Constructive Decision via Redundancy-free Proof-Search

Dominique Larchey-Wendling

Université de Lorraine, CNRS, LORIA, Nancy, France
dominique.larchey-wendling@loria.fr

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 · Redundancy-free Proof-Search · Almost Full relations

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 in [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. 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 fragment of linear logic* [1]. However, both of these recent papers are now contested inside the community because of deeply hidden flaws in the arguments [9, footnote 1], [20, footnote 4] and [6, pp 360-362]. 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.

Work partially supported by the TICAMORE project (ANR grant 16-CE91-0002).

From a complexity perspective, S. Schmitz recently gave a 2-ExpTime complexity characterization [19] of the entailment problem for $\mathbf{R}_{\rightarrow}$, implying a decision procedure. However, 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 $\mathbf{R}_{\rightarrow}$, 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.[1]

Our interest in the entailment problem for $\mathbf{R}_{\rightarrow}$ lies in the inherent simplicity and genericity 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 $\mathbf{R}_{\rightarrow}$ is specific because *redundancy is not reduced to repetition*: 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 sequent system **LJ** has a finite search space.

In the case of $\mathbf{R}_{\rightarrow}$, the sequent S_2 is redundant over S_1 if they are cognate 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[2] is a consequence of Ramsey's theorem which, stated positively, can be viewed as the following result [23]: 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 suited in a constructive argumentation.

Let us now present the content of this paper. In Section 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 T. Coquand's [3] direct intuitionistic proof of Ramsey's theorem through an intuitionistic formulation of WQOs as *almost full relations* (AF) [24]. These results are recalled in Section 3. We also give a constructive version of König's lemma based on AF relations.

In Section 4, we give an account of what could be called the central ingredients of Kripke-Curry's proof by outlining the essential steps of our constructive mechanization in the inductive type theory on which Coq is based. Our `proof_decider` of Fig. 3 abstracts away from the particular case of $\mathbf{R}_{\rightarrow}$, by isolating the essential ingredient: an almost full redundancy relation which satisfies Curry's lemma. Finally, in Section 4.6, we instantiate the `proof_decider` into a constructive decision procedure for $\mathbf{R}_{\rightarrow}$. We also discuss the potential applications to other sub-structural logics.

[1] 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).

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

$$\vdash A \supset A \qquad \vdash (A \supset B) \supset (C \supset A) \supset (C \supset B) \qquad \frac{\vdash A \supset B \quad \vdash A}{\vdash B}$$
$$\vdash (A \supset A \supset B) \supset (A \supset B) \qquad \vdash (A \supset B \supset C) \supset (B \supset A \supset C)$$

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

The technical aspects of our proofs are sustained by a Coq v8.6 mechanization which is available under a Free Software license [14]. 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. 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).

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 J. Riche in [17] for solving the missing link — a constructive proof of IDP or of Dickson's lemma, — the use of T. Coquand's approach [4] to *Bar induction* 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 [17,18,19]. But in this paper, we rather use $\supset$ to denote logical implications to avoid conflicting with the Coq notation for function types $T_1 \to T_2$ (see e.g. the below definition of `HR_proof`).

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 provability or proofs, at least for $\mathbf{R}_{\to}$. This can easily be done in Coq using the (informative) inductive predicate:

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

i.e. `HR_proof` A or ($\vdash A$ for short) encodes the *type of proofs of the formula* A in the Hilbert system for $\mathbf{R}_{\to}$ of Fig. 1. A *constructive decidability proof* for $\mathbf{R}_{\to}$

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

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

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

would then be given by a term `HR_decidability` of type:

$$\texttt{HR\_decidability} : \forall A : \texttt{Form},\ \{\texttt{inhabited}(\vdash A)\} + \{\neg\texttt{inhabited}(\vdash A)\}$$

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:

$$\texttt{HR\_decider} : \forall A : \texttt{Form},\ (\vdash A) + (\vdash A \to \texttt{False})$$

that is a total computable function that maps every formula A to either a proof of A or else ensures that no such proof exists. 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 $\mathbf{R}_{\to}$ formulation is (unsurprisingly) not really suited to designing decision procedures based on proof-search. A standard approach is to convert Hilbert's systems into *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 Γ 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 $\vdash A$ if and only if the sequent $\emptyset \vdash A$ has a proof in $\mathbf{LR1}_{\to}$ (with $\emptyset$ as the empty multiset). That result is mechanized in the file relevant_equiv.v.

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 $\mathbf{LR1}_{\to}$. Cut-admissibility is proved using a relational phase semantic in the 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.[3] This is done by replacing both rules $\langle {}^\star\mathrm{W}\rangle$ and $\langle {}^\star\supset\rangle$ with the rule $\langle {}^\star\supset_2\rangle$:

$$\frac{\Gamma \vdash A \quad B, \Delta \vdash C}{\Theta, A \supset B \vdash C}\ \langle {}^\star\supset_2\rangle \qquad \text{with } \texttt{LR2\_condition}(A \supset B, \Gamma, \Delta, \Theta)$$

obtaining the system $\mathbf{LR2}_{\to}$ composed of $\langle \mathrm{AX}\rangle$, $\langle \supset^\star\rangle$ and $\langle {}^\star\supset_2\rangle$. The side $\texttt{LR2\_condition}(A \supset B, \Gamma, \Delta, \Theta)$ 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 Γ, Δ can be contracted at most once. See the definition of `LR2_condition` in file relevant_contract.v for a 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 (see file relevant_equiv.v for the 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 $\Delta \vdash B$ *is redundant over* a sequent $\Gamma \vdash A$ and we denote $\Gamma \vdash A \prec_{\mathrm{r}} \Delta \vdash B$ if $\Gamma \vdash A$ can be obtained from $\Delta \vdash B$ by repeated top-down applications of the contraction rule $\langle {}^\star\mathrm{W}\rangle$. We also characterize redundancy using the number of *occurrences* $|\Gamma|_X$ of the formula X in the multiset Γ:

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

where the binary relation $n \prec_{\mathrm{r}}^{\mathbb{N}} m$ on $\mathbb{N}$ is defined by $n \leqslant m \wedge (n = 0 \Leftrightarrow m = 0)$. Now we can state Curry's lemma.

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

The Coq proof term is `LR2_Curry` in the file relevant_LR2.v. 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. $\Gamma \vdash A \prec_{\mathrm{r}} \Delta \vdash B$ where $\Delta \vdash B$ occurs in the sub-proof of $\Gamma \vdash A$. 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, see Section 4). Hence, while searching for proofs in $\mathbf{LR2}_{\to}$, one can safely stop at redundancies.

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

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 [15]), 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 $\leqslant$ 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 $x_i \leqslant x_j$.*

Lemma 2 (Kripke [12], 1959). *Given a finite set of (sub-)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. We give a "modernized" account of the proof. By Ramsey's theorem, the product (or intersection) of two WQOs is WQO.[4] Hence, the relation $\prec_{\mathrm{r}}^{\mathbb{N}}$ over $\mathbb{N}$ is a WQO as the intersection of two WQOs. By finiteness of $\mathcal{S}$, the identity relation $=_{\mathcal{S}}$ on $\mathcal{S}$ is also a WQO (this is an instance of *the pigeon hole principle*).

Denoting $\prec_{\mathrm{r}}^{\mathcal{S}}$ as the restriction of $\prec_{\mathrm{r}}$ to the sequents composed of formulæ in the finite set $\mathcal{S}$, we can derive the equivalence

$$\Gamma \vdash A \prec_{\mathrm{r}}^{\mathcal{S}} \Delta \vdash B \quad \Longleftrightarrow \quad A =_{\mathcal{S}} B \wedge \textstyle\bigwedge_{X \in \mathcal{S}} |\Gamma|_X \prec_{\mathrm{r}}^{\mathbb{N}} |\Delta|_X$$

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

Kripke's proof of decidability of $\mathbf{LR2}_{\to}$ (and hence $\mathbf{R}_{\to}$) 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 $\mathbf{LR2}_{\to}$, 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 $\langle {}^{\star}\supset_2\rangle$);
- $\mathcal{T}$ cannot have an infinite branch (by Kripke's lemma);
- hence by König's lemma, the proof-search tree $\mathcal{T}$ is finite.

In [17], J. 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 Section 4, we explain how to overcome these two difficulties and transform this method into a mechanized `HR_decider`.

[4] 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 $\mathcal{S}$.

3 Inductive Well Quasi Orders

In this section we describe an inductive formulation of the notion of WQO. We are now going to use the language of Inductive Type Theory instead of Set theoretical language. Type theoretically, Definition 2 becomes: a WQO is a reflexive and transitive predicate $\prec_{\mathrm{r}} : X \to X \to \mathtt{Prop}$ such that for any $f : \mathtt{nat} \to X$, there exists $i, j : \mathtt{nat}$ s.t. $i < j$ and $f_i \prec_{\mathrm{r}} f_j$ (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* [24], from which we derive a constructive proof of Kripke's lemma. Using the inductive *Fan theorem* [10], we derive a *constructive König's lemma*. The corresponding Coq code can be found in the library file almost_full.v.

3.1 Good lists, almost full relations and bar inductive predicates

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 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 a redundancy relation $\prec_{\mathrm{r}} : X \to X \to \mathtt{Prop}$. We define the $\mathtt{good}\ \prec_{\mathrm{r}} : \mathtt{list}\ X \to \mathtt{Prop}$ predicate that characterizes the (finite) lists which contain a good pair:

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

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

Definition 3 (Ir/redundant). *Given a relation* $\prec_{\mathrm{r}} : X \to X \to \mathtt{Prop}$ *called redundancy, a list of values* $l : \mathtt{list}\ X$ *is* redundant *if* $\mathtt{good}\ \prec_{\mathrm{r}}\ l$ *holds and is* irredundant *if* $\neg(\mathtt{good}\ \prec_{\mathrm{r}}\ l)$ *holds.*

We write $\mathtt{bad}\ \prec_{\mathrm{r}}\ l$ when the list l satisfies $\neg(\mathtt{good}\ \prec_{\mathrm{r}}\ l)$. The *lifting of a relation* $R : X \to X \to \mathtt{Prop}$ by $x : X$ is denoted $R \uparrow x$ and characterized by:

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

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 $\mathtt{af}_{\mathrm{t}}$ predicate:[5]

```
Inductive af_t {X : Type} (R : X → X → Prop) : Type :=
  | in_af_t0 : (∀x y, x R y)          → af_t R
  | in_af_t1 : (∀x, af_t (R ↑ x))     → af_t R.
```

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 $\mathtt{af}_{\mathrm{t}}$ is *informative*: it contains a well-founded

[5] The braces around $\{X : \mathtt{Type}\}$ specify an *implicit argument*.

tree of liftings until the relation becomes full (see [24]). This information is important to compute bounds for proof-search. 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 $f : \mathtt{nat} \to X$ 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:

$$\mathtt{Lemma\ af\_t\_inf\_chain}\ (X : \mathtt{Type})\ (\prec_\mathrm{r} : X \to X \to \mathtt{Prop}) :$$
$$\mathtt{af_t}(\prec_\mathrm{r}) \to \forall (f : \mathtt{nat} \to X),\ \{n : \mathtt{nat} \mid \exists\, i\ j,\ i < j < n \wedge f_i \prec_\mathrm{r} f_j\}.$$

The constructive Ramsey theorem [24] states that almost full relations are closed under (binary) intersection, and as a consequence, under direct products:

$$\mathtt{Theorem\ af\_t\_prod}\ (X\ Y : \mathtt{Type})\ (R : X \to X \to \mathtt{Prop})\ (S : Y \to Y \to \mathtt{Prop}) :$$
$$\mathtt{af_t}\ R \to \mathtt{af_t}\ S \to \mathtt{af_t}\,(R \times S).$$

Notice that reflexivity and transitivity of WQOs are completely orthogonal to almost fullness in these results. They play no important role in our development.

The $\mathtt{af_t}(\prec_\mathrm{r})$ property can alternatively be defined by bar inductive predicates [10] as $\mathtt{bar_t}\ (\mathtt{good}\ \prec_\mathrm{r})\ [\,]$ with the following inductive definition:[6]

$$\begin{array}{l}\mathtt{Inductive\ bar_t}\ \{X : \mathtt{Type}\}\ (P : \mathtt{list}\ X \to \mathtt{Prop})\ (l : \mathtt{list}\ X) : \mathtt{Type} :=\\ \quad \mid \mathtt{in\_bar\_t0} : P\ l \to \mathtt{bar_t}\ P\ l\\ \quad \mid \mathtt{in\_bar\_t1} : (\forall x, \mathtt{bar_t}\ P\ (x :: l)) \to \mathtt{bar_t}\ P\ l.\end{array}$$

Hence, $\mathtt{bar_t}\ P\ l$ 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. With this definition, we can derive the (informative) equivalence:

$$\mathtt{Theorem\ bar\_t\_af\_t\_eq}\ X\ \prec_\mathrm{r}\ l : \mathtt{af_t}(\prec_\mathrm{r} \uparrow\uparrow l) \iff \mathtt{bar_t}\ (\mathtt{good}\ \prec_\mathrm{r})\ l.$$

where $R \uparrow\uparrow [x_1, \ldots, x_n] := R \uparrow x_n \ldots \uparrow x_1$ (see file af_bar_t.v for details). And we deduce the equivalence $\mathtt{af_t}(\prec_\mathrm{r})$ iff $\mathtt{bar_t}\ (\mathtt{good}\ \prec_\mathrm{r})\ [\,]$ as the particular case.

3.2 A constructive form of 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 $X : \mathtt{Type}$. We recall the inductive interpretation of the Fan theorem [10]. Given a list of lists $ll : \mathtt{list}\,(\mathtt{list}\ X)$, we define the list of choice sequences (or fan) of ll denoted $\mathtt{list\_fan}\ ll$

$$\mathtt{Definition\ list\_fan} : \mathtt{list}\,(\mathtt{list}\ X) \to \mathtt{list}\,(\mathtt{list}\ X).$$

[6] $[\,]$ and _ :: _ are shorthand notations for the two list constructors.

The precise definition of `list_fan` uses auxiliary functions (see 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.[7]

$$[x_1; \ldots; x_n] \in_1 \texttt{list\_fan}\ [l_1; \ldots; l_n] \iff x_1 \in_1 l_1 \wedge \cdots \wedge x_n \in_1 l_n \qquad \text{(FAN)}$$

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.

$$\texttt{Theorem fan\_t\_on\_list}\ (P : \texttt{list}\ X \to \texttt{Prop})\ \big(H_P : \forall\, x\ l, P\ l \to P(x :: l)\big) :$$
$$\texttt{bar}_\texttt{t}\ P\ [\,] \to \texttt{bar}_\texttt{t}\ (\texttt{fun}\ ll \mapsto \forall l, l \in_1 \texttt{list\_fan}\ ll \to P\ l)\ [\,].$$

The proof of this result is recalled in bar_t.v. Combining `fan_t_on_list` with `bar_t_af_t_eq` and `af_t_inf_chain`, we derive the following strong form of König's lemma. Given an almost full redundancy relation $\prec_\text{r} : X \to X \to \texttt{Prop}$ and a sequence of finitary choices $f : \texttt{nat} \to \texttt{list}\ X$, beyond an effective lower-bound n, every finite prefix of a choice sequence for f is redundant (see koenig.v).

$$\texttt{Theorem Constructive\_Koenigs\_lemma}\ \prec_\text{r}\ (f : \texttt{nat} \to \texttt{list}\ X) :$$
$$\texttt{af}_\texttt{t}(\prec_\text{r}) \to \{n \mid \forall m\ l,\ n \leqslant m \to l \in_1 \texttt{list\_fan}\ [f_{m-1}; \ldots; f_0] \to \texttt{good}\ \prec_\text{r}\ l\}.$$

In Section 4.5, we over-approximate the branches of the proof-search tree as the choice sequences of $f : \texttt{nat} \to \texttt{list stm}$ which collects in $f\,n$ the finitely many sequents that occur at height n in the proof-search tree. Thus we get a uniform upper-bound of the length of irredundant (i.e. $\texttt{bad} \prec_\text{r}$) proof-search branches.

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 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 `stm` of *statements* representing 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.

[7] The notation $x \in_1 l$ is a shortcut for `In` $x\ l$, the (finitary) membership predicate.

Statements are items to be proved or refuted (by showing the impossibility of a proof as a term of type `has_proof` $s \to$ `False`). For this, we describe a proof system as a set of valid rule instances. These instances are generally represented as in the right figure where C : `stm` is the *conclusion* of the instance and $[H_1; \ldots; H_n]$: `list stm` is the list of *premises*. We collect the set of valid rule instances into a binary relation `rules` : `stm` $\to$ `list stm` $\to$ `Prop` between individual statements (`stm` viewed as conclusions) and list of statements (`list stm` viewed as premises). Hence, the validity of the above rule instance in the proof system is expressed by the predicate `rules` C $[H_1; \ldots; H_n]$. When dealing with proof-search based decision, infinite horizontal branching of proof-search is usually forbidden. Hence, for a given C : `stm`, 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 `rules` *has finite inverse images*.[8]

$$\frac{H_1 \ \ldots \ H_n}{C}$$

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. 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 which satisfies Curry's lemma 1: 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 binary 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 for $\mathbf{LR2}_{\to}$, 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 s_0.[9] By using the constructive version of König's lemma of Section 3.2, 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 only. 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 set of n_0-bounded proofs of s_0 is empty or not.

[8] 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.

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

4.2 Finiteness, trees and branches

In this section, we consider a fixed X : Type. The predicate $\texttt{finite}_\texttt{t}\ P$ expresses that a sub-type $P : X \to \texttt{Prop}$ of X is finite and computable into a list:

$$\texttt{Definition finite}_\texttt{t}\ (P : X \to \texttt{Prop}) := \{l : \texttt{list}\ X \mid \forall x, x \in_1 l \iff P\ x\}.$$

We will use this predicate to encode the rules_fin property. We will also need a type of finitely branching (oriented) trees:

$$\texttt{Inductive tree}\ X := \texttt{in\_tree} : X \to \texttt{list}\,(\texttt{tree}\ X) \to \texttt{tree}\ X.$$

where we denote $\langle x|l\rangle$ for (in_tree x l). The root $\texttt{root} : \texttt{tree}\ X \to X$ of a tree verifies $\texttt{root}\ \langle x|l\rangle = x$ and the height $\texttt{ht} : \texttt{tree}\ X \to \texttt{nat}$ of a tree verifies $\texttt{ht}\,\langle\_|l\rangle = 1 + \texttt{max}\,(\texttt{map ht}\ l)$. Branches of trees are represented as specific lists of elements of X. We inductively define a branch predicate:

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

s.t. the lists b which satisfy 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.

4.3 Proofs, minimal proofs and everywhere minimal proofs

We consider a type stm of logical statements and a collection of valid rule instances rules which has the finite inverse image property.

$$\begin{array}{ll}\texttt{Variables} & (\texttt{stm} : \texttt{Type})\ (\texttt{rules} : \texttt{stm} \to \texttt{list stm} \to \texttt{Prop}). \\ \texttt{Hypothesis} & \big(\texttt{rules\_fin} : \forall c : \texttt{stm},\ \texttt{finite}_\texttt{t}\,(\texttt{rules}\ c)\big).\end{array}$$

Hence, not only are there finitely many rule instances for a given conclusion c but the predicate rules_fin c contains ll_c : list (list stm), an *effective list* of those instances which verifies the property:

$$[h_1; \ldots; h_n] \in_1 ll_c \iff \texttt{rules}\ c\ [h_1; \ldots; h_n] \quad \text{for any } [h_1; \ldots; h_n] : \texttt{list stm}$$

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 notion of proof is based on that of *proof tree*. We define a predicate proof_tree that satisfies the below (recursive) characteristic property:

$$\begin{array}{l}\texttt{Definition proof\_tree} : \texttt{tree stm} \to \texttt{Prop}. \\ \forall s\ l, \texttt{proof\_tree}\ \langle s|l\rangle \iff \texttt{rules}\ s\ (\texttt{map root}\ l) \wedge \forall t, t \in_1 l \to \texttt{proof\_tree}\ t.\end{array}$$

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 : stm) (t : tree stm) := proof_tree t ∧ root t = s.
Definition bproof (n : nat) (s : stm) (t : tree stm) := 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 : nat) (s : stm) : finite_t (bproof n s).
```

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

```
Definition   emin_ptree : tree stm → Prop.
 ∀s l, emin_ptree ⟨s|l⟩ ⟺ min_proof s ⟨s|l⟩ ∧ ∀t, t ∈₁ l → emin_ptree t.
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. □

4.4 The completeness of irredundant proofs via Curry's lemma

We assume a notion of redundancy on statements, that is a binary relation $\prec_r$: `stm`→`stm`→`Prop`. A list of statements l : `list stm` is redundant if it contains a good pair for $\prec_r$, which is denoted by `good` $\prec_r$ l (see Section 3.1). The list l is *irredundant* if it contains no good pair, i.e. `bad` $\prec_r$ l. A tree t : `tree stm` is an *irredundant proof* if it is a proof and every branch of the tree is irredundant.[10]

```
Definition irred_proof s t := proof s t ∧ ∀b, branch t b → bad ≺_r (rev b).
```

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

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

[10] Branches are read from the root to leaves, hence the use of `rev` to reverse lists.

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. Let $s_1 \prec_{\mathrm{r}} s_2$ be a good pair in b and let t_1/t_2 be the sub (proof) trees of roots s_1/s_2. As s_2 occurs after s_1 in b, 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 $\texttt{ht}\ t_1' \leqslant \texttt{ht}\ t_2$. 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. □

As a consequence, every proof can be transformed into an irredundant one by direct combination of `proof_eminimize` and `proof_emin_irred`.

$\texttt{Theorem proof\_reduce}\ s\ t : \texttt{proof}\ s\ t \to \{t_{\mathrm{irr}} \mid \texttt{irred\_proof}\ s\ t_{\mathrm{irr}}\}.$

4.5 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. 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 Section 3.2.

To capture the *sub-formula property* in our setting, we assume an abstract notion of sub-statement denoted by $s_1 \sqsupseteq_{\mathrm{sf}} s_2$ and which intuitively reads as the sub-formulæ of s_2 are also sub-formulæ of s_1. We postulate that $\sqsupseteq_{\mathrm{sf}}$ is both reflexive (`sf_refl`) and transitive (`sf_trans`) and more importantly, that every rule instance preserves sub-statements bottom-up (`sf_rules`):

$$\begin{aligned}\texttt{Variables}\ & (\sqsupseteq_{\mathrm{sf}} : \texttt{stm} \to \texttt{stm} \to \texttt{Prop})\ (\texttt{sf\_refl} : \forall s,\ s \sqsupseteq_{\mathrm{sf}} s)\\ & (\texttt{sf\_trans} : \forall s_1\ s_2\ s_3,\ s_1 \sqsupseteq_{\mathrm{sf}} s_2 \to s_2 \sqsupseteq_{\mathrm{sf}} s_3 \to s_1 \sqsupseteq_{\mathrm{sf}} s_3)\\ & (\texttt{sf\_rules} : \forall c\ l,\ \texttt{rules}\ c\ l \to \forall s, s \in_1 l \to c \sqsupseteq_{\mathrm{sf}} s).\end{aligned}$$

Starting from an initial statement $s_0 : \texttt{stm}$, 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

$$\begin{aligned}&\texttt{Let rules\_next} : \texttt{list stm} \to \texttt{list stm}.\\ &\quad \forall l\ h,\ h \in_1 \texttt{rules\_next}\ l \iff \exists c\ m,\ c \in_1 l \wedge h \in_1 m \wedge \texttt{rules}\ c\ m.\end{aligned}$$

i.e. `rules_next` l is the (finite) inverse image of l by valid rules instances. By `sf_rules`, the proof-search sequence is composed of sub-statements of s_0:

$\texttt{Proposition proof\_search\_sf}\ s_0\ n\ s : s \in_1 \texttt{rules\_next}^n\ [s_0] \to s_0 \sqsupseteq_{\mathrm{sf}} s.$

We can cover all the proof-search branches of length n starting from s_0 using the choices sequences over the proof-search sequence $\texttt{fun}\ n \mapsto \texttt{rules\_next}^n\ [s_0]$. Indeed, we establish the following covering property:

$$\begin{aligned}&\texttt{Let}\quad \texttt{FAN}\ n\ s_0 := \texttt{list\_fan}\left[\texttt{rules\_next}^{n-1}\ [s_0]; \ldots; \texttt{rules\_next}^0\ [s_0]\right].\\ &\texttt{Lemma ptree\_proof\_search}\ (t : \texttt{tree stm})\ (b : \texttt{list stm}) :\\ &\quad \texttt{branch}\ t\ b \to \texttt{proof\_tree}\ t \to \texttt{rev}\ b \in_1 \texttt{FAN}\ (\texttt{length}\ b)\ (\texttt{root}\ t).\end{aligned}$$

$$\begin{aligned}
&\texttt{Theorem proof_decider } (\texttt{stm} : \texttt{Type}) \ (\texttt{rules} : \texttt{stm} \to \texttt{list stm} \to \texttt{Prop}) \\
&\qquad (\texttt{rules_fin} : \big(\forall c, \texttt{finite}_\texttt{t}\ (\texttt{rules}\ c)\big)) \\
&\qquad (\sqsupseteq_\texttt{sf} : \texttt{stm} \to \texttt{stm} \to \texttt{Prop}) \ (\texttt{sf_refl} : \forall s, s \sqsupseteq_\texttt{sf} s) \\
&\qquad (\texttt{sf_trans} : \forall\, r\ s\ t,\ r \sqsupseteq_\texttt{sf} s \to s \sqsupseteq_\texttt{sf} t \to r \sqsupseteq_\texttt{sf} t) \\
&\qquad (\texttt{sf_rules} : \forall\, c\ l,\ \texttt{rules}\ c\ l \to \forall h,\ h \in_1 l \to c \sqsupseteq_\texttt{sf} h) \\
&\qquad (\prec_\texttt{r} : \texttt{stm} \to \texttt{stm} \to \texttt{Prop}) \\
&\qquad (\texttt{Curry} : \forall\, s\ t\ p,\ \texttt{pf}\ t\ p \to s \prec_\texttt{r} t \to \exists q,\ \texttt{pf}\ s\ q \wedge \texttt{ht}\, q \leqslant \texttt{ht}\, p) \\
&\qquad \big(\texttt{Kripke} : \forall s_0, \texttt{af}_\texttt{t}\ (\prec_\texttt{r} \texttt{restr}\ (\texttt{fun}\ s \mapsto s_0 \sqsupseteq_\texttt{sf} s))\big) : \\
&\quad \forall s_0 : \texttt{stm},\ \big\{p : \texttt{tree stm} \mid \texttt{pf}\ s_0\ p\big\} + \big\{\forall p : \texttt{tree stm},\ \neg(\texttt{pf}\ s_0\ p)\big\}.
\end{aligned}$$

Fig. 3. Constructive decider by redundancy-free proof-search ($\texttt{pf} := \texttt{proof rules}$).

Beware that this fan is a strict upper-approximation of proof-search branches.

We postulate our *redundancy hypothesis* denoted `Kripke` which states that the relation $\prec_\texttt{r}$ 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 Section 3.2 (`Constructive_Koenigs_lemma`), we derive:

$$\begin{aligned}
&\texttt{Hypothesis}\quad \texttt{Kripke} : \forall s_0 : \texttt{stm},\ \texttt{af}_\texttt{t}\ \big(\prec_\texttt{r} \texttt{restr}\ (\texttt{fun}\ s \mapsto s_0 \sqsupseteq_\texttt{sf} s)\big). \\
&\texttt{Proposition}\ \texttt{irredundant_max_length}\ (s_0 : \texttt{stm}) : \\
&\qquad \{n_0 : \texttt{nat} \mid \forall\, m\, l,\ n_0 \leqslant m \to l \in_1 \texttt{FAN}\ m\ s_0 \to \texttt{good} \prec_\texttt{r} l\}.
\end{aligned}$$

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

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

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

4.6 The constructive decider based on redundancy-free proof-search

The proof decider follows trivially. Indeed, the corresponding algorithm uses `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 `bproof_finite_t`. If that list is non-empty, then 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), and then there is no proof for s_0 at all using `proof_reduce`. The full abstract result `proof_decider` is displayed in Fig. 3 and established in the file proof.v.

We instantiate the `proof_decider` on the $\mathbf{LR2}_{\to}$ sequent calculus in the file relevant_LR2_dec.v:

$$\begin{aligned}
&\texttt{Theorem LR2_decider}\ (s : \texttt{Seq}) : \\
&\qquad \big\{t \mid \texttt{proof LR2_rules}\ s\ t\big\} + \big\{\forall t,\ \neg(\texttt{proof LR2_rules}\ s\ t)\big\}.
\end{aligned}$$

Using soundness and completeness results between $\mathbf{R}_{\rightarrow} \leftrightsquigarrow \mathbf{LR1}_{\rightarrow} \leftrightsquigarrow \mathbf{LR2}_{\rightarrow}$ (see the summary file relevant_equiv.v), we get the constructive decider for $\mathbf{R}_{\rightarrow}$ specified in Section 2.1, the proof of which can be found in logical_deciders.v:

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

5 Conclusion and Perspectives

We present an abstract and constructive view of Kripke-Curry's method for deciding Implicational Relevance Logic $\mathbf{R}_{\rightarrow}$. We get an axiom-free Coq implementation [14] that we instantiate on $\mathbf{LR2}_{\rightarrow}$ to derive a constructive decider for $\mathbf{R}_{\rightarrow}$. Although not presented in this paper, our implementation includes a constructive decider for implicational intuitionistic logic $\mathbf{J}_{\rightarrow}$ which shares the same language for formulæ as $\mathbf{R}_{\rightarrow}$. It is based on a variant of Gentzen's sequent calculus **LJ**. Unlike what happens which richer fragments of Relevance Logic [22], extensions of this method to full propositional **IL** would present no difficulty.

From a complexity perspective, Kripke's decidability proof for $\mathbf{R}_{\rightarrow}$ 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 that via control functions nor Dickson's lemma.

Kripke-Curry's method has a potential use well beyond $\mathbf{R}_{\rightarrow}$ 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 $\mathbf{LR1}_{\rightarrow}$, contraction (and weakening) cannot be completely removed from the bunched sequent calculus **LBI**. It is not obvious what notion of redundancy should be used in that case.

Analyzing the "glitches" in the decidability proof of ticket entailment [6] is another obvious perspective of this work. Indeed, the attempt of [2] is also based on Kripke-Curry's method. This decidability result was independently obtained by V. 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. Theoret. 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: Category Theory and Computer Science. pp. 164–172. Springer (1991)

4. Coquand, T.: Constructive topology and combinatorics. In: Constructivity in Computer Science. pp. 159–164. Springer (1992)
5. Curry, H.B.: A Theory of Formal Deductibility. Notre Dame mathematical lectures, University of 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 (2017)
7. Demri, S., Jurdziński, M., Lachish, O., Lazić, R.: The covering and boundedness problems for branching vector addition systems. J. Comput. System Sci. 79(1), 23–38 (2012)
8. Figueira, D., Figueira, S., Schmitz, S., Schnoebelen, P.: Ackermannian and Primitive-Recursive Bounds with Dickson's Lemma. In: LICS 2011. pp. 269–278 (2011)
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)
10. Fridlender, D.: An Interpretation of the Fan Theorem in Type Theory. In: TYPES'98 Selected Papers. LNCS, vol. 1657, pp. 93–105. Springer (1999)
11. Galmiche, D., Méry, D., Pym, D.: The semantics of BI and resource tableaux. Math. Structures Comput. Sci. 15(6), 1033–1088 (2005)
12. Kripke, S.: The Problem of Entailment (abstract). J. Symb. Log. 24, 324 (1959)
13. Larchey-Wendling, D.: A mechanized inductive proof of Kruskal's tree theorem. `http://www.loria.fr/~larchey/Kruskal` (2015)
14. Larchey-Wendling, D.: A Constructive Mechanization of Kripke-Curry's method for the Decidability of Implicational Relevance Logic. `https://github.com/DmxLarchey/Relevant-decidability` (2018)
15. Meyer, R.K.: Improved Decision Procedures for Pure Relevant Logic, pp. 191–217. Springer Netherlands, Dordrecht (2001)
16. Padovani, V.: Ticket Entailment is decidable. Math. Structures Comput. Sci. 23(3), 568–607 (2013)
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)
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. Urquhart, A.: The Undecidability of Entailment and Relevant Implication. J. Symb. Log. 49(4), 1059–1073 (1984)
23. Veldman, W., Bezem, M.: Ramsey's Theorem and the Pigeonhole Principle in Intuitionistic Mathematics. J. Lond. Math. Soc. s2-47(2), 193–211 (1993)
24. 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)