrestricted contraction would generate *infinitely branching* proof-search.

*occurrences* $|\Gamma|_\alpha$ of the formula $\alpha$ in the multiset $\Gamma$:

$$\Delta \vdash B \prec_{\mathrm{r}} \Gamma \vdash A \iff A = B \wedge \forall \alpha, |\Delta|_\alpha \prec_{\mathrm{r}}^{\mathbb{N}} |\Gamma|_\alpha \qquad (\prec_{\mathrm{r}})$$

where the binary relation $n \prec_{\mathrm{r}}^{\mathbb{N}} m$ on $\mathbb{N}$ is defined by

$$n \prec_{\mathrm{r}}^{\mathbb{N}} m \iff n \leq m \wedge (n = 0 \Leftrightarrow m = 0) \qquad (\prec_{\mathrm{r}}^{\mathbb{N}})$$

Now we can formulate Curry's lemma which (in modern terms) states that the contraction rule $\langle \star \mathrm{W} \rangle$ is *height preserving admissible* in $\mathbf{LR2}_{\to}$.

**Lemma 1** (Curry [5], 1950) *Consider two sequents such that* $\Gamma \vdash A$ *is redundant over* $\Delta \vdash B$*, i.e.* $\Delta \vdash B \prec_r \Gamma \vdash A$*. Then any* $\mathbf{LR2}_{\to}$*-proof of* $\Gamma \vdash A$ *can be contracted into an* $\mathbf{LR2}_{\to}$*-proof of* $\Delta \vdash B$*, that is, a proof of lesser height.*

The Coq proof term is `LR2_Curry` and it is presented in Sect. 5.3 and implemented in file relevant_LR2.v. Regular admissibility of contraction follows trivially from Curry's lemma and hence the completeness of $\mathbf{LR2}_{\to}$ w.r.t. $\langle \mathrm{CUT} \rangle$-free $\mathbf{LR1}_{\to}$. Another critical consequence of Curry's lemma is related to irredundant proofs.

**Definition 1** *(Irredundant proof)* A proof is *redundant* if there is a redundant pair in one of its branches, i.e. $\Delta \vdash B \prec_{\mathrm{r}} \Gamma \vdash A$ where $\Gamma \vdash A$ occurs in the sub-proof of $\Delta \vdash B$. A proof is *irredundant* if none of its branches contain a redundant pair.

By Curry's lemma, any sequent provable in $\mathbf{LR2}_{\to}$ has an irredundant proof in $\mathbf{LR2}_{\to}$. The argument is not completely trivial and involves the notion of *everywhere minimal proof* (see Sect. 4). As a consequence, while searching for proofs in $\mathbf{LR2}_{\to}$ one can safely mark redundancies as dead ends: avoiding redundancies is a complete strategy.

### 2.4 Kripke's Decidability Proof

Building on Curry's lemma, the key insight of Kripke's proof of decidability is the following result. As explained in [8,17], it was discovered many times in different fields of mathematics, as e.g. Hilbert's finite basis theorem, the infinite division principle (IDP by Meyer [14]), Dickson's lemma, etc. We express Kripke's lemma with a concept that was not clearly spotted at that time but was popularized later on, that of well quasi order.

**Definition 2** *(Well Quasi Order)* A binary relation $\leq$ over a set $X$ is a *well quasi order (WQO)* if it is reflexive, transitive and any infinite sequence $x : \mathbb{N} \to X$ contains a *good pair* $(i, j)$, which means both $i < j$ and $x_i \leq x_j$.

**Lemma 2** (Kripke [12], 1959) *Given a finite set of formulæ* $\mathcal{S}$*, the redundancy relation* $\prec_{\mathrm{r}}$ *is a WQO when it is restricted to sequents composed exclusively of formulæ in* $\mathcal{S}$*.*

***Proof*** By Ramsey's theorem, the product (or intersection) of two WQOs is WQO.[7] Hence, the relation $\prec_{\mathrm{r}}^{\mathbb{N}}$ over $\mathbb{N}$ is a WQO as the intersection of two WQOs; see Equation $(\prec_{\mathrm{r}}^{\mathbb{N}})$. By finiteness of $\mathcal{S}$, the identity relation $=_{\mathcal{S}}$ on $\mathcal{S}$ is also a WQO (this is an instance of *the pigeon*

[7] This result is known as Dickson's lemma when restricted to $\mathbb{N}^k$ with the point-wise product order. The inclusion relation between multisets built from the finite set $\mathcal{S}$ is a particular case of the product order $\mathbb{N}^k$ where $k$ is the cardinal of $\mathcal{S}$.

*hole principle*). Denoting $\prec_{\mathrm{r}}^{\mathcal{S}}$ for the restriction of $\prec_{\mathrm{r}}$ to the sequents composed of formulæ in the finite set $\mathcal{S}$, we can derive the equivalence

$$\Delta \vdash B \prec_{\mathrm{r}}^{\mathcal{S}} \Gamma \vdash A \iff A =_{\mathcal{S}} B \wedge \bigwedge_{\alpha \in \mathcal{S}} |\Delta|_{\alpha} \prec_{\mathrm{r}}^{\mathbb{N}} |\Gamma|_{\alpha}$$

hence $\prec_{\mathrm{r}}^{\mathcal{S}}$ is a WQO as a finite intersection of WQOs. □

Kripke's argument of decidability for entailment in the sequent calculus $\mathbf{LR2}_{\rightarrow}$ (and hence $\mathbf{R}_{\rightarrow}$) can be summarized in the following steps:

- consider a start sequent $\Gamma \vdash A$ and let $\mathcal{S}$ be its finite set of sub-formulæ;
- launch backward proof-search for irredundant proofs of $\Gamma \vdash A$ in $\mathbf{LR2}_{\rightarrow}$, i.e. search stops when no rule is applicable or at a redundancy. We denote by $\mathcal{T}$ the corresponding (potentially infinite) proof-search tree;
- by the sub-formula property, no formula outside of $\mathcal{S}$ can occur in $\mathcal{T}$;
- $\mathcal{T}$ is finitely branching (critically relies on the side condition of rule $\langle^{\star}\supset_2\rangle$);
- if $\mathcal{T}$ had an infinite branch, it would contain a redundancy (Kripke's lemma);
- hence by König's lemma, the proof-search tree $\mathcal{T}$ is finite.

In [17], Riche focuses on Kripke/Dickson's lemma as the main difficulty to get an argumentation that could be accepted from a constructive point of view. We think that König's lemma is also a potentially non-constructive result [21], depending on its precise formulation. In Sect. 4, we explain how to overcome these two difficulties and transform this method into a generic decider by an axiom-free constructive proof of decidability, that we later instantiate on $\mathbf{LR2}_{\rightarrow}$ in Sect. 5 to get a constructive decider for $\mathbf{R}_{\rightarrow}$ in Coq.

## 3 Inductive Well Quasi Orders in Type Theory

From now on, we switch to the language of Inductive Type Theory instead of the usual set theoretical language. After a summary of basic type theoretic notations, we define the notions of finiteness, the type of finite trees and their branches, and then of almost full relations, a notion which constructively characterizes WQOs. We finish with a proof of a constructive form of König's lemma.

### 3.1 Some Basic Notions in Type Theory

We recall the basic notions of propositions, dependent sub-types, Peano natural numbers and polymorphic lists that are the building blocks over which we develop our theory. They all belong to the Coq standard library.[8] We denote by `Type` the members of the family of type universes (hiding their indices). The type `Prop` of propositions will be denoted by $\mathbb{P}$ for conciseness and `False` : $\mathbb{P}$ represents the empty proposition which types no proof term and thus implements absurdity. Given a type $X$ : `Type` and a predicate $P : X \to \mathbb{P}$, we denote by $\{x : X \mid P\,x\}$ the sub-type defined by $P$, i.e. the type theoretic dependent sum $\Sigma x : X, P\,x$ composed of pairs $(x, H_x)$ where $x : X$ and $H_x : P\,x$ is a proof that $P$ holds at point $x$.

Peano natural numbers are inductively defined by $n : \mathbb{N} ::= 0 \mid 1 + n$ and by $\mathbb{L}\,X$ the (polymorphic) type of lists over $X$ inductively defined by $l : \mathbb{L}\,X ::= [] \mid x :: l$ where $x : X$. The symbol [] denotes the empty list and we may write $[x_1; \ldots; x_n]$ for the list $x_1 :: \cdots :: x_n :: []$. Given a function $f : X \to Y$, we denote `map` $f$ as the lifting of $f$

[8] The Coq standard library is documented at https://coq.inria.fr/library.

to $\mathbb{L}\,X \to \mathbb{L}\,Y$ and characterized by $\texttt{map}\ f\ [x_1; \ldots; x_n] = [f\ x_1; \ldots; f\ x_n]$. The function $\texttt{rev} : \mathbb{L}\,X \to \mathbb{L}\,X$ implements list reversal and satisfies $\texttt{rev}\,[] = []$, $\texttt{rev}(x :: []) = x :: []$ and $\texttt{rev}(l \mathbin{++} m) = \texttt{rev}\,m \mathbin{++} \texttt{rev}\,l$. We write $x \in_1 l$ as a short infix notation for $\texttt{In}\ (x : X)\ (l : \mathbb{L}\,X) : \mathbb{P}$, the list membership predicate defined by the following fixpoint equations $x \in_1 [] := \texttt{False}$ and $x \in_1 (y :: l) := x = y \vee x \in_1 l$. We also use $\subseteq_1$ as a short notation for list inclusion, i.e. $l \subseteq_1 m := \forall x,\ x \in_1 l \to x \in_1 m$. For any predicate $P : X \to \mathbb{P}$, we denote by $\forall_1 P\ l$ for finite universal quantification over the list $l$, i.e. $\forall_1 P\ l \leftrightarrow (\forall x,\ x \in_1 l \to P\ x)$. Notice that the $\forall_1 P$ predicate (denoted $\texttt{Forall}\ P$ in Coq standard library) is equivalently defined as an inductive predicate in the `List` module of the standard library.

### 3.2 Finiteness, Trees and Branches

The predicate $\texttt{fin}_\texttt{t}\ P$ states that a sub-type $P : X \to \mathbb{P}$ is finite and computable into a list:

$$\texttt{Definition fin}_\texttt{t}\ \{X : \texttt{Type}\}\ (P : X \to \mathbb{P}) := \{l : \mathbb{L}\,X \mid \forall x,\ x \in_1 l \leftrightarrow P\ x\}.$$

Inhabitants of $\texttt{fin}_\texttt{t}\ P$ are dependent pairs $(l, H_l)$ where $H_l$ is a proof that $l$ is indeed a finite extensional description of $P$: membership in $l$ is equivalent to $P$. Notice that the braces around parameter $\{X : \texttt{Type}\}$ specify an *implicit parameter* in the definition of $\texttt{fin}_\texttt{t}$.

We define the type of finitely branching (oriented) trees as generated by a single inductive rule (see file tree.v). A tree is a label in $X$ together with a list of (sub-)trees:

$$\texttt{Inductive}\ \mathbb{T}\,X := \texttt{in\_tree} : X \to \mathbb{L}\,(\mathbb{T}\,X) \to \mathbb{T}\,X.$$

We use $\langle x \,|\, l \rangle$ as a short notation for $\texttt{in\_tree}\ x\ l$. The $\texttt{root} : \mathbb{T}\,X \to X$ of a tree verifies $\texttt{root}\,\langle x \,|\, \_ \rangle = x$, the $\texttt{sons} : \mathbb{T}\,X \to \mathbb{L}\,(\mathbb{T}\,X)$ verifies $\texttt{sons}\,\langle \_ \,|\, l \rangle = l$, and the height $\texttt{ht} : \mathbb{T}\,X \to \mathbb{N}$ of a tree verifies $\texttt{ht}\,\langle \_ \,|\, l \rangle = 1 + \texttt{max}\,(\texttt{map ht}\ l)$.

Branches of trees are represented by specific lists of elements of type $X$. We inductively characterize the `branch` predicate

$$\begin{array}{l}\texttt{Inductive branch} : \mathbb{T}\,X \to \mathbb{L}\,X \to \mathbb{P} := \\ \quad \mid \texttt{in\_tb0} : \forall t,\ \texttt{branch}\ t\ [] \\ \quad \mid \texttt{in\_tb1} : \forall x,\ \texttt{branch}\ \langle x \,|\, [] \rangle\ (x :: []) \\ \quad \mid \texttt{in\_tb2} : \forall b\ x\ l\ t,\ t \in_1 l \to \texttt{branch}\ t\ b \to \texttt{branch}\ \langle x \,|\, l \rangle\ (x :: b).\end{array}$$

such that the lists $b$ which satisfy $\texttt{branch}\ t\ b$ collect all the nodes encountered on paths from the root of $t$ to one of its internal nodes. The empty list [] is among them. Notice that branches are listed from the tree root to its leaves. If one wants branches ordered from leaves to the root, one should further apply the list reversal function `rev`.

### 3.3 Good Lists, Almost Full Relations and Bar Inductive Predicates

In this section we describe an inductive formulation of the notion of WQO. Type theoretically, Definition 2 becomes: a WQO is a reflexive and transitive predicate $\prec_\text{r} : X \to X \to \mathbb{P}$ such that for any $f : \mathbb{N} \to X$, there exists $i, j : \mathbb{N}$ such that $i < j$ and $f_i \prec_\text{r} f_j$ (good pair). We can say that any infinite sequence is bound to be redundant. We recall the inductive characterization of WQO due to Fridlender and Coquand [10] and the *constructive Ramsey theorem* [25], from which we derive a constructive proof of Kripke's lemma. The corresponding Coq code can be found in our library file almost_full.v.

Much like *well founded relations* can be defined inductively by *accessibility predicates* (see module Wf of Coq standard library), WQOs can inductively be defined either by the *almost full inductive predicate* (AF) or by *bar inductive predicates*. Notice that these two equivalent inductive characterizations are constructively stronger than the usual classical definition (like in the case of well-foundedness).

Let us consider a type $X : \mathtt{Type}$ and an abstract (redundancy) relation $R : X \to X \to \mathbb{P}$.[9] We define the $\mathtt{good}\ R : \mathbb{L}\ X \to \mathbb{P}$ predicate that characterizes the (finite) lists which contain a good pair:

$$\mathtt{good}\ R\ [x_{n-1}; \ldots; x_0] \leftrightarrow \exists i\ j,\ i < j < n \wedge x_i\ R\ x_j \qquad (\mathtt{good})$$

Hence the list $[\ldots; b; \ldots; a; \ldots]$ is good when $a\ R\ b$. The list is read from right to left because we represent *the n-prefix of a sequence* $f : \mathbb{N} \to X$ by $[f_{n-1}; \ldots; f_0]$.

**Definition 3** *(Ir/redundant)* Given a relation $R : X \to X \to \mathbb{P}$ called redundancy, a list of values $l : \mathbb{L}\ X$ is *redundant* if $\mathtt{good}\ R\ l$ holds, and is *irredundant* if $\neg(\mathtt{good}\ R\ l)$ holds. We denote the later case by $\mathtt{bad}\ R\ l := \neg(\mathtt{good}\ R\ l)$.

The *lifting of a (binary) relation* $R : X \to X \to \mathbb{P}$ by $x : X$ is denoted $R \uparrow x$ and characterized by:

$$u\ (R \uparrow x)\ v \leftrightarrow u\ R\ v \vee x\ R\ u \qquad \text{for any } u, v : X$$

The disjunct $x\ R\ u$ prohibits any $u$ which is $R$-greater than $x$ to occur in $(R \uparrow x)$-bad sequences. AF relations are defined as those satisfying the $\mathtt{af_t}$ predicate:

$$\begin{array}{l}\mathtt{Inductive\ af_t}\ \{X : \mathtt{Type}\}\ (R : X \to X \to \mathbb{P}) : \mathtt{Type} :=\\ \quad \mid \mathtt{in\_af\_t0} : \big(\forall x\ y,\ x\ R\ y\big) \to \mathtt{af_t}\ R\\ \quad \mid \mathtt{in\_af\_t1} : \big(\forall x,\ \mathtt{af_t}\ (R \uparrow x)\big) \to \mathtt{af_t}\ R.\end{array}$$

Hence any full relation (i.e. $\forall x\ y,\ x\ R\ y$) is AF, and if every lifting of $R$ is AF, then so is $R$. Notice that the predicate $\mathtt{af_t}$ is *informative*: it contains a well-founded tree of liftings until the relation becomes full (see [25]). We will use this information to compute bounds.

It is trivial to show that $\mathtt{af_t}$ is a monotonic predicate, i.e. $R \subseteq S \to \mathtt{af_t}\ R \to \mathtt{af_t}\ S$. Also somewhat easy to show but very useful to transfer almost fullness between relations over different base types, the $\mathtt{af_t}$ predicate is preserved by surjective relational morphisms:

$$\begin{array}{l}\mathtt{Proposition\ af\_t\_relmap}\ X\ Y\ (f : X \to Y \to \mathbb{P})\ R\ S :\\ \big(\forall y,\ \{x \mid f\ x\ y\}\big) \to \big(\forall x x' y y',\ f\ x\ y \to f\ x'\ y' \to x\ R\ x' \to y\ S\ y'\big) \to \mathtt{af_t}\ R \to \mathtt{af_t}\ S.\end{array}$$

***Proof*** By induction on $\mathtt{af_t}\ R$ after having generalized the hypotheses that depend on $S$. □

The fact that $f$ is a *relational* morphism (and not just a *functional* morphism) in this result is especially useful when transporting $\mathtt{af_t}$ to a $\Sigma$-type like $\{x \mid P\ x\}$ because it is generally not possible to define surjective functions to such a type.

Reflexive and transitive relations which satisfy $\mathtt{af_t}$ are WQOs in the classical interpretation. But constructively speaking, they are stronger in the following sense: any sequence $f : \mathbb{N} \to X$ can *effectively* be transformed into an upper-bound $n$ under which there exists a good pair, upper-bound obtained by finite inspection of the prefixes of $f$:

$$\begin{array}{l}\mathtt{Lemma\ af\_t\_inf\_chain}\ (X : \mathtt{Type})\ (R : X \to X \to \mathbb{P}) :\\ \qquad \mathtt{af_t}\ R \to \forall f : \mathbb{N} \to X,\ \{n : \mathbb{N} \mid \exists i\ j,\ i < j < n \wedge f_i\ R\ f_j\}.\end{array}$$

[9] We temporarily use letters like $R$ or $S$ to represent binary redundancy relations because some of the next results involve several of such relations.

***Proof*** By induction on the $\texttt{af}_\texttt{t}\ R$ predicate. □

The constructive Ramsey theorem [25] states that almost full relations are closed under (binary) intersection:

$$\texttt{Theorem af\_t\_inter}\ (X : \texttt{Type})\ (R\ S : X \to X \to \mathbb{P}) : \\ \texttt{af}_\texttt{t}\ R \to \texttt{af}_\texttt{t}\ S \to \texttt{af}_\texttt{t}(R \cap S).$$

and as an immediate consequence, under direct products:

$$\texttt{Theorem af\_t\_prod}\ (X\ Y : \texttt{Type})\ (R : X \to X \to \mathbb{P})\ (S : Y \to Y \to \mathbb{P}) : \\ \texttt{af}_\texttt{t}\ R \to \texttt{af}_\texttt{t}\ S \to \texttt{af}_\texttt{t}\ (R \times S).$$

We do not recall the beautiful proof of this theorem which is implemented in file af_t.v following Coquand's outline [25]. Notice that reflexivity and transitivity of WQOs are completely orthogonal to almost fullness in these results. They play no role in our development.

The $\texttt{af}_\texttt{t}\ R$ predicate characterizing the almost fullness of the (redundancy) relation $R$ can alternatively[10] be defined by bar inductive predicates [10] as $\texttt{bar}_\texttt{t}\ (\texttt{good}\ R)\ []$ with the following inductive definition:

$$\begin{array}{l}\texttt{Inductive bar}_\texttt{t}\ \{X : \texttt{Type}\}\ (P : \mathbb{L}\ X \to \mathbb{P})\ (l : \mathbb{L}\ X) : \texttt{Type} := \\ \quad \mid \texttt{in\_bar\_t0} : P\ l \to \texttt{bar}_\texttt{t}\ P\ l \\ \quad \mid \texttt{in\_bar\_t1} : \left(\forall x, \texttt{bar}_\texttt{t}\ P\ (x :: l)\right) \to \texttt{bar}_\texttt{t}\ P\ l.\end{array}$$

Hence, $\texttt{bar}_\texttt{t}\ P\ l$ means that regardless of the repeated extensions of the list $l$ by adding elements at its head, the predicate $P$ is bound to be reached at some point. In particular, assuming $\texttt{bar}_\texttt{t}\ P\ []$, then, for any given sequence $f : \mathbb{N} \to X$, one can compute $n$ such that $P\ [f_{n-1}; \ldots; f_0]$ holds:

$$\texttt{Theorem bar\_t\_inv}\ X\ (P : \mathbb{L}\ X \to \mathbb{P}) : \ \texttt{bar}_\texttt{t}\ P\ [] \to \forall f, \left\{n \mid P\ [f_{n-1}; \ldots; f_0]\right\}.$$

***Proof*** We generalize to $\texttt{bar}_\texttt{t}\ P\ l \to \forall\ f\ n,\ l = [f_{n-1}; \ldots; f_0] \to \left\{m \mid P\ [f_{m-1}; \ldots; f_0]\right\}$ and prove that statement by induction on $\texttt{bar}_\texttt{t}\ P\ l$; see file bar_t.v for this short proof. □

We now establish the equivalence between $\texttt{af}_\texttt{t}$ predicates and the $\texttt{bar}_\texttt{t}$ predicates. We denote $R \Uparrow l$ for $R \Uparrow [x_1, \ldots, x_n] := R \uparrow x_n \ldots \uparrow x_1$ and we show the following theorem which establishes an informative equivalence between the $\texttt{af}_\texttt{t}$ predicate and the $\texttt{bar}_\texttt{t}$ predicate:

$$\texttt{Theorem bar\_t\_af\_t\_eq}\ X\ (R : X \to X \to \mathbb{P})\ l : \ \texttt{af}_\texttt{t}(R \Uparrow l) \leftrightarrow \texttt{bar}_\texttt{t}\ (\texttt{good}\ R)\ l.$$

***Proof*** It is easy to prove $\texttt{bar}_\texttt{t}\ (\texttt{good}\ R)\ l \to \texttt{af}_\texttt{t}(R \Uparrow l)$ by induction on $\texttt{bar}_\texttt{t}\ (\texttt{good}\ R)\ l$. For the converse implication, we generalize the statement to $\texttt{af}_\texttt{t}\ R' \to \forall S\ l,\ R' \subseteq S \Uparrow l \to \texttt{bar}_\texttt{t}\ (\texttt{good}\ S)\ l$ which we show by induction on $\texttt{af}_\texttt{t}\ R'$. Then, given $R$ and $l$, we instantiate the result with $R' := R \Uparrow l$ and $S := R$ and get $\texttt{af}_\texttt{t}(R \Uparrow l) \to \texttt{bar}_\texttt{t}\ (\texttt{good}\ R)\ l$; see file af_bar_t.v for details. □

$$\texttt{Corollary af\_t\_bar\_t}\ X\ (R : X \to X \to \mathbb{P}) : \ \texttt{af}_\texttt{t}\ R \leftrightarrow \texttt{bar}_\texttt{t}\ (\texttt{good}\ R)\ [].$$

***Proof*** Straightforward instance of bar_t_af_t_eq with $l := []$. □

[10] see Corollary af_t_bar_t below.

### 3.4 A Constructive form of König's Lemma

Using the inductive *Fan theorem* [10], we derive a *constructive König's lemma*. Brouwer's Fan theorem can be proved equivalent to the binary form of König's lemma [21]. So one could wrongfully be led to the conclusion that both of these results cannot be constructively established. Here we explain that using suitable inductive definitions, such results can perfectly be established constructively.

For the rest of this section, we assume a type $X$ : `Type`. We recall the inductive interpretation of the Fan theorem [10]. Given a list of lists $ll : \mathbb{L}\,(\mathbb{L}\,X)$, we define the list of choice sequences (or Fan) of $ll$ denoted `list_fan` $ll$

$$\texttt{Definition list\_fan} : \mathbb{L}\,(\mathbb{L}\,X) \to \mathbb{L}\,(\mathbb{L}\,X).$$

The precise definition of `list_fan` uses auxiliary functions (see file list_fan.v) but this is unimportant here. Only the following specification which characterizes the elements of `list_fan` $[l_1; \ldots; l_n]$ as choices sequences for $[l_1; \ldots; l_n]$ matters.

$$[x_1; \ldots; x_m] \in_1 \texttt{list\_fan}\,[l_1; \ldots; l_n] \leftrightarrow n = m \wedge x_1 \in_1 l_1 \wedge \cdots \wedge x_n \in_1 l_n \quad \text{(FAN)}$$

These are the lists composed of one element of $l_1$, then one element of $l_2$ ... and one element of $l_n$. The following result [10] states that if $P$ is monotonic and bound to be reached by successive extensions starting from [], then $P$ is bound to be reached *uniformly* over the finitary Fan represented by choices sequences.

$$\texttt{Theorem fan\_t\_on\_list}\,(P : \mathbb{L}\,X \to \mathbb{P})\,\big(\mathit{Mono}_P : \forall\, x\,l,\, P\,l \to P(x :: l)\big) :$$
$$\texttt{bar}_\texttt{t}\;P\;[] \to \texttt{bar}_\texttt{t}\big(\texttt{fun}\;ll \mapsto \forall_1\;P\;(\texttt{list\_fan}\;ll)\big)\;[].$$

***Proof*** This proof is implemented in the file bar_t.v and follows the script of [10]. We give a short overview here. Using $\mathit{Mono}_P$, we show by induction on $l$ that $P\,m \to P(l +\!\!+ m)$. Then we define the predicate $Q := \texttt{fun}\,u\,ll \mapsto \forall_1\,\big(\texttt{fun}\;v \mapsto P(v +\!\!+ u)\big)\;(\texttt{list\_fan}\;ll)$ and show $\mathit{Mono}_Q : \forall\, u\,x\,l,\ Q\,u\,l \to Q\,u\,(x :: l)$. By induction on the predicate $\texttt{bar}_\texttt{t}\;P\;u$, we show $\texttt{bar}_\texttt{t}\;P\;u \to \texttt{bar}_\texttt{t}\;(Q\,u)\;[]$. As a consequence, from $\texttt{bar}_\texttt{t}\;P\;[]$ we deduce $\texttt{bar}_\texttt{t}\;(Q\,[])\;[]$ hence the result since $Q\;[]$ is equivalent to $\texttt{fun}\;ll \mapsto \forall_1\;P\;(\texttt{list\_fan}\;ll)$. □

We derive the following strong form of König's lemma. Given an almost full (redundancy) relation $\prec_\texttt{r} : X \to X \to \mathbb{P}$ and a sequence of finitary choices $f : \mathbb{N} \to \mathbb{L}\,X$, beyond an effective lower-bound $n$, every finite prefix of a choice sequence for $f$ is redundant.

$$\texttt{Theorem Constructive\_Koenigs\_lemma}\,(\prec_\texttt{r} : X \to X \to \mathbb{P})\,(f : \mathbb{N} \to \mathbb{L}\,X) :$$
$$\texttt{af}_\texttt{t}(\prec_\texttt{r}) \to \big\{n : \mathbb{N} \;\big|\; \forall m,\ n \le m \to \forall_1\,(\texttt{good} \prec_\texttt{r})\,(\texttt{list\_fan}\,[f_{m-1}; \ldots; f_0])\big\}.$$

***Proof*** From $\texttt{af}_\texttt{t}(\prec_\texttt{r})$ one gets $\texttt{bar}_\texttt{t}\;(\texttt{good} \prec_\texttt{r})\;[]$ using Corollary `af_t_bar_t`. Because $\texttt{good} \prec_\texttt{r}$ is a monotonic relation (i.e. $\texttt{good} \prec_\texttt{r} l \to \texttt{good} \prec_\texttt{r} (x :: l)$), we can apply Theorem `fan_t_on_list` to get $\texttt{bar}_\texttt{t}\;(\texttt{fun}\;ll \mapsto \forall_1\;(\texttt{good} \prec_\texttt{r})\;(\texttt{list\_fan}\;ll))\;[]$. We apply Theorem `bar_t_inv` using $f$ and compute $n$ s.t. $\forall_1\;(\texttt{good} \prec_\texttt{r})\;(\texttt{list\_fan}\,[f_{n-1}; \ldots; f_0])$.

Now let $m \ge n$ and $l \in_1 \texttt{list\_fan}\,[f_{m-1}; \ldots; f_0]$. By Equation (FAN), we can write $l = [x_{m-1}; \ldots; x_n; x_{n-1}; \ldots; x_0]$ and split it into $l_1 := [x_{m-1}; \ldots; x_n]$ and $l_2 := [x_{n-1}; \ldots; x_0]$ so that $l = l_1 +\!\!+ l_2$ and $l_2 \in_1 \texttt{list\_fan}\,[f_{n-1}; \ldots; f_0]$. As a consequence, $\texttt{good} \prec_\texttt{r} l_2$ holds and hence, so does $\texttt{good} \prec_\texttt{r} (l_1 +\!\!+ l_2)$. See file koenig.v for details. □

### 3.5 Redundancy Avoiding Search via König's Lemma

We explain how to use `Constructive_Koenigs_lemma` to bound a redundancy avoiding finitely branching search space described by a (potentially infinite) tree $\mathcal{T}$ indexed by some type $X$. Because $\mathcal{T}$ is finitely branching, we can define a function $\texttt{next}_{\mathcal{T}} : \mathbb{L}\,X \to \mathbb{L}\,X$ s.t.

$$y \in_{\mathbb{l}} \texttt{next}_{\mathcal{T}}\, l \;\leftrightarrow\; y \text{ is the son in } \mathcal{T} \text{ of some } x \in_{\mathbb{l}} l.$$

Given a start node $x_0$ in $\mathcal{T}$, the term $\texttt{next}^n_{\mathcal{T}}\,[x_0]$ lists all the nodes of $\mathcal{T}$ which are the $n$th generation descendants of $x_0$. Hence, the Fan

$$\texttt{list\_fan}\left[\texttt{next}^{n-1}_{\mathcal{T}}\,[x_0]; \ldots; \texttt{next}^{0}_{\mathcal{T}}\,[x_0]\right]$$

contains all the branches of length $n$ starting at $x_0$ in the tree $\mathcal{T}$. Beware that this over-approximation is usually strict. When the redundancy relation $\prec_{\mathrm{r}}$ is almost full, we can apply `Constructive_Koenigs_lemma` to $f := \texttt{fun}\, n \mapsto \texttt{next}^n_{\mathcal{T}}\,[x_0]$ and compute an upper bound on the length of irredundant (i.e. $\texttt{bad} \prec_{\mathrm{r}}$) branches originating at $x_0$ in $\mathcal{T}$.

## 4 Decision via Redundancy-Free Proof-Search

In this section, we describe the mechanization of a generic constructive decider based on redundancy-avoiding proof-search. We first give an informal account of the main arguments, then we proceed with a more formal description of these steps in the language of Coq. Except for the previously described tree.v and almost_full.v libraries, all the following development is contained in the file proof.v.

### 4.1 Overview of the Assumptions and Main Arguments

Let us consider a type $\mathbb{S}$ of *statements* representing object-level logical propositions. These statements could, depending on the intended application, be formulæ like in Hilbert style proof systems, or sequents in sequent proof systems or in some versions of natural deduction, or more generally structures like nested sequents or even consecutions in Display logic.

Statements are items to be proved or refuted (by showing the impossibility of a proof as a term of type $\texttt{has\_proof}\ s \to \texttt{False}$). For this, we describe object-level proof systems as sets of valid rule instances. These instances are generally represented as

$$\frac{H_1 \quad \cdots \quad H_n}{C} \qquad \text{or alternatively} \qquad C \blacktriangleleft [H_1; \ldots; H_n]$$

where $C : \mathbb{S}$ is the *conclusion* of the instance and $[H_1; \ldots; H_n] : \mathbb{L}\,\mathbb{S}$ is the list of *premises*. We collect the set of valid rule instances into a binary relation $\blacktriangleleft : \mathbb{S} \to \mathbb{L}\,\mathbb{S} \to \mathbb{P}$ between individual statements ($\mathbb{S}$ viewed as conclusions) and list of statements ($\mathbb{L}\,\mathbb{S}$ viewed as premises). Hence, the validity of the above rule instance in the proof system is expressed by the predicate $C \blacktriangleleft [H_1; \ldots; H_n]$. When dealing with proof-search based decision, infinite horizontal branching of proof-search is usually forbidden. Hence, for a given $C : \mathbb{S}$, only finitely many rule instances exist with $C$ as conclusion. Moreover, that finite set of instances must be

computable to be able to enumerate the next steps of backward proof-search. We denote this property by `rules_fin` and we say that $\blacktriangleleft$ *has finite inverse images*.[11]

Valid rules instances are combined to form proof trees. A proof tree is a finite tree of statements where each node is a valid rule instance. Object-level proofs are proof trees of their root node and $n$-bounded proofs are proofs of height bounded by $n$. Because of the finite inverse images property `rules_fin`, the set of $n$-bounded proofs of a given statement $s_0$ is finite and computable. We define the notion of *minimal proof*, which is a proof of minimal height among the proofs of the same statement. Every proof can effectively be transformed into a minimal proof by searching among the finitely many proofs of lesser height. An *everywhere minimal proof* is such that each of its sub-proof is minimal. Every proof can effectively be transformed into an everywhere minimal proof.

Our generic constructive technique assumes a binary redundancy relation $\prec_{\mathrm{r}}$ between statements, redundancy which satisfies Curry's lemma: every proof containing a redundancy can be contracted into a lesser proof. As a consequence, everywhere minimal proofs are redundancy free. If we moreover assume that the redundancy relation $\prec_{\mathrm{r}}$ is almost full, i.e. a *constructive WQO*, then every infinite sequence of statements contains a redundancy. However, remember that in Kripke's lemma 2, only the restriction of $\prec_{\mathrm{r}}$ to finitely generated sequents is a WQO. Hence we only assume $\prec_{\mathrm{r}}$ to be almost full on the set of sub-statements of an initial statement,[12] say $s_0$. By using the constructive version of König's lemma of Sect. 3.4, we show that every sequence of sub-statements of $s_0$ longer than a bound $n_0$ contains a redundancy. The bound $n_0$ can be computed from $s_0$ alone. As a consequence, every irredundant proof of $s_0$ is a $n_0$-bounded proof. And deciding the provability of $s_0$ is reduced to testing whether the (finite) set of $n_0$-bounded proofs of $s_0$ is empty or not.

### 4.2 Proofs, Minimal Proofs and Everywhere Minimal Proofs

In this section, we fix an (abstract) type $\mathbb{S}$ of statements and a collection of valid rule instances represented by a relation $\blacktriangleleft : \mathbb{S} \to \mathbb{L}\,\mathbb{S} \to \mathbb{P}$ which has the finite inverse image property.

```
Variables   (𝕊 : Type) (◀ : 𝕊 → 𝕃 𝕊 → ℙ).
Hypothesis  rules_fin : ∀c : 𝕊, fin₁ (c ◀ ·).
```

Here, we use $c \blacktriangleleft \cdot$ as a short notation for $\mathtt{fun}\ hs : \mathbb{L}\,\mathbb{S} \mapsto c \blacktriangleleft hs$. Hence, not only are there finitely many rule instances for a given conclusion $c$ but the predicate $\mathtt{rules\_fin}\,c$ contains $ll_c : \mathbb{L}\,(\mathbb{L}\,\mathbb{S})$, an *effective list* of those instances which verifies the property $hs \in_{\mathbb{l}} ll_c \leftrightarrow c \blacktriangleleft hs$ for any $hs : \mathbb{L}\,\mathbb{S}$. This effective aspect of finite branching is often implicit in studies on proof-search, because if one cannot even compute the valid instances for a given conclusion, then there is no way to implement backward proof-search.

The object-level notion of proof is based on that of *proof tree*, a sub-type of the type $\mathbb{T}\,\mathbb{S}$ of trees (of statements) as defined in Sect. 3.2. We define the predicate `proof_tree` that satisfies the below (recursive) characteristic property:

```
Definition proof_tree : 𝕋 𝕊 → ℙ.
  ∀s l, proof_tree ⟨s | l⟩ ↔ s ◀ map root l ∧ ∀t, t ∈₁ l → proof_tree t.
```

[11] Typically, systems which include a *cut-rule* do not satisfy the `rules_fin` property which is why *cut-elimination* is viewed as a critical requisite to design sequent-based decision procedures. The same remark holds for the *modus-ponens* rule of Hilbert systems, usually making them unsuited for decision procedures.

[12] For this, we need a notion of sub-statement that is reflexive, transitive and such that valid rules instances possess the sub-statement property.

i.e. trees of statements where each node is a valid rule instance, the conclusion being the node itself and the premises being the children of the node. Given a statement $s$, a *proof* of $s$ is a proof tree $t$ with root $s$, and a *n-bounded proof* is a proof of height bounded by $n$:

```
Definition proof (s : 𝕊) (t : 𝕋 𝕊) := proof_tree t ∧ root t = s.
Definition bproof (n : ℕ) (s : 𝕊) (t : 𝕋 𝕊) := proof s t ∧ ht t ≤ n.
```

Proofs of a given statement $s$ are not necessarily finitely many but because of the finite inverse image property `rules_fin`, $n$-bounded proofs are:

```
Proposition bproof_finite_t (n : ℕ) : ∀s : 𝕊, fin_t(bproof n s).
```

***Proof*** We proceed by induction on $n$. For $n = 0$, we have `bproof 0 s p ↔ False` hence `bproof 0 s` is empty and thus finite. For $1 + n$, we show the following equivalence

$$\texttt{bproof}\ (1+n)\ s\ p \leftrightarrow \exists k, s_1, \ldots s_k, p_1, \ldots p_k, \begin{cases} p = \langle s \mid [p_1; \ldots; p_k] \rangle \wedge s \blacktriangleleft [s_1; \ldots; s_k] \\ \wedge\ \texttt{bproof}\ n\ s_1\ p_1 \wedge \cdots \wedge \texttt{bproof}\ n\ s_k\ p_k. \end{cases}$$

Because of `rules_fin`, there are only finitely many lists $[s_1; \ldots; s_k]$ s.t. $s \blacktriangleleft [s_1; \ldots; s_k]$. By the induction hypothesis, we know that `bproof` $n\ s$ is finite for any $s$. Hence, for any list $[s_1; \ldots; s_k]$, there are only finitely many lists $[p_1; \ldots; p_k]$ such that `bproof` $n\ s_1\ p_1 \wedge \cdots \wedge$ `bproof` $n\ s_k\ p_k$. Since $p$ must be of the form $p = \langle s \mid [p_1; \ldots; p_k] \rangle$, this gives only finitely many possible values for trees $p$ such that `bproof` $(1 + n)\ s\ p$. □

We introduce the notion of *minimal proof*, that is a proof of minimal height among the proofs with a given root $s$. We show that every proof $t$ can be transformed into a minimal proof by a straightforward finitary search of the shortest among the (`ht` $t$)-bounded proofs of $s$, of which a list can be computed using `bproof_finite_t`.

```
Definition  min_proof s t := proof s t ∧ ∀t', proof s t' → ht t ≤ ht t'.
Proposition proof_minimize s t : proof s t → {t_min | min_proof s t_min}.
```

But to exploit Curry's lemma, we need a much stronger minimality property: this is the notion of *everywhere minimal proof tree*, where every sub-tree is a minimal proof (of its own root). Contrary to minimal proofs of which some e.g. short sub-proofs can be further shortened (because this would not impact the overall height of the proof), everywhere minimal proof trees cannot be shortened further in any way.

```
Definition emin_ptree : 𝕋 𝕊 → ℙ.
  ∀s l, emin_ptree ⟨s | l⟩ ↔ min_proof s ⟨s | l⟩ ∧ ∀t, t ∈₁ l → emin_ptree t.
```

We show that every proof can effectively be transformed into an everywhere minimal proof.

```
Definition  emin_proof s t := proof s t ∧ emin_ptree t.
Proposition proof_eminimize s t : proof s t → {t_em | emin_proof s t_em}.
```

***Proof*** The argument proceeds by induction on the height `ht` $t$ of the proof tree $t$. It uses `proof_minimize` to compute a minimal proof $t_1$ for $s$ and then proceeds inductively on every immediate sub-proof of $t_1$. □

### 4.3 The Completeness of Irredundant Proofs via Curry's Lemma

We assume an abstract notion of redundancy on statements as binary relation $\prec_r : \mathbb{S} \to \mathbb{S} \to \mathbb{P}$. A list of statements $b : \mathbb{L}\,\mathbb{S}$ (such as e.g. a proof branch) is redundant if it contains a good

pair for $\prec_r$, which is denoted by $\texttt{good}\ \prec_r\ b$ (see Sect. 3.3). The list $b$ is *irredundant* if it contains no good pair, i.e. $\texttt{bad}\ \prec_r\ b$. A tree $t : \mathbb{T}\,\mathbb{S}$ is an *irredundant proof* if it is a proof and every branch of the tree is irredundant. Since branches are read from the root to leaves, we use of `rev` to reverse lists.

$$\texttt{Definition irred\_proof}\ s\ t := \texttt{proof}\ s\ t \wedge \forall b,\ \texttt{branch}\ t\ b \to \texttt{bad} \prec_r (\texttt{rev}\ b).$$

We now state the assumption `Curry` abstracting Curry's lemma:

$$\texttt{Hypothesis Curry} : \forall s_1\ s_2\ t,\ \texttt{proof}\ s_2\ t \to s_1 \prec_r s_2 \to \exists t',\ \wedge \begin{cases} \texttt{proof}\ s_1\ t' \\ \texttt{ht}\ t' \le \texttt{ht}\ t. \end{cases}$$

assumption under which everywhere minimal proofs become irredundant:

$$\texttt{Lemma proof\_emin\_irred}\ s\ t :\ \texttt{emin\_proof}\ s\ t \to \texttt{irred\_proof}\ s\ t.$$

***Proof*** Given any branch $b$ of an everywhere minimal proof tree $t$, we show that $b$ cannot contain a redundancy, i.e. we show $\texttt{bad} \prec_r (\texttt{rev}\ b)$. So let us assume a good pair $s_1 \prec_r s_2$ in $\texttt{rev}\ b = l_1 +\!\!+ s_2 :: l_2 +\!\!+ s_1 :: l_3$ and let us derive a contradiction. Let $t_1/t_2$ be the sub (proof) trees of roots $s_1/s_2$. We have $b = \texttt{rev}\ l_3 +\!\!+ s_1 :: \texttt{rev}\ l_2 +\!\!+ s_2 :: \texttt{rev}\ l_1$ hence $s_2$ occurs after $s_1$ in $b$. Then $t_2$ is a strict sub-tree of $t_1$ and thus $\texttt{ht}\ t_2 < \texttt{ht}\ t_1$. Using `Curry`, we get a proof $t_1'$ of $s_1$ with $\texttt{ht}\ t_1' \le \texttt{ht}\ t_2$. We derive $\texttt{ht}\ t_1' < \texttt{ht}\ t_1$, and thus $t_1$ is not a minimal proof of $s_1$, contradicting the everywhere minimality of $t$. □

We now show the completeness of irredundant proofs, i.e. that under `Curry`'s assumption, it is enough to search only for irredundant proofs to determine the provability of a statement. We show that every proof can effectively be transformed into an irredundant one.

$$\texttt{Theorem proof\_reduce}\ s\ t :\ \texttt{proof}\ s\ t \to \left\{t_{\text{irr}} \;\middle|\; \texttt{irred\_proof}\ s\ t_{\text{irr}}\right\}.$$

***Proof*** Direct combination of `proof_eminimize` and `proof_emin_irred`. □

### 4.4 Bounding the Height of Irredundant Proofs

Kripke used König's lemma to prove the finiteness of the finitely branching irredundant proof-search tree by showing that it cannot have infinite branches, because those would necessarily be redundant. The constructive argument works positively by showing that one can compute a uniform upper-bound over the length of irredundant proof-search branches. We use the constructive version of König's lemma of Sect. 3.4 to compute that bound.

To capture the *sub-formula property* in our setting, we assume an abstract notion of sub-statement denoted by $s_1 \sqsupseteq_{sf} s_2$. Beware that

$$s_1 \sqsupseteq_{sf} s_2 \text{ intuitively reads as "the sub-formulæ of } s_2 \text{ are also sub-formulæ of } s_1\text{"}$$

and *not* as "$s_2$ is a sub-formula of $s_1$." We postulate that $\sqsupseteq_{sf}$ is both reflexive (`sf_refl`) and transitive (`sf_trans`) and more importantly, that every rule instance preserves sub-statements bottom-up:

$$\begin{aligned}\texttt{Variables}\ (\sqsupseteq_{sf} : \mathbb{S} \to \mathbb{S} \to \mathbb{P})\ &(\texttt{sf\_refl} : \forall s,\ s \sqsupseteq_{sf} s) \\ &(\texttt{sf\_trans} : \forall s_1\ s_2\ s_3,\ s_1 \sqsupseteq_{sf} s_2 \to s_2 \sqsupseteq_{sf} s_3 \to s_1 \sqsupseteq_{sf} s_3) \\ &(\texttt{sf\_rules} : \forall c\ hs,\ c \blacktriangleleft hs \to \forall h, h \in_1 hs \to c \sqsupseteq_{sf} h).\end{aligned}$$

The hypothesis `sf_rules` states that any premise $h \in_1 hs$ of any valid rule instance $c \blacktriangleleft hs$ is composed of sub-formulæ of the conclusion $c$.

We now follow the construction outlined in Sect. 3.5 to bound the irredundant proof-search space. Starting from an initial statement $s_0 : \mathbb{S}$, we build the *proof-search sequence from* $s_0$ as the sequence of iterations $\texttt{fun}\, n \mapsto \texttt{rules\_next}^n\, [s_0]$ of the operator

$$\texttt{Let rules\_next} : \mathbb{L}\,\mathbb{S} \to \mathbb{L}\,\mathbb{S}.$$
$$\forall l\, h,\ h \in_1 \texttt{rules\_next}\, l \leftrightarrow \exists c\, hs,\ c \in_1 l \wedge h \in_1 hs \wedge c \blacktriangleleft hs.$$

which collects in a list the statements that occur as premises of some rule instance which has a conclusion in $l$, i.e. `rules_next` $l$ is the (finite) inverse image of $l$ by valid rules instances. We prove the monotonicity of `rules_next`, i.e. $l \subseteq_1 m \to \texttt{rules\_next}\, l \subseteq_1 \texttt{rules\_next}\, m$. The proof-search sequence is composed of sub-statements of $s_0$:

$$\texttt{Proposition proof\_search\_sf}\ s_0\ n :\ \forall s,\ s \in_1 \texttt{rules\_next}^n\, [s_0] \to s_0 \supseteq_{\text{sf}} s.$$

***Proof*** By induction on $n$ using `sf_refl`, `sf_trans` and `sf_rules`. □

Because $\texttt{rules\_next}^n\, [s_0]$ collects all the statements that might occur at height $n$ in any proof of $s_0$, the branches of length $n$ of proofs of $s_0$ can be covered using the choices sequences over $\left[\texttt{rules\_next}^0\, [s_0]; \ldots; \texttt{rules\_next}^{n-1}\, [s_0]\right]$. We show the following covering property:

$$\texttt{Let}\quad \texttt{FAN}\, n\, s_0 := \texttt{list\_fan}\left[\texttt{rules\_next}^{n-1}\, [s_0]; \ldots; \texttt{rules\_next}^0\, [s_0]\right].$$
$$\texttt{Lemma ptree\_proof\_search}\ (t : \mathbb{T}\,\mathbb{S})\ (b : \mathbb{L}\,\mathbb{S}) :$$
$$\texttt{branch}\, t\, b \to \texttt{proof\_tree}\, t \to \texttt{rev}\, b \in_1 \texttt{FAN}\, (\texttt{length}\, b)\, (\texttt{root}\, t).$$

***Proof*** The proof proceeds by induction on the `branch` $t$ $b$ predicate. The two base cases are trivial. In the inductive case, it involves the following monotonicity property of `rules_next`: if $s_1 \in_1 \texttt{rules\_next}\ [s_2]$ then $\texttt{rules\_next}^n\ [s_1] \subseteq_1 \texttt{rules\_next}^{1+n}\ [s_2]$. □

We assume our *redundancy hypothesis* denoted `Kripke` which states that the (abstract) relation $\prec_{\text{r}}$ is almost full when restricted to sub-statements of the initial statement $s_0$ (of which the provability is tested). Using constructive König's lemma of Sect. 3.4, we derive:

$$\texttt{Hypo. Kripke} : \forall s_0 : \mathbb{S},\ \texttt{af}_{\text{t}}\left(\prec_{\text{r}} \texttt{restr}\, (\texttt{fun}\, s \mapsto s_0 \supseteq_{\text{sf}} s)\right).$$
$$\texttt{Lemma irredundant\_max\_length}\ s_0 :$$
$$\left\{n_0 \;\middle|\; \forall m,\ n_0 \le m \to \forall_1\, (\texttt{good} \prec_{\text{r}})\, (\texttt{FAN}\, m\, s_0)\right\}.$$

***Proof*** Apply `Constructive_Koenigs_lemma` of Sect. 3.4 to $(\texttt{Kripke}\ s_0)$. □

Notice that we need the informative predicate $\texttt{af}_{\text{t}}$ in `Kripke` to effectively compute the upper-bound. We conclude that irredundant proofs are bounded proofs:

$$\texttt{Lemma proof\_irred\_bounded}\, s_0 : \{n_0 \mid \texttt{irred\_proof}\ s_0 \subseteq \texttt{bproof}\ n_0\ s_0\}.$$

***Proof*** By Lemma `irredundant_max_length`, from $s_0$ we get $n_0$ so that for any $m$ over $n_0$, any list in $\texttt{FAN}\, m\, s_0$ is redundant. Let us now fix an irredundant proof $p$ of $s_0$. To show that $p$ has height bounded by $n_0$, i.e. $\texttt{bproof}\, n_0\, s_0\, p$, it is enough to establish that all the branches of $p$ are shorter than $n_0$. So let us suppose that there is a branch $b$ of $p$ of length $m$ greater than $n_0$. By Lemma `ptree_proof_search`, we derive that $\texttt{rev}\, b \in_1 \texttt{FAN}\, m\, s_0$. But then $\texttt{rev}\, b$ must be redundant, i.e. $\texttt{good} \prec_{\text{r}} (\texttt{rev}\, b)$, contradicting $\texttt{irred\_proof}\ s_0\ p$. □

Hence, given a starting statement $s_0$, we can compute (from $s_0$ and $s_0$ alone) an upper-bound $n_0$ such that every irredundant proof of $s_0$ is $n_0$-bounded.

$$
\begin{aligned}
&\texttt{Theorem proof\_decider } (\mathbb{S} : \texttt{Type})\ (\blacktriangleleft : \mathbb{S} \to \mathbb{L}\,\mathbb{S} \to \mathbb{P})\ (\supseteq_{\texttt{sf}}, \prec_{\texttt{r}} : \mathbb{S} \to \mathbb{S} \to \mathbb{P})\\
&\qquad (\texttt{rules\_fin} : \forall c,\ \texttt{fin}_{\texttt{t}}(c \blacktriangleleft \cdot))\\
&\qquad (\texttt{sf\_refl} : \forall s,\ s \supseteq_{\texttt{sf}} s)\\
&\qquad (\texttt{sf\_trans} : \forall r\ s\ t,\ r \supseteq_{\texttt{sf}} s \to s \supseteq_{\texttt{sf}} t \to r \supseteq_{\texttt{sf}} t)\\
&\qquad (\texttt{sf\_rules} : \forall c\ hs,\ c \blacktriangleleft hs \to \forall h,\ h \in_1 hs \to c \supseteq_{\texttt{sf}} h)\\
&\qquad (\texttt{Curry} : \forall s\ t\ p,\ \texttt{pf}\ t\ p \to s \prec_{\texttt{r}} t \to \exists q,\ \texttt{pf}\ s\ q \wedge \texttt{ht}\ q \leqslant \texttt{ht}\ p)\\
&\qquad (\texttt{Kripke} : \forall s_0,\ \texttt{af}_{\texttt{t}}\,(\prec_{\texttt{r}} \texttt{restr}\,(\texttt{fun}\ s \mapsto s_0 \supseteq_{\texttt{sf}} s))) :\\
&\forall s_0 : \mathbb{S},\ \{p : \mathbb{T}\,\mathbb{S} \mid \texttt{pf}\ s_0\ p\} + \{\forall p : \mathbb{T}\,\mathbb{S},\ \neg(\texttt{pf}\ s_0\ p)\}.
\end{aligned}
$$

**Fig. 3** Constructive decider by redundancy-free proof-search (with $\texttt{pf} := \texttt{proof}\ \{\mathbb{S}\}\ \blacktriangleleft$)

### 4.5 The Constructive Decider Based on Redundancy-Free Proof-Search

Under the accumulated hypotheses starting at Sect. 4.2, we build the constructive decider:

$$\texttt{Theorem proof\_decider}\ (s_0 : \mathbb{S}) :\ \{p \mid \texttt{proof}\ s_0\ p\} + \{\forall p,\ \neg(\texttt{proof}\ s_0\ p)\}.$$

***Proof*** From $s_0$, the algorithm uses Lemma `proof_irred_bounded` to first compute a bound $n_0$ such that every irredundant proof of $s_0$ has height bounded by $n_0$. Second, the algorithm computes the list of $n_0$-bounded proofs of $s_0$ using Proposition `bproof_finite_t`, a list that we can test for emptiness. If it is non-empty then $s_0$ has a $n_0$-bounded proof, and hence $s_0$ has a proof. Otherwise, there is no $n_0$-bounded proofs for $s_0$, thus there is no irredundant proof for $s_0$: this is the property of the upper-bound $n_0$ given by `proof_irred_bounded`. And then, using Theorem `proof_reduce`, there is no proof for $s_0$ at all. □

To get a standalone theorem not dependent on any assumption, we discharge all the assumptions that were stated from Sect. 4.2 upwards and we display the full abstract result `proof_decider` in Fig. 3. Notice that the abbreviation `pf` denotes `proof` $\{\mathbb{S}\}\ \blacktriangleleft$, the argument $\mathbb{S}$ being recognized as implicit.

## 5 The Constructive Decidability of $\mathbf{R}_{\to}$

In this section, we use Theorem `proof_decider` in Fig. 3 to get a mechanized constructive decision procedure for $\mathbf{R}_{\to}$. We follow the steps outlined in Sect. 2 and explain how they are implemented. We have to complete the straightforward inductive definition of the type of Hilbert proofs for $\mathbf{R}_{\to}$ described in Sect. 2.1 with formal definitions of $\mathbf{LR1}_{\to}$ and $\mathbf{LR2}_{\to}$. Then we show the equivalence of these systems $\mathbf{R}_{\to} \leftrightsquigarrow \mathbf{LR1}_{\to} \leftrightsquigarrow \mathbf{LR2}_{\to}$ by giving translations which preserve provability.

### 5.1 Sequents, Redundancy and Its Almost-Fullness

The sequent calculi $\mathbf{LR1}_{\to}$ and $\mathbf{LR2}_{\to}$ have a lot in common. They share the same data-structure for formulæ, sequents —a multiset of hypotheses and a single conclusion,— and they share most of their logical rules.

However, the data-structure of multisets is not an inductive type. There are arguably several ways to deal with them. In our case, we choose to model multisets as lists up to permutations. We write $l \sim_p m$ when the lists $l$ and $m$ are identical modulo some permutation. Hence for every predicate that manipulates multisets, we define it on lists and, when needed, we show that it is closed under arbitrary permutations.

Let us write $\Gamma \vdash A$ for the sequent $(\Gamma, A) : \mathbb{L}\,\mathbb{F} \times \mathbb{F}$ that is a pair composed of a list $\Gamma$ of (hypothesis) formulæ in $\mathbb{F}$ and a single conclusion formula $A$. The notation $\Gamma \vdash A$ denotes a sequent and is irrelevant to its derivability in some system of rules. We define the notion of sub-statement $\supseteq_{\text{sf}}$ as follows: first we define the sub-formula relation $\texttt{sf}_{\mathbb{F}} : \mathbb{F} \to \mathbb{F} \to \mathbb{P}$ as usual, the notation $\texttt{sf}_{\mathbb{F}}\ A\ B$ meaning that "$B$ is a sub-formula of $A$"; then, for a sequent $\Gamma \vdash A$, we define

$$\texttt{sf\_Seq}\ (\Gamma \vdash A)\ (B : \mathbb{F}) := \texttt{sf}_{\mathbb{F}}\ A\ B \vee \exists \gamma, \gamma \in_{\mathbb{l}} \Gamma \wedge \texttt{sf}_{\mathbb{F}}\ \gamma\ B$$

i.e. $\texttt{sf\_Seq}\ (\Gamma \vdash A)$ collects all the sub-formulæ occurring in $\Gamma \vdash A$; finally, we define the sub-statement relation between sequents as

$$(\Gamma \vdash A) \supseteq_{\text{sf}} (\Delta \vdash B) := \texttt{sf\_Seq}\ (\Delta \vdash B) \subseteq \texttt{sf\_Seq}\ (\Gamma \vdash A)$$

meaning that the sub-formulæ of $\Delta \vdash B$ are also sub-formulæ of $\Gamma \vdash A$. Given the previous definition of the sub-formulæ of a given sequent, we show

$$\texttt{Proposition sf\_Seq\_finite\_t}\ (\Gamma \vdash A) :\ \texttt{fin}_{\text{t}}\big(\texttt{sf\_Seq}\ (\Gamma \vdash A)\big).$$

***Proof*** We compute a list composed of exactly the sub-formulæ of all the formulæ that occur in $\Gamma \cup \{A\}$, by induction on formulæ and then on lists of formulæ. □

The *contraction* relation denoted $\succ_{\text{c}}$, and its converse, the *redundancy* relation denoted $\prec_{\text{r}}$, of the same type $\succ_{\text{c}}, \prec_{\text{r}} : \mathbb{L}\,\mathbb{F} \to \mathbb{L}\,\mathbb{F} \to \mathbb{P}$, are defined between two lists of formulæ by

$$\Gamma \succ_{\text{c}} \Delta := \Delta \prec_{\text{r}} \Gamma := \forall \alpha,\ |\Delta|_\alpha \prec_{\text{r}}^{\mathbb{N}} |\Gamma|_\alpha \quad \text{and} \quad n \prec_{\text{r}}^{\mathbb{N}} m := 1 \leq n \leq m \vee n = m = 0.$$

Since the number of occurrences $|\Delta|_\alpha$ is closed under permutations, this definition is suitable for multisets as well. We show that redundancy $\prec_{\text{r}}$ between lists of formulæ is an almost full relation when restricted to a finite sub-type of $\mathbb{F}$:

$$\texttt{Theorem af\_t\_list\_contract}\ (P : \mathbb{F} \to \mathbb{P}) :\ \texttt{fin}_{\text{t}}\ P \to \texttt{af}_{\text{t}}\ (\prec_{\text{r}}\ \texttt{restr}\ \forall_{\mathbb{l}} P).$$

***Proof*** We first establish that $\prec_{\text{r}}^{\mathbb{N}}$ is almost full, i.e. $\texttt{af}_{\text{t}}\left(\prec_{\text{r}}^{\mathbb{N}}\right)$. For this we show $n \prec_{\text{r}}^{\mathbb{N}} m$ iff $n \leq m \wedge (n = 0 \leftrightarrow m = 0)$ and we use Ramsey's theorem `at_t_inter` of Sect. 3.3. For the binary inequality relation $\leq : \mathbb{N} \to \mathbb{N} \to \mathbb{P}$, we show that it is almost full by first proving $\texttt{af}_{\text{t}}(\leq \uparrow n)$ by induction on $n : \mathbb{N}$. Then we get $\texttt{af}_{\text{t}}(\leq)$ using the second rule `in_af_t1`. For the proof of $\texttt{af}_{\text{t}}(\texttt{fun}\ n\ m \mapsto n = 0 \leftrightarrow m = 0)$, apply `in_af_t1` twice and then `in_af_t0`.

Now, assuming $P$ is finite, we show that $\prec_{\text{r}}$ is almost full when restricted to lists $l : \mathbb{L}\,\mathbb{F}$ which contain solely elements in $P$, i.e. which lists $l$ satisfy $\forall_{\mathbb{l}} P\ l$. We first get a list $l_P : \mathbb{L}\,\mathbb{F}$ such that $x \in_{\mathbb{l}} l_P \leftrightarrow P\ x$. Then we prove $\texttt{af}_{\text{t}}\left(\texttt{fun}\ l\ m \mapsto \forall x,\ x \in_{\mathbb{l}} l_P \to |l|_x \prec_{\text{r}}^{\mathbb{N}} |m|_x\right)$ by structural induction on $l_P$. For the induction step, we use Ramsey's theorem `af_t_prod` of Sect. 3.3. These mechanized proofs occur in files af_t.v and list_contract_af.v. □

Notice than in case $P$ is an infinite sub-type then $\prec_{\text{r}}$ restricted to $\forall_{\mathbb{l}} P$ is not an almost full relation. To see that, consider an injective sequence $f : \mathbb{N} \to \mathbb{F}$ such that $\forall n,\ P\ f_n$. Then the sequence $\texttt{fun}\ n \mapsto [f_n]$ of singleton lists in $\forall_{\mathbb{l}} P$ does not contain any redundant pair.

The *redundancy relation is extended to sequents*, overloading the $\prec_{\text{r}}$ notation with

$$\Delta \vdash B \prec_{\text{r}} \Gamma \vdash A := A = B \wedge \Delta \prec_{\text{r}} \Gamma$$

and we can now establish Kripke's lemma for $\mathbf{LR2}_{\to}$:

$$\texttt{Theorem Kripke\_LR2}\ (\Gamma \vdash A) :\ \texttt{af}_{\text{t}}\left(\prec_{\text{r}}\ \texttt{restr}\ (\texttt{fun}\ \Delta \vdash B \mapsto \Gamma \vdash A \supseteq_{\text{sf}} \Delta \vdash B)\right).$$

***Proof*** We have $\mathtt{af_t}\left(\prec_{\mathrm{r}}\ \mathtt{restr}\ \forall_1(\mathtt{sf\_Seq}\ (\Gamma \vdash A))\right)$ by `sf_Seq_finite_t` and Theorem `af_t_list_contract` above. Then we show $\mathtt{af_t}\left(=\mathtt{restr}\ (\mathtt{sf\_Seq}\ (\Gamma \vdash A))\right)$ using the fact that the identity relation $=$ is almost full on any finite subset; this follows from the pigeon hole principle on which we do not elaborate here. Then we form the product of these two restricted relations $\left(\prec_{\mathrm{r}}\ \mathtt{restr}\ \forall_1(\mathtt{sf\_Seq}\ (\Gamma \vdash A))\right) \times \left(=\mathtt{restr}\ (\mathtt{sf\_Seq}\ (\Gamma \vdash A))\right)$ which is almost full by Ramsey's theorem `af_t_prod` of Sect. 3.3. We project this product on $\prec_{\mathrm{r}}\ \mathtt{restr}\ (\mathtt{fun}\ \Delta \vdash B \mapsto \Gamma \vdash A \supseteq_{\mathrm{sf}} \Delta \vdash B)$ using a *surjective relational morphism* to get the result via Proposition `af_t_relmap` of Sect. 3.3. The `Kripke_LR2` result itself is mechanized in file relevant_LR2_dec.v. □

### 5.2 The Sequent Calculi LR1$_\to$ and LR2$_\to$

We instantiate the abstract notion of statement assumed in Sect. 4 with sequents, hence we define $\mathbb{S} := \mathbb{L}\,\mathbb{F} \times \mathbb{F}$. We implement individual rules are subsets of (valid) instances over $\mathbb{S}$, i.e. predicates of type $\mathbb{S} \to \mathbb{L}\,\mathbb{S} \to \mathbb{P}$. Hence, for instance, the rule $\langle {}^\star\supset\rangle$ is implemented by:

$$\mathtt{Inductive\ LR\_rule\_l} : \mathbb{S} \to \mathbb{L}\,\mathbb{S} \to \mathbb{P} := \mathtt{in\_LR}_l : \forall\, \Gamma\ \Delta\ \Theta\ A\ B\ C,$$
$$\Theta \sim_p A \supset B :: \Gamma +\!\!+ \Delta \to \mathtt{LR\_rule\_l}\ (\Theta \vdash C)\ [\Gamma \vdash A;\ B :: \Delta \vdash C].$$

In the file sequent_rules.v, we define `rule_id`, `LR_rule_r`, `LR_rule_l`, `LR2_rule_l`, `rule_cntr` and `rule_cut` all of type $\mathbb{S} \to \mathbb{L}\,\mathbb{S} \to \mathbb{P}$ to implement the rules $\langle \mathrm{AX}\rangle$, $\langle \supset^\star\rangle$, $\langle {}^\star\supset\rangle$, $\langle {}^\star\supset_2\rangle$, $\langle {}^\star\mathrm{W}\rangle$ and $\langle \mathrm{CUT}\rangle$ respectively. The rule $\langle {}^\star\supset_2\rangle$ is interesting to compare with $\langle {}^\star\supset\rangle$

$$\mathtt{Inductive\ LR2\_rule\_l} : \mathbb{S} \to \mathbb{L}\,\mathbb{S} \to \mathbb{P} := \mathtt{in\_LR2}_l : \forall\, \Gamma\ \Delta\ \Theta\ \Sigma\ A\ B\ C,$$
$$\Sigma \sim_p A \supset B :: \Theta \to \mathtt{LR2c}\ (A \supset B)\ \Gamma\ \Delta\ \Theta \to \mathtt{LR2\_rule\_l}\ (\Sigma \vdash C)\ [\Gamma \vdash A;\ B :: \Delta \vdash C].$$

Indeed notice the side condition $\mathtt{LR2c}\ (A \supset B)\ \Gamma\ \Delta\ \Theta$ which implements the controlled absorption of the contraction rule in $\langle {}^\star\supset_2\rangle$. The predicate $\mathtt{LR2c}\ \alpha\ \Gamma\ \Delta\ \Theta$ is defined as

$$\text{where}\quad \begin{array}{l} \mathtt{LR2c}\ \alpha\ \Gamma\ \Delta\ \Theta := \mathtt{LR2c}_2\ |\Gamma|_\alpha\ |\Delta|_\alpha\ |\Theta|_\alpha \wedge \forall \beta,\ \alpha \neq \beta \to \mathtt{LR2c}_1\ |\Gamma|_\beta\ |\Delta|_\beta\ |\Theta|_\beta \\ \begin{cases} \mathtt{LR2c}_1\ x\ y\ z\ :=\ z \leq 1 \wedge z \leq x + y \wedge \max x\ y \leq z \vee 2 \leq z = x + y \\ \mathtt{LR2c}_2\ x\ y\ z\ :=\ z \leq 1 \wedge z \leq x + y \wedge \max x\ y \leq 1 \vee 2 \leq z = x + y \end{cases} \end{array}$$

but other choices are possible for this $\mathtt{LR2c}\ \alpha\ \Gamma\ \Delta\ \Theta$ side condition.

We define the systems **LR1**$_\to$ (cut-free), **LR1**$_\to$ (with cut) and **LR2**$_\to$ (cut-free) as

$$\begin{array}{lll} \mathtt{LR1\_rules} & := & \mathtt{rule\_id} \cup \mathtt{LR\_rule\_r} \cup \mathtt{LR\_rule\_l} \cup \mathtt{rule\_cntr} \\ \mathtt{LR1\_rules\_wc} & := & \mathtt{LR1\_rules} \cup \mathtt{rule\_cut} \\ \mathtt{LR2\_rules} & := & \mathtt{rule\_id} \cup \mathtt{LR\_rule\_r} \cup \mathtt{LR2\_rule\_l} \end{array}$$

where each term has type $\mathbb{S} \to \mathbb{L}\,\mathbb{S} \to \mathbb{P}$ and $\cup$ represents binary predicate union. We can then easily define refined notions of proofs/provability such as e.g.:

- **LR1**$_\to$ proofs (with cuts) as $\mathtt{LR1\_proof} := \mathtt{proof}\ \mathtt{LR1\_rules\_wc}$;
- cut-free **LR1**$_\to$ provability $\mathtt{LR1\_cf\_provable}\ s := \exists t, \mathtt{proof}\ \mathtt{LR1\_rules}\ s\ t$;
- **LR2**$_\to$ provability up-to height $n$ as $\mathtt{LR2\_bprovable}\ n\ s := \exists t, \mathtt{bproof}\ \mathtt{LR2\_rules}\ n\ s\ t$.

Then we establish soundness theorems for translations between these proof systems, leading to the equivalence between those systems; see file relevant_equiv.v for these mechanized high-level results.

`Theorem HR_LR1` $A$ : `HR_proof` $A \to$ `LR1_proof`$([] \vdash A)$.

***Proof*** Straightforward by induction on `HR_proof` $A$. □

`Theorem LR1_cut_admissibility : LR1_provable` $\subseteq$ `LR1_cf_provable`.

***Proof*** This is a cut-admissibility result and many proofs are possible but none of them are really straightforward. We use a semantic proof based the soundness of phase semantics and Okada's argument [15]; see file sem_cut_adm.v. □

`Theorem LR1_cf_LR2 : LR1_cf_bprovable` $\subseteq$ `LR2_bprovable`.

Of which the proof is postponed to Sect. 5.3 because it involves Curry's lemma.

`Theorem LR2_HR` $A$ : `LR2_proof` $([] \vdash A) \to$ `HR_proof` $A$.

***Proof*** We generalize the statement to `LR2_proof` $(\Gamma \vdash A) \to$ `HR_proof` $(\Gamma \ni A)$ where $\ni$ is defined by $[\gamma_1; \ldots; \gamma_k] \ni A := \gamma_k \supset \cdots \supset \gamma_1 \supset A$. We prove the generalized statement by induction on `LR2_proof` $(\Gamma \vdash A)$. The only difficulty is to show that the side condition `LR2c` $(A \supset B)\ \Gamma\ \Delta\ \Theta$ implies contraction, i.e. that $A \supset B :: \Gamma +\!\!+ \Delta \succ_c A \supset B :: \Theta$ holds. □

We finish with a proof that $\mathbf{LR2}_{\to}$ has finite inverse images:

`Lemma LR2_rules_finite_t` $A$ : $\mathtt{fin}_t$(`LR2_rules` $A$).

***Proof*** It is sufficient to show that this property holds for each of its individual rules: `rule_id`, `LR_rule_r` and `LR2_rule_l`. In particular, for rule `LR2_rule_l`, we use $\mathtt{fin}_t\big(\mathtt{fun}\, c \mapsto$ `let` $(\Gamma, \Delta) := c$ `in LR2c` $\alpha\ \Gamma\ \Delta\ \Theta\big)$ which states that there are only finitely many pairs $(\Gamma, \Delta)$ such that `LR2c` $\alpha\ \Gamma\ \Delta\ \Theta$. Hence even though the side condition `LR2c` $(A \supset B)\ \Gamma\ \Delta\ \Theta$ allows for many contractions to occur in the instances of rule `LR2_rule_l`, it still limits the number of possible instances to finitely many; see file relevant_LR2.v for the mechanized proof. □

### 5.3 Curry's Lemma for $\mathbf{LR2}_{\to}$

We state and prove Curry's lemma using the notion of bounded provability.

`Lemma LR2_Curry` $n\ \Gamma\ \Delta\ A\ B$ :
$\Delta \vdash B \prec_r \Gamma \vdash A \to$ `LR2_bprovable` $n\ (\Gamma \vdash A) \to$ `LR2_bprovable` $n\ (\Delta \vdash B)$.

***Proof*** The proof is implemented in file relevant_LR2.v and proceeds by induction on the second argument `LR2_bprovable` $n\ (\Gamma \vdash A)$, using a tailored induction principle implemented under the name `LR2_bprovable_ind`. The following simulation property

$$\mathtt{LR2c}\ \alpha\ \Gamma\ \Delta\ \Theta \to \alpha :: \Theta \succ_c \Sigma \to \exists \Gamma'\ \Delta'\ \Theta', \begin{cases} \Gamma \succ_c \Gamma' \wedge \Delta \succ_c \Delta' \wedge \\ \Sigma \sim_p \alpha :: \Theta' \wedge \mathtt{LR2c}\ \alpha\ \Gamma'\ \Delta'\ \Theta' \end{cases}$$

is essential to deal with the case of rule `LR2_rule_l` and proved in the file relevant_contract.v under the name `LR2c_contract_cons`. □

`Theorem LR1_cf_LR2 : LR1_cf_bprovable` $\subseteq$ `LR2_bprovable`.

***Proof*** The only difficulty in this proof by induction on `LR1_cf_bprovable` $n$ $(\Gamma \vdash A)$ is the case of the contraction rule `rule_cntr` and it is solved using Lemma `LR2_Curry` above. □

### 5.4 Decidability for $\mathbf{R}_{\to}$ via $\mathbf{LR2}_{\to}$

We can derive a constructive decider for the $\mathbf{LR2}_{\to}$ sequent calculus:

```
Theorem LR2_decider (Γ ⊢ A) :
  {t | proof LR2_rules (Γ ⊢ A) t} + {∀t, ¬ proof LR2_rules (Γ ⊢ A) t}.
```

***Proof*** We use the `proof_decider` theorem in Fig. 3 in combination with `Kripke_LR2` for Sect. 5.1 and `LR2_Curry` from Sect. 5.3; see file relevant_LR2_dec.v. □

Using the soundness of translations between $\mathbf{R}_{\to} \leftrightsquigarrow \mathbf{LR1}_{\to} \leftrightsquigarrow \mathbf{LR2}_{\to}$, we get the constructive decider for $\mathbf{R}_{\to}$ specified in Sect. 2.1:

```
Theorem HR_decider :  ∀A : 𝔽, HR_proof A + (HR_proof A → False).
```

***Proof*** Given a formula $A$, use `LR2_decider` to decide whether $[] \vdash A$ has an $\mathbf{LR2}_{\to}$ proof or not. In case $[] \vdash A$ has an $\mathbf{LR2}_{\to}$ proof, then by Theorem `LR2_HR` of Sect. 5.2, we get an inhabitant of `HR_proof` $A$. Otherwise, from the function ∀$t$, ¬ `proof LR2_rules` ($\Gamma \vdash A$) $t$ we show that no term of type `HR_proof` $A$ can exist. So let us assume $p_A$ : `HR_proof` $A$. By Theorems `HR_LR1`, `LR1_cut_admissibility` and `LR1_cf_LR2` of Sect. 5.2, their must exist a proof $t_A$ of $[] \vdash A$ in $\mathbf{LR2}_{\to}$, which thus contradicts ¬ `proof LR2_rules` ($[] \vdash A$) $t_A$; see file logical_deciders.v for the mechanized proof. □

## 6 Conclusion and Perspectives

We present an abstract and constructive view of Kripke–Curry's method for deciding Implicational Relevance Logic $\mathbf{R}_{\to}$. We get an axiom-free Coq implementation that we instantiate on $\mathbf{LR2}_{\to}$ to derive a constructive decider for $\mathbf{R}_{\to}$. Although not presented in this paper, our implementation includes a constructive decider for implicational intuitionistic logic $\mathbf{J}_{\to}$ which shares the same language for formulæ as $\mathbf{R}_{\to}$. It is based on a variant of Gentzen's sequent calculus **LJ**. Unlike what happens with richer fragments of Relevance Logic [23], extensions of this method to full propositional **IL** would present no difficulty.

From a complexity perspective, Kripke's decidability proof for $\mathbf{R}_{\to}$ based on Dickson's lemma might be analyzed using control functions as in [8] to classify its complexity in the Fast Growing Hierarchy. Notice however that these techniques involve classical formulations of WQOs and their conversion to a constructive setting is far from evident. Furthermore, the 2- ExpTime complexity characterization of [19] was not obtained via control functions nor Dickson's lemma.

Kripke–Curry's method has a potential use well beyond $\mathbf{R}_{\to}$ or Dickson's lemma and might be able to provide decidability for logics of still unknown and presumably high complexities. A very difficult case would be to get a constructive proof of decidability for the logic of Bunched Implications **BI** [11] based on Kripke–Curry's method. Indeed, as is the case for $\mathbf{LR1}_{\to}$, contraction (and weakening) cannot be completely removed from the bunched sequent calculus **LBI**. It is not obvious what notion of redundancy could be used in that case.

The somewhat short decidability "proof" of ticket entailment [2] contains glitches that were uncovered in [6]. Analyzing that proof attempt is another obvious perspective of this work because it is also based on Kripke–Curry's method. We hope to better characterize the source of the problem and determine if it can be repaired or not, like what was done for the case of a faulty proof of decidability of MELL [22]. The situation is a bit different however because the decidability result for ticket entailment was independently obtained by

Padovani [16] with seemingly much more involved techniques such as the use of Kruskal's tree theorem. Still, Kruskal's tree theorem is also a result about WQOs of which we do already have a mechanized constructive proof in Coq.[13] The mechanization of ticket entailment might not be completely out of reach.

## References


1. Bimbó, K.: The decidability of the intensional fragment of classical linear logic. Theor. Comput. Sci. **597**, 1–17 (2015)
2. Bimbó, K., Dunn, J.M.: On the decidability of implicational ticket entailment. J. Symb. Log. **78**(1), 214–236 (2013)
3. Coquand, T.: A direct proof of the intuitionistic Ramsey Theorem. In: Pitt D.H., Curien PL., Abramsky S., Pitts A.M., Poigné A., Rydeheard D.E. (eds) Category Theory and Computer Science. CTCS 1991. Lecture Notes in Computer Science, vol 530, pp. 164–172. Springer (1991)
4. Coquand, T.: Constructive topology and combinatorics. In: Myers J.P., O'Donnell M.J. (eds) Constructivity in Computer Science. Constructivity in CS 1991. Lecture Notes in Computer Science, vol 613, pp. 159–164. Springer (1992)
5. Curry, H.B.: A Theory of Formal Deductibility. Notre Dame mathematical lectures. University of Notre Dame, Notre Dame (1957)
6. Dawson, J.E., Goré, R.: Issues in Machine-Checking the Decidability of Implicational Ticket Entailment. In: TABLEAUX 2017, LNAI, vol. 10501, pp. 347–363. Springer International Publishing, Berlin (2017)
7. Demri, S., Jurdziński, M., Lachish, O., Lazić, R.: The covering and boundedness problems for branching vector addition systems. J. Comput. Syst. Sci. **79**(1), 23–38 (2012). https://doi.org/10.1016/j.jcss.2012.04.002
8. Figueira, D., Figueira, S., Schmitz, S., Schnoebelen, P.: Ackermannian and Primitive-Recursive Bounds with Dickson's Lemma. LICS **2011**, 269–278 (2011). https://doi.org/10.1109/LICS.2011.39
9. Figueira, D., Lazic, R., Leroux, J., Mazowiecki, F., Sutre, G.: Polynomial-Space Completeness of Reachability for Succinct Branching VASS in Dimension One. In: ICALP 2017, LIPIcs, vol. 80, pp. 119:1–14. Schloss Dagstuhl (2017). https://doi.org/10.4230/LIPIcs.ICALP.2017.119
10. Fridlender, D.: An Interpretation of the Fan Theorem in Type Theory. In: TYPES'98 Selected Papers, LNCS, vol. 1657, pp. 93–105. Springer, Berlin (1999)
11. Galmiche, D., Méry, D., Pym, D.: The semantics of BI and resource tableaux. Math. Struct. Comput. Sci. **15**(6), 1033–1088 (2005). https://doi.org/10.1017/S0960129505004858
12. Kripke, S.: The Problem of Entailment (abstract). J. Symb. Log. **24**, 324 (1959)
13. Larchey-Wendling, D.: Constructive decision via redundancy-free proof-search. In: Automated Reasoning—9th International Joint Conference, IJCAR 2018, Lecture Notes in Computer Science, vol. 10900, pp. 422–438. Springer, Berlin (2018)
14. Meyer, R.K.: Improved Decision Procedures for Pure Relevant Logic, pp. 191–217. Springer, Dordrecht (2001)
15. Okada, M.: A uniform semantic proof for cut-elimination and completeness of various first and higher order logics. Theor. Comput. Sci. **281**(1–2), 471–498 (2002)
16. Padovani, V.: Ticket Entailment is decidable. Math. Struct. Comput. Sci. **23**(3), 568–607 (2013). https://doi.org/10.1017/S0960129512000412
17. Riche, J.: Decision procedure of some relevant logics: a constructive perspective. J. Appl. Non-Class. Log. **15**(1), 9–23 (2005)
18. Riche, J., Meyer, R.K.: Kripke, Belnap, Urquhart and Relevant Decidability & Complexity. In: CSL'99, pp. 224–240. Springer (1999)
19. Schmitz, S.: Implicational Relevance Logic is 2-EXPTIME-Complete. J. Symb. Log. **81**(2), 641–661 (2016)
20. Schmitz, S.: The complexity of reachability in vector addition systems. ACM SIGLOG News **3**(1), 4–21 (2016). https://doi.org/10.1145/2893582.2893585
21. Schwichtenberg, H.: A direct proof of the equivalence between Brouwer's Fan Theorem and König's Lemma with a uniqueness hypothesis. J. UCS **11**(12), 2086–2095 (2005)
22. Straßburger, L.: On the decision problem for MELL. Theor. Comput. Sci. **768**, 91–98 (2019)
23. Urquhart, A.: The undecidability of entailment and relevant implication. J. Symb. Log. **49**(4), 1059–1073 (1984)


[13] See http://www.loria.fr/~larchey/Kruskal.

24. Veldman, W., Bezem, M.: Ramsey's theorem and the pigeonhole principle in intuitionistic mathematics. J. Lond. Math. Soc. **s2–47**(2), 193–211 (1993). https://doi.org/10.1112/jlms/s2-47.2.193
25. Vytiniotis, D., Coquand, T., Wahlstedt, D.: Stop when you are almost-full—adventures in constructive termination. In: ITP 2012, LNCS, vol. 7406, pp. 250–265 (2012)