

# A general proof certification framework for modal logic

TOMER LIBAL<sup>†</sup> and MARCO VOLPE<sup>‡</sup>

<sup>†</sup>*The American University of Paris, Paris, France*

*Email: tlibal@aup.edu*

<sup>‡</sup>*Fortiss GmbH, Munich, Germany*

*Email: volpe@fortiss.org*

*Received 30 June 2017; revised 26 November 2018*

One of the main issues in proof certification is that different theorem provers, even when designed for the same logic, tend to use different proof formalisms and produce outputs in different formats. The project ProofCert promotes the usage of a common specification language and of a small and trusted kernel in order to check proofs coming from different sources and for different logics. By relying on that idea and by using a classical focused sequent calculus as a kernel, we propose here a general framework for checking modal proofs. We present the implementation of the framework in a Prolog-like language and show how it is possible to specialize it in a simple and modular way in order to cover different proof formalisms, such as labelled systems, tableaux, sequent calculi and nested sequent calculi. We illustrate the method for the logic  $K$  by providing several examples and discuss how to further extend the approach.

## 1. Introduction

The main difficulty in having general and comprehensive approaches to proof checking and proof certification derives from the fact that proof evidences, even for a single, specific logic, are produced by using several different proof formalisms and proof calculi. This is the case both for human-generated proofs and for proofs provided by automated theorem provers, which moreover tend to produce outputs in different formats. Addressing such an issue is one of the goals of the project ProofCert (Miller 2011). By using well-established concepts of proof theory, ProofCert proposes *foundational proof certificates* (FPC) as a framework to specify proof evidence formats. Describing a format in terms of an FPC allows software to check proofs in this format over a small kernel.

Checkers (Chihani et al. 2015) is a generic proof certifier based on the ProofCert ideas. It allows for the certification of arbitrary proof evidences using various trusted kernels, like the focused classical sequent calculus  $LKF$  (Liang and Miller 2009). Such kernels are enriched with additional predicates, which allow more control on the construction of a proof. Dedicated FPC specifications can be defined, over these predicates, in order to interpret the information coming from a specific proof evidence format, so that the kernel is forced to produce a proof that mirrors, and thus certifies in case of success, the original one.

Different kernels, though, offer different levels of confidence in the correctness of the proof. An important quality of a kernel is that it is as small as possible. The idea behind it – called the ‘de Bruijn Criterion’ (de Bruijn 1970) – is that small and simple kernels offer higher trust. The kernel employed in this paper, based on *LKF*, consists in 93 lines of  $\lambda$ Prolog code and is the same used in other ProofCert publications (e.g., Chihani et al. 2015, 2017; Libal and Volpe 2016)<sup>†</sup>.

The problem of the great variety of different proof formalisms and proof systems to be considered, when dealing with proof checking, is especially apparent in the case of modal logics, whose proof theory is notoriously non-trivial. In fact, in the last decades, several proposals have been provided (a general account is, e.g., in Fitting (2007)). Such proposals range over a set of different proof formalisms (e.g., sequent, nested sequent, labelled sequent, hypersequent calculi and semantic tableaux), each of them presenting its own features and drawbacks. Several results concerning correspondences and connections between the different formalisms are also known (Fitting 2012; Goré and Ramanayake 2012; Lellmann 2015).

In Marin et al. (2016), a general framework for emulating and comparing existing modal proof systems has been presented. Such a framework is based on the setting of *labelled deduction systems* (Gabbay 1996), which consists in enriching the syntax of modal logic with elements coming from the semantics, i.e., with elements referring explicitly to the worlds of a Kripke model and to the accessibility relation between such worlds. In particular, the framework is designed as a focused version of Negri’s system *G3K* (Negri 2005), further enriched with a few parametric devices. Playing with such parameters produces concrete instantiations of the framework, which, by exploiting the expressiveness of the labelled approach and the control mechanisms of focusing, can be used to emulate the behaviour of a range of existing formalisms and proof systems for modal logic with high precision.

In this paper, we rely on the close relationship between labelled sequent systems and *LKF* (Miller and Volpe 2015) in order to propose an implementation of such a framework that uses *LKF* as a kernel, and is developed as a module of the more general Checkers implementation project. This work also capitalizes on (and, in a sense, generalizes) the one in Libal and Volpe (2016), which was limited to the case of prefixed tableaux (*PT*). The implementation is extremely modular and based on the use of layers that mirror quite closely the instantiations of the framework presented in Marin et al. (2016). Concretely, we are able to certify, via this implementation, proofs given in the formalisms of labelled sequents, *PT*, ordinary sequent (*OS*) systems and nested sequents. We cover for the moment only the modal logic *K*, but the modularity of the approach should allow for an easy extension to other modal logics, in particular, those whose Kripke frames are defined by geometric axioms, according to the treatment described in Marin et al. (2016). Extension to other formalisms seems also possible; we discuss this in more detail in the conclusion.

An approach related to ours is in Benzmueller and Woltzenogel Paleo (2015), where the authors present a technique to generate and certify modal proofs using the Coq proof

<sup>†</sup> When calculating the size of the program, we placed each atomic predicate on a new line.

assistant. The aim of their work is to allow interactive theorem proving over higher order modal logics. To this end, they encode the semantics of higher order modal logics into the system used by Coq – the Calculus of Inductive Constructions. Their work and ours are similar in that they both certify modal logic proofs by using trusted kernels, but they also differ in several ways. Their system targets higher order modal logics and is also directed towards interactive theorem proving, while ours is for the moment restricted to the task of certification for propositional modal logic. On the other hand, while their encoding focuses on one specific proof format and calculus, we aim, via our framework, at supporting different formalisms and proof systems.

We proceed as follows. In Section 2, we present some background on ProofCert, modal logic and proof systems for modal logic. In Section 3, we recall the general framework of Marin et al. (2016). In Section 4, we describe its implementation, by presenting the FPC specifications of the different layers and by providing a few examples. In Section 5, we discuss possible directions for future work, compare with some related approaches, and conclude.

## 2. Background

### 2.1. Proof systems for modal logic

In this section, we review several proof systems that are among the most popular calculi (Fitting 2007) for automated theorem proving in modal logic as well as for manual proof generation. Before that, we recall a few key notions about modal logic and its relation with first-order classical logic.

We remark that throughout this paper, we will work with formulas in *negation normal form*, i.e., such that only atoms may possibly occur negated in them. Notice that this is not a restriction, as it is always possible to convert a propositional formula into an equivalent formula in negation normal form (both in classical and in modal logic).

**2.1.1. Modal logic.** The language of (*propositional*) *modal formulas* consists of a functionally complete set of classical propositional connectives, a *modal operator*  $\Box$  (here we will also use explicitly its dual  $\Diamond$ ) and a denumerable set  $\mathcal{P}$  of *propositional symbols*. The grammar is specified as follows:

$$A ::= P \mid \neg P \mid A \vee A \mid A \wedge A \mid \Box A \mid \Diamond A$$

where  $P \in \mathcal{P}$ . We say that a formula is a  $\Box$ -*formula* ( $\Diamond$ -*formula*) if its main connective is  $\Box$  ( $\Diamond$ ). The semantics of the modal logic  $K$  is usually defined by means of *Kripke frames*, i.e., pairs  $\mathcal{F} = (W, R)$  where  $W$  is a non-empty set of *worlds* and  $R$  is a binary relation on  $W$ . We say that a world  $w$  is *accessible from a world*  $w'$  iff  $(w, w') \in R$ . A *Kripke model* is a triple  $\mathcal{M} = (W, R, V)$  where  $(W, R)$  is a Kripke frame and  $V : W \rightarrow 2^{\mathcal{P}}$  is a function that assigns to each world in  $W$  a (possibly empty) set of propositional symbols.

In the basic modal logic  $K$ , we define the *truth* of a modal formula at a world  $w$  in a Kripke model  $\mathcal{M} = (W, R, V)$  as the smallest relation  $\models$  satisfying

$$\begin{aligned} \mathcal{M}, w \models P & \text{ iff } P \in V(w) \\ \mathcal{M}, w \models \neg P & \text{ iff } P \notin V(w) \\ \mathcal{M}, w \models A \vee B & \text{ iff } \mathcal{M}, w \models A \text{ or } \mathcal{M}, w \models B \\ \mathcal{M}, w \models A \wedge B & \text{ iff } \mathcal{M}, w \models A \text{ and } \mathcal{M}, w \models B \\ \mathcal{M}, w \models \Box A & \text{ iff } \mathcal{M}, w' \models A \text{ for all } w' \text{ s.t. } wRw' \\ \mathcal{M}, w \models \Diamond A & \text{ iff there exists } w' \text{ s.t. } wRw' \text{ and } \mathcal{M}, w' \models A. \end{aligned}$$

By extension, we write  $\mathcal{M} \models A$  when  $\mathcal{M}, w \models A$  for all  $w \in W$  and we write  $\models A$  when  $\mathcal{M} \models A$  for every Kripke model  $\mathcal{M}$ .

2.1.2. *The standard translation from modal logic into classical logic.* The following *standard translation* (see, e.g., Blackburn and Van Benthem 2007) provides a bridge between propositional (classical) modal logic and first-order classical logic:

$$\begin{aligned} ST_x(P) &= P(x) & ST_x(A \wedge B) &= ST_x(A) \wedge ST_x(B) \\ ST_x(\neg P) &= \neg P(x) & ST_x(\Box A) &= \forall y(R(x, y) \supset ST_y(A)) \\ ST_x(A \vee B) &= ST_x(A) \vee ST_x(B) & ST_x(\Diamond A) &= \exists y(R(x, y) \wedge ST_y(A)) \end{aligned}$$

where  $x$  is a free variable denoting the world in which the formula is being evaluated. The first-order language into which modal formulas are translated is usually referred to as *first-order correspondence language* (Blackburn and Van Benthem 2007) and consists of a binary predicate symbol  $R$  and a unary predicate symbol  $P$  for each  $P \in \mathcal{P}$ . When a modal operator is translated, a new fresh variable is introduced<sup>‡</sup>. It is easy to show that for any modal formula  $A$ , any model  $\mathcal{M}$  and any world  $w$ , we have that  $\mathcal{M}, w \models A$  if and only if  $\mathcal{M} \models ST_x(A)[x \leftarrow w]$ .

2.1.3. *Labelled sequent systems.* Several different deductive formalisms have been used for modal proof theory and theorem proving. One of the most interesting approaches has been presented in Gabbay (1996) with the name of labelled deduction. The basic idea behind labelled proof systems for modal logic is to internalize elements of the corresponding Kripke semantics (namely, the worlds of a Kripke model and the accessibility relation between such worlds) into the syntax. A concrete example of such a system is the sequent calculus  $G3K$  presented in Negri (2005) (we present it here in a single-sided formulation and refer to it as  $LS$ ).  $LS$  formulas are either *labelled formulas* of the form  $x : A$  or *relational atoms* of the form  $xRy$ , where  $x, y$  range over a set of variables and  $A$  is a modal formula. In the following, we will use  $\varphi, \psi$  to denote  $LS$  formulas.  $LS$  sequents have the form  $\mathcal{G} \vdash \Delta$ , where  $\Delta$  is a multiset containing labelled formulas and  $\mathcal{G}$  is a set of relational atoms. Being  $LS$  a labelled system, we say that  $A$  is provable in  $LS$  if there is

<sup>‡</sup> In fact, it is possible to show that every modal formula can be translated into a formula in the fragment of first-order logic which uses only two variables (Blackburn and Van Benthem 2007). By the decidability of such a fragment, an easy proof of the decidability of the modal logic  $K$  follows.

CLASSICAL RULES

$$\frac{}{\mathcal{G} \vdash \Delta, x : \neg P, x : P} \text{init}_{LS} \quad \frac{\mathcal{G} \vdash \Delta, x : A \quad \mathcal{G} \vdash \Delta, x : B}{\mathcal{G} \vdash \Delta, x : A \wedge B} \wedge_{LS} \quad \frac{\mathcal{G} \vdash \Delta, x : A, x : B}{\mathcal{G} \vdash \Delta, x : A \vee B} \vee_{LS}$$

MODAL RULES

$$\frac{\mathcal{G} \cup \{xRy\} \vdash \Delta, y : A}{\mathcal{G} \vdash \Delta, x : \Box A} \Box_{LS} \quad \frac{\mathcal{G} \cup \{xRy\} \vdash \Delta, x : \Diamond A, y : A}{\mathcal{G} \cup \{xRy\} \vdash \Delta, x : \Diamond A} \Diamond_{LS}$$

In  $\Box_{LS}$ ,  $y$  does not occur in the conclusion.

Fig. 1. *LS*: a labelled sequent system for the modal logic *K*.

CLASSICAL RULES

$$\frac{\sigma : A \wedge B}{\sigma : A, \sigma : B} \wedge_{PT} \quad \frac{\sigma : A \vee B}{\sigma : A \mid \sigma : B} \vee_{PT}$$

MODAL RULES

$$\frac{\sigma : \Box A}{\sigma.n : A} \Box_{PT} \quad \frac{\sigma : \Diamond A}{\sigma.n : A} \Diamond_{PT}$$

In  $\Box_{PT}$ ,  $\sigma.n$  is used. In  $\Diamond_{PT}$ ,  $\sigma.n$  is new.

Fig. 2. *PT*: a prefixed tableau system for the modal logic *K*.

a proof of  $\vdash x : A$  for any variable  $x$ . In Figure 1, we present the rules of *LS*, which is proved to be sound and complete for the basic modal logic *K* (Negri 2005).

2.1.4. *Prefixed tableau systems.* *PT* can also be seen as a particular kind of labelled deductive system. They were introduced in Fitting (1972), although the formulation that we use here is closer to the one in Fitting (2007). Differently from *LS*, *PT* are refutation proof systems, i.e., in order to prove a formula, we negate it and derive from it a contradiction. A *prefix* is a finite sequence of positive integers (written by using dots as delimiters). Intuitively, prefixes denote possible worlds and they are such that if  $\sigma$  is a prefix, then  $\sigma.1$  and  $\sigma.2$  denote two worlds accessible from  $\sigma$ . A *prefixed formula* is  $\sigma : A$ , where  $\sigma$  is a prefix and  $A$  is a modal formula in negation normal form. A prefixed tableau proof of  $A$  starts with a root node containing  $1 : A$ , informally asserting that  $A$  is false in the world named by the prefix 1. It continues by using the branch extension rules given in Figure 2. We say that a branch of a tableau is a *closed branch* if it contains  $\sigma : P$  and  $\sigma : \neg P$  for some  $\sigma$  and some  $P$ . The goal is to produce a *closed tableau*, i.e., a tableau such that all its branches are closed. Classical rules in Figure 2 are the prefixed version of the standard ones. For what concerns the modal rules, the  $\Diamond$  rule applied to a formula  $\sigma : \Diamond A$  intuitively allows for generating a new world, accessible from  $\sigma$ , where  $A$  holds, while the  $\Box$  rule applied to a formula  $\sigma : \Box A$  allows for moving the formula  $A$  to an already existing world accessible from  $\sigma$ .

2.1.5. *Ordinary sequent systems.* Several ‘ordinary’ sequent systems have been proposed in the literature for different modal logics (a general account is, e.g., in Indrzejczak (2010) and Poggiolesi (2011)). In our treatment, we will use the formalization *OS* presented in Figure 3, which is adapted from the presentations in Fitting (2007) and Stewart and

CLASSICAL RULES

$$\frac{}{\vdash \Gamma, P, \neg P} \text{init}_{OS} \qquad \frac{\vdash \Gamma, A \quad \vdash \Gamma, B}{\vdash \Gamma, A \wedge B} \wedge_{OS} \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \vee B} \vee_{OS}$$

MODAL RULES

$$\frac{\vdash \Gamma, A}{\vdash \diamond \Gamma, \square A, \Delta} \square_{OS}$$

Fig. 3. OS: an ordinary sequent system for the modal logic K.

CLASSICAL RULES

$$\frac{}{\mathcal{N}\{P, \neg P\}} \text{init}_{NS} \qquad \frac{\mathcal{N}\{A\} \quad \mathcal{N}\{B\}}{\mathcal{N}\{A \wedge B\}} \wedge_{NS} \qquad \frac{\mathcal{N}\{A, B\}}{\mathcal{N}\{A \vee B\}} \vee_{NS}$$

MODAL RULES

$$\frac{\mathcal{N}\{[A]\}}{\mathcal{N}\{\square A\}} \square_{NS} \qquad \frac{\mathcal{N}\{\diamond A, [A, \mathcal{M}]\}}{\mathcal{N}\{\diamond A, [\mathcal{M}]\}} \diamond_{NS}$$

Fig. 4. NS: a nested sequent system for the modal logic K.

Stouppa (2004). The base classical system (consisting of *identity*, *structural* and *classical connective* rules) is extended by a modal rule that, works on one  $\square$ -formula and several  $\diamond$ -formulas, bottom-up.

2.1.6. *Nested sequent systems.* Nested sequents (introduced by Kashima (1994), and then independently rediscovered by Poggiolesi (2011), as *tree-hypersequents*, and by Br unnler (2009)) are an extension of OS to a structure of tree, where each  $[]$ -node represents the scope of a modal  $\square$ . We write a nested sequent as a multiset of formulas and *boxed sequents*, according to the following grammar, where  $A$  can be any modal formula in negative normal form:  $\mathcal{N} ::= \emptyset \mid A, \mathcal{N} \mid [\mathcal{N}], \mathcal{N}$ .

In a nested sequent calculus, a rule can be applied at any depth in this tree structure, that is, inside a certain nested sequent context. A *context* written as  $\mathcal{N}\{ \} \cdots \{ \}$  is a nested sequent with a number of holes occurring in place of formulas (and never inside a formula). Given a context  $\mathcal{N}\{ \} \cdots \{ \}$  with  $n$  holes, and  $n$  nested sequents  $\mathcal{M}_1, \dots, \mathcal{M}_n$ , we write  $\mathcal{N}\{\mathcal{M}_1\} \cdots \{\mathcal{M}_n\}$  to denote the nested sequent where the  $i$ th hole in the context has been replaced by  $\mathcal{M}_i$ , with the understanding that if  $\mathcal{M}_i = \emptyset$  then the hole is simply removed. We are going to consider the nested sequent system (in Figure 4) introduced by Br unnler in Br unnler (2009), that we call here NS.

2.2. A general proof checker

There is no consensus about what shape a formal proof evidence should take. The notion of structural proofs, which is based on derivations in some calculus, is of no help as long as the calculus is not fixed. One of the ideas of the ProofCert project is to try to amend this problem by defining the notion of a FPC as a pair of an arbitrary proof evidence and an executable specification which denotes its semantics in terms of some well-known target calculus, such as the sequent calculus. These two elements of an FPC are then given to a

universal proof checker which, by the help of the FPC, is capable of deriving a proof in the target calculus. Since the proof generated is over a well-known and low-level calculus which is easy to implement, one can obtain a high degree of trust in its correctness. Such an approach seems to be applicable to a large class of proof formalisms.

The proof certifier *Checkers* is a  $\lambda$ Prolog (Miller 2012) implementation of this idea. Its main components are the following:

- **Kernel.** The kernels are the implementations of several trusted proof calculi. Currently, there are kernels over the classical and intuitionistic focused sequent calculus. Section 2.3 is devoted to the presentation of *LKF*, i.e., the classical focused sequent calculus that will be used in the paper.
- **Proof evidence.** The first component of an FPC, a proof evidence is a  $\lambda$ Prolog description of a proof output of a theorem prover. Given the high-level declarative form of  $\lambda$ Prolog, the structure and form of the evidence are very similar to the original proof. We specify the form of the different proof evidences we handle in Section 4.
- **FPC specification.** The basic idea of *Checkers* is to try and generate a proof of the theorem of the evidence in the target kernel. In order to achieve that, the different kernels have additional predicates which take into account the information given in the evidence. Since the form of this information is not known to the kernel, *Checkers* uses FPC specifications in order to interpret it. These logical specifications are written in  $\lambda$ Prolog and interface with the kernel in a sound way in order to certify proofs. Writing these specifications is the main task for supporting the different outputs of the modal theorem provers we consider in this paper and they are, therefore, explained in detail in Section 4. We mention here the existence of two different types of specifications. The *clerks*, which simply perform some bookkeeping computations without using any information from the evidence, and the *experts*, which, in addition, also use information from the evidence in order to guide the kernel with regard to choices to make.

### 2.3. A focused sequent calculus for classical logic

Theorem provers often use efficient but non-trivial proof calculi, possibly employing heuristics or optimization techniques, whose complexity leads to a lower degree of trust. On the other hand, traditional proof calculi, like the sequent calculus, enjoy a high degree of trust but are quite inefficient for proof search. In order to use the sequent calculus as the basis of automated deduction, much more structure within proofs needs to be established. Focused sequent calculi, first introduced by Andreoli Andreoli (1992) for linear logic, combine the higher degree of trust of sequent calculi with a more efficient proof search. They take advantage of the fact that some of the rules are ‘invertible,’ i.e., can be applied without requiring backtracking, and that some other rules can ‘focus’ on the same formula for a batch of deduction steps. In this paper, we will make use of the classical focused sequent calculus (*LKF*) system defined in Liang and Miller (2009). Figure 5 presents, in the black font, the rules of *LKF*.

Formulas in *LKF* which are expressed in negation normal form, can have either positive or negative polarity and are constructed from atomic formulas, whose polarity has to be assigned, and from logical connectives whose polarity is pre-assigned. The choice

of polarization does not affect the provability of a formula, but it can have a big impact on proof search and on the structure of proofs: one can observe, e.g., that in *LKF* the rule for  $\vee^-$  is invertible while the one for  $\vee^+$  is not. The connectives  $\wedge^-, \vee^-$  and  $\forall$  are of negative polarity, while  $\wedge^+, \vee^+$  and  $\exists$  are of positive polarity. A composed formula has the same polarity of its main connective. In order to polarize literals, we are allowed to fix the polarity of atomic formulas in any way we see fit. We may ask that all atomic formulas are positive, that they are all negative, or we can mix polarity assignments. In any case, if  $A$  is a positive atomic formula, then it is a positive formula and  $\neg A$  is a negative formula: conversely, if  $A$  is a negative atomic formula, then it is a negative formula and  $\neg A$  is a positive formula.

Deductions in *LKF* are done during synchronous or asynchronous phases. A synchronous phase, in which sequents have the form  $\vdash \Theta \Downarrow B$ , corresponds to the application of synchronous rules to a specific positive formula  $B$  under focus (and possibly its immediate positive subformulas). An asynchronous phase, in which sequents have the form  $\vdash \Theta \Uparrow \Gamma$ , consists in the application of invertible rules to negative formulas contained in  $\Gamma$  (and possibly their immediate negative subformulas). Phases can be changed by the application of the *release* rule. A *bipole* is a pair of a synchronous phase below an asynchronous phase within *LKF*: thus, bipoles are macro inference rules in which the conclusion and the premises are  $\Uparrow$ -sequents with no formulas to the right of the up-arrow.

It is useful sometimes to delay the application of invertible rules (focused rules) on some negative formulas (positive formulas)  $A$ . In order to achieve that, we define the following delaying operators  $\partial^+(A) = \text{true} \wedge^+ A$  and  $\partial^-(A) = \text{false} \vee^- A$ . Clearly,  $A, \partial^+(A)$  and  $\partial^-(A)$  are all logically equivalent but  $\partial^+(A)$  is always a positive formula and  $\partial^-(A)$  is always a negative one.

In order to integrate the use of FPC into the calculus, we enrich each rule of *LKF* with proof evidences and additional predicates, given in blue font in Figure 5. We call the resulted calculus *LKF<sup>a</sup>*. *LKF<sup>a</sup>* extends *LKF* in the following way. Each sequent now contains additional information in the form of the proof evidence  $\Xi$ . At the same time, each rule is associated with a predicate (for example, *initial<sub>e</sub>*( $\Xi, l$ )) which, according to the proof evidence, might prevent the rule from being called or guide it by supplying such information as the cut formula to be used.

Note that adding the FPC definitions in Figure 5 does not harm the soundness of the system but only restricts the possible rules which can be applied at each step. Therefore, a proof obtained using *LKF<sup>a</sup>* is also a proof in *LKF*. Since the additional predicates do not compromise the soundness of *LKF<sup>a</sup>*, we allow their definition to be external to the kernel and in fact these definitions, which are supplied by the user, are what allow *Checkers* to check arbitrary proof formats.

### 3. A general focused framework for modal logic

#### 3.1. A focused labelled calculus for modal logic

In Miller and Volpe (2015), a focused labelled sequent system (*LMF*) for the modal logic  $K$  has been presented. Such a calculus can be seen either as a focused version of *LS* or as



ASYNCHRONOUS INTRODUCTION RULES

$$\frac{\Xi' \vdash \Theta \uparrow A, \Gamma \quad \Xi'' \vdash \Theta \uparrow B, \Gamma \quad \text{andNeg}_e(\Xi, \Xi', \Xi'')}{\Xi \vdash \Theta \uparrow A \wedge^- B, \Gamma}$$

$$\frac{\Xi' \vdash \Theta \uparrow A, B, \Gamma \quad \text{orNeg}_e(\Xi, \Xi')}{\Xi \vdash \Theta \uparrow A \vee^- B, \Gamma} \quad \frac{(\Xi' y) \vdash \Theta \uparrow [y/x]B, \Gamma \quad \text{all}_e(\Xi, \Xi')}{\Xi \vdash \Theta \uparrow \forall x.B, \Gamma} \dagger$$

SYNCHRONOUS INTRODUCTION RULES

$$\frac{\Xi' \vdash \Theta \downarrow B_1 \quad \Xi'' \vdash \Theta \downarrow B_2 \quad \text{andPos}_e(\Xi, \Xi', \Xi'')}{\Xi \vdash \Theta \downarrow B_1 \wedge^+ B_2}$$

$$\frac{\Xi' \vdash \Theta \downarrow B_i \quad \text{orPos}_e(\Xi, \Xi', i)}{\Xi \vdash \Theta \downarrow B_1 \vee^+ B_2} \quad \frac{\Xi' \vdash \Theta \downarrow [t/x]B \quad \text{some}_e(\Xi, t, \Xi')}{\Xi \vdash \Theta \downarrow \exists x.B}$$

IDENTITY RULES

$$\frac{\Xi' \vdash \Theta \uparrow B \quad \Xi'' \vdash \Theta \uparrow \neg B \quad \text{cut}_e(\Xi, \Xi', \Xi'', B)}{\Xi \vdash \Theta \uparrow \cdot} \text{cut} \quad \frac{\langle l, \neg P_a \rangle \in \Theta \quad \text{initial}_e(\Xi, l)}{\Xi \vdash \Theta \downarrow P_a} \text{init}$$

STRUCTURAL RULES

$$\frac{\Xi' \vdash \Theta \uparrow N \quad \text{release}_e(\Xi, \Xi')}{\Xi \vdash \Theta \downarrow N} \text{release} \quad \frac{\Xi' \vdash \Theta, \langle l, C \rangle \uparrow \Gamma \quad \text{store}_e(\Xi, C, l, \Xi')}{\Xi \vdash \Theta \uparrow C, \Gamma} \text{store}$$

$$\frac{\Xi' \vdash \Theta \downarrow P \quad \langle l, P \rangle \in \Theta \quad \text{decide}_e(\Xi, l, \Xi')}{\Xi \vdash \Theta \uparrow \cdot} \text{decide}$$

Fig. 5. The proof system  $LKF^a$ , augmented version of  $LKF$ . Here,  $P$  is a positive formula;  $N$  a negative formula;  $P_a$  a positive literal;  $C$  a positive formula or negative literal; and  $\neg B$  is the negation normal form of the negation of  $B$ . The proviso marked  $\dagger$  requires that  $y$  is not free in  $\Xi, \Theta, \Gamma, B$ .  $LKF$  is obtained by ignoring the blue elements in the figure.

the restriction of  $LKF$  to the first-order correspondence language of Section 2.1.2 (where modalities are considered as synthetic connectives).

Figure 6 presents a multi-focused version (denoted  $LMF_m$ ) of the calculus, i.e., a variant where it is possible to focus on several positive formulas at the same time. Such a variant will also be considered in the rest of the paper.  $LMF$  can be read from the figure by ignoring the elements in blue font, or, equivalently, by imposing the condition that  $\Omega, \Omega_1$  and  $\Omega_2$  are empty in all rules.

In these systems, sequents have the form  $\mathcal{G} \vdash \Theta \downarrow \Gamma$  (with  $\Gamma$  containing exactly one formula in the case of  $LMF$ ) or  $\mathcal{G} \vdash \Theta \uparrow \Gamma$ , where the *relational set (of the sequent)*  $\mathcal{G}$  is a set of relational atoms and  $\Theta$  and  $\Gamma$  are multisets of labelled formulas.

### 3.2. A general framework for modal logic

In the context of modal logics, labelled proof systems have been shown to be quite expressive and encodings of other approaches into this formalism have also been presented in the literature (Fitting 2012; Goré and Ramanayake 2012; Lellmann 2015). It seems therefore quite natural to explore the possibility of reproducing the behaviour of modal proof systems based on different formalisms inside  $LMF$ , by exploiting at the same time

ASYNCHRONOUS INTRODUCTION RULES

$$\frac{\mathcal{G} \vdash \Theta \uparrow x : A, \Gamma \quad \mathcal{G} \vdash \Theta \uparrow x : B, \Gamma}{\mathcal{G} \vdash \Theta \uparrow x : A \wedge^- B, \Gamma} \wedge_K^- \quad \frac{\mathcal{G} \vdash \Theta \uparrow x : A, x : B, \Gamma}{\mathcal{G} \vdash \Theta \uparrow x : A \vee^- B, \Gamma} \vee_K^-$$

$$\frac{\mathcal{G} \cup \{xRy\} \vdash \Theta \uparrow y : B, \Gamma}{\mathcal{G} \vdash \Theta \uparrow x : \Box B, \Gamma} \Box_K$$

SYNCHRONOUS INTRODUCTION RULES

$$\frac{\mathcal{G} \vdash \Theta \Downarrow x : A, \Omega_1 \quad \mathcal{G} \vdash \Theta \Downarrow x : B, \Omega_2}{\mathcal{G} \vdash \Theta \Downarrow x : A \wedge^+ B, \Omega_1, \Omega_2} \wedge_K^+$$

$$\frac{\mathcal{G} \vdash \Theta \Downarrow x : B_i, \Omega}{\mathcal{G} \vdash \Theta \Downarrow x : B_1 \vee^+ B_2, \Omega} \vee^+, i \in \{1, 2\} \quad \frac{\mathcal{G} \cup \{xRy\} \vdash \Theta \Downarrow y : B, \Omega}{\mathcal{G} \cup \{xRy\} \vdash \Theta \Downarrow x : \Diamond B, \Omega} \Diamond_K$$

IDENTITY RULES

$$\frac{}{\mathcal{G} \vdash x : \neg B, \Theta \Downarrow x : B} \text{init}_K$$

STRUCTURAL RULES

$$\frac{\mathcal{G} \vdash \Theta, x : B \uparrow \Gamma}{\mathcal{G} \vdash \Theta \uparrow x : B, \Gamma} \text{store}_K \quad \frac{\mathcal{G} \vdash \Theta \uparrow x : B, \Omega}{\mathcal{G} \vdash \Theta \Downarrow x : B, \Omega} \text{release}_K \quad \frac{\mathcal{G} \vdash x : B, \Omega, \Theta \Downarrow x : B, \Omega}{\mathcal{G} \vdash x : B, \Omega, \Theta \uparrow \cdot} \text{decide}_K$$

In  $\text{decide}_K$ ,  $B$  and the formulas in  $\Omega$  are positive; in  $\text{release}_K$ ,  $B$  and the formulas in  $\Omega$  are negative; in  $\text{store}_K$ ,  $B$  is a positive formula or a negative literal; in  $\text{init}_K$ ,  $B$  is a positive literal. In  $\Box_K$ ,  $y$  is different from  $x$  and does not occur in  $\Theta, \Gamma, \mathcal{G}$ .

Fig. 6.  $LMF_m$ : a multi-focused labelled proof system for the modal logic  $K$ . The single-focused version  $LMF$  is obtained by forcing  $\Omega, \Omega_1$  and  $\Omega_2$  to be empty in all rules.

the expressivity of labelling and the control mechanisms provided by focusing. Such an analysis has been carried out in Marin et al. (2016) and has shown that, by enriching  $LMF$  with a few further technical devices, it is possible to get enough power to drive construction of proofs so as to emulate the proof structure of a wide range of formalisms.

The general framework  $LMF_*$  is presented in Figure 7. In the rest of this paper, when talking of  $LMF_*$  and its instantiations, a *labelled formula* will have the form  $\varphi \equiv x\sigma : A$ , where  $\sigma$  is either empty or a label  $y$ . We say that  $x$  is the *present* of  $\varphi$  and  $\sigma$  is the *future* of  $\varphi$ . Intuitively, the present of a formula has the usual role of labels in labelled systems, i.e., it refers to the world where the formula holds. The future, when present, is used to drive (bottom-up) the applications of the  $\Diamond$  introduction rule, i.e., it specifies the label to be used as a witness in the rule. An  $LMF_*$  *sequent* has the form  $\mathcal{G} \vdash_{\mathcal{H}} \Theta \Downarrow \Omega$  or  $\mathcal{G} \vdash_{\mathcal{H}} \Theta \uparrow \Omega$ , where  $\mathcal{G}$  is a set of relational atoms, the *present (of the sequent)*  $\mathcal{H}$  is a non-empty multiset of labels, and  $\Theta$  and  $\Omega$  are multisets of labelled formulas. Intuitively, the present of a sequent specifies which labels are currently ‘active,’ in the sense that when building a proof (bottom-up) a *decide* rule can only put the focus on labelled formulas whose present is contained in the present of the sequent. Like  $LMF_m$ ,  $LMF_*$  is a multi-focused system, as one can notice by observing that we do not necessarily have a single formula on the right of  $\Downarrow$ .

We refer the reader to Marin et al. (2016) for a more comprehensive explanation of the devices introduced in  $LMF_*$  with respect to  $LMF$  and  $LMF_m$ . We just remark that

ASYNCHRONOUS INTRODUCTION RULES

$$\frac{\mathcal{G} \vdash_{\mathcal{H}} \Theta \uparrow x : A, \Omega \quad \mathcal{G} \vdash_{\mathcal{H}} \Theta \uparrow x : B, \Omega}{\mathcal{G} \vdash_{\mathcal{H}} \Theta \uparrow x : A \wedge B, \Omega} \wedge_F \quad \frac{\mathcal{G} \vdash_{\mathcal{H}} \Theta \uparrow x : A, x : B, \Omega}{\mathcal{G} \vdash_{\mathcal{H}} \Theta \uparrow x : A \vee B, \Omega} \vee_F$$

$$\frac{\mathcal{G} \cup \{xRy\} \vdash_{\mathcal{H}} \Theta \uparrow y : B, \Omega}{\mathcal{G} \vdash_{\mathcal{H}} \Theta \uparrow x : \Box B, \Omega} \Box_F$$

SYNCHRONOUS INTRODUCTION RULES

$$\frac{\mathcal{G} \vdash_{\mathcal{H}} \Theta \Downarrow x\sigma : B_1, \Omega_1 \quad \mathcal{G} \vdash_{\mathcal{H}} \Theta \Downarrow x\sigma : B_2, \Omega_2}{\mathcal{G} \vdash_{\mathcal{H}} \Theta \Downarrow x\sigma : B_1 \wedge^+ B_2, \Omega_1, \Omega_2} \wedge_F^+$$

$$\frac{\mathcal{G} \vdash_{\mathcal{H}} \Theta \Downarrow x\sigma : B_i, \Omega}{\mathcal{G} \vdash_{\mathcal{H}} \Theta \Downarrow x\sigma : B_1 \vee^+ B_2, \Omega} \vee_F^+, i \in \{1, 2\} \quad \frac{\mathcal{G} \cup \{xRy\} \vdash_{\mathcal{H}} \Theta \Downarrow y : B, \Omega}{\mathcal{G} \cup \{xRy\} \vdash_{\mathcal{H}} \Theta \Downarrow xy : \Diamond B, \Omega} \Diamond_F$$

IDENTITY RULES

$$\frac{}{\mathcal{G} \vdash_{\mathcal{H}} x : \neg B, \Theta \Downarrow x : B} \text{init}_F$$

STRUCTURAL RULES

$$\frac{\mathcal{G} \vdash_{\mathcal{H}} \Theta, x : B \uparrow \Omega}{\mathcal{G} \vdash_{\mathcal{H}} \Theta \uparrow x : B, \Omega} \text{store}_F \quad \frac{\mathcal{G} \vdash_{\mathcal{H}} \Theta \uparrow \Omega'}{\mathcal{G} \vdash_{\mathcal{H}} \Theta \Downarrow \Omega} \text{release}_F \quad \frac{\mathcal{G} \vdash_{\mathcal{H}'} \Theta \Downarrow \Omega}{\mathcal{G} \vdash_{\mathcal{H}} \Theta \uparrow \cdot} \text{decide}_F$$

In  $\text{store}_F$ ,  $B$  is a positive formula or a negative literal.

In  $\text{init}_F$ ,  $B$  is a positive literal.

In  $\Box_F$ ,  $y$  is different from  $x$  and does not occur in  $\mathcal{G}$  nor in  $\Theta$ .

In  $\text{decide}_F$ , if  $xy : A \in \Omega$  then  $x : A \in \Theta$ . Moreover,  $\Omega$  contains only positive formulas of the form: (i)  $x\sigma : A$ , where  $A$  is not a  $\Diamond$ -formula and  $x \in \mathcal{H}$ ; or (ii)  $xy : A$  where  $A$  is a  $\Diamond$ -formula,  $xRy \in \mathcal{G}$ ,  $x \in \mathcal{H}$ .

In  $\text{release}_F$ ,  $\Omega$  contains no positive formulas and  $\Omega' = \{x : A \mid x\sigma : A \in \Omega\}$ .

Fig. 7.  $LMF_*$ : a focused labelled framework for the modal logic  $K$ .

the framework presented here is slightly different from the one proposed in Marin et al. (2016), since considering only the logic  $K$  allows for a few simplifications.

### 3.3. Emulation of modal proof systems

In order to emulate proofs given in other proof calculi by means of the focused framework  $LMF_*$ , we need first of all to define a translation from the original modal language to the polarized one.

When translating a modal formula into a polarized one, we are often in a situation where we are interested in putting a delay in front of the formula only in the case when it is negative and not a literal. For that purpose, we define  $A^{\delta^+}$ , where  $A$  is a modal formula in negation normal form, to be  $A$  if  $A$  is a literal or a positive formula and  $\delta^+(A)$  otherwise. We then define the translation  $[\cdot]$  as follows:

$$\begin{aligned} [P] &= P & [A \wedge B] &= [A]^{\delta^+} \wedge^- [B]^{\delta^+} \\ [\neg P] &= \neg P & [A \vee B] &= [A]^{\delta^+} \vee^- [B]^{\delta^+} \\ [\Box A] &= \Box([A]^{\delta^+}) & [\Diamond A] &= \Diamond(\delta^-([A]^{\delta^+})) \end{aligned}$$

In this translation, delays are used to ensure that only one connective of the original formula is processed along a given bipole of a focused derivation of the translated formula. This will be useful, in our proof checking procedure, in order to keep a tight connection between the original proof and the reconstructed one.

Finally, in order to emulate existing calculi in  $LMF_*$ , we need to give specialized versions of the rule  $decide_F$ . For  $LS$ , we do it as follows:

$$\frac{\mathcal{G} \vdash_{\mathcal{L}} \Theta \Downarrow x\sigma : A}{\mathcal{G} \vdash_{\mathcal{L}} \Theta \Uparrow} \text{decide}_{LS}$$

where

- $\mathcal{L}$  denotes the set of all labels;
- if  $A$  is a  $\diamond$  formula, then  $\sigma$  is  $y$  for some  $xRy \in \mathcal{G}$ ; otherwise,  $\sigma$  is empty.

Given the similar nature of the approaches, in the case of the logic  $K$ , the same rule can be used also for emulating the systems  $PT$  and  $NS$  (for convenience, in the following we will use for the same rule also the names  $decide_{PT}$  and  $decide_{NS}$ ).

For the system  $OS$ , we specialize instead the rule  $decide_F$  as follows:

$$\frac{\mathcal{G} \vdash_{\{y\}} \Theta \Downarrow \Omega}{\mathcal{G} \vdash_{\{x\}} \Theta \Uparrow} \text{decide}_{OS}$$

where (in addition to the general conditions of Figure 7) we have that

1. if  $x \neq y$ , then
  - $xRy \in \mathcal{G}$ ; and
  - $\Omega$  is a multiset of formulas of the form  $xy : \diamond A$ ;
2. if  $x = y$ , then  $\Omega = \{x : A\}$  for some formula  $A$  that is not a  $\diamond$ -formula.

Intuitively, the specialization with respect to the general framework consists in (i) restricting the use of multi-focusing to  $\diamond$ -formulas; (ii) forcing such  $\diamond$ -formulas to be labelled with the same future. This restriction is driven by the need for reproducing the behaviour of the  $OS$  modal rule  $\Box_K$ .

Let  $X$  range over  $\{LS, PT, OS, NS\}$ . We call  $LMF_X$  the system obtained from  $LMF_*$  by replacing the rule  $decide_F$  with the rule  $decide_X$ . The following adequacy result is proved by associating to each rule in  $X$  a corresponding sequence of bipoles in  $LMF_X$ . We refer the reader to Marin et al. (2016) for a more formal statement of the theorem as well as for its complete proof.

**Theorem 3.1.** Let  $X$  range over  $\{LS, PT, OS, NS\}$ . There exists a proof  $\Pi$  of  $A$  in the proof system  $X$  iff there exists a proof  $\Pi'$  of  $\emptyset \vdash_{\{x\}} x : (\downarrow A)^{\delta^+} \Uparrow$ , for any  $x$ , in  $LMF_X$ . Moreover, for each application of a rule  $r$  in  $\Pi$  there is a sequence of bipoles in  $\Pi'$  corresponding to  $r$ .

The result in Theorem 3.1 establishes a relation between the original calculi to be emulated and  $LMF_*$ . Since our ultimate goal is to certify proofs in a kernel which consists in  $LKF$ , we need to be able to relate  $LMF_*$  and  $LKF$  as well. First of all, based on the standard translation of Section 2.1.2, we refine the translation above in order to consider also the translation of modalities into quantified formulas.

Given a variable  $x$ , we define the translation  $[\cdot]_x$  from modal formulas in negation normal form into polarized first-order formulas as follows:

$$\begin{aligned}
 [P]_x &= P(x) & [A \wedge B]_x &= [A]_x^{\partial^+} \wedge^- [B]_x^{\partial^+} \\
 [\neg P]_x &= \neg P(x) & [A \vee B]_x &= [A]_x^{\partial^+} \vee^- [B]_x^{\partial^+} \\
 [\Box A]_x &= \forall y(\neg R(x, y) \vee^- [A]_y^{\partial^+}) & [\Diamond A]_x &= \exists y(R(x, y) \wedge^+ \partial^-([A]_y^{\partial^+}))
 \end{aligned}$$

We observe that the translation of modalities also makes use of delays, in such a way that the processing of a modality in the labelled calculus corresponds to a bipole in *LKF*, e.g., when in *LKF* we focus on a formula  $[\Diamond A]_x$ , the formula  $A$  is delayed in such a way that it gets necessarily stored at the end of the bipole. Based on that, we define the translation  $[\cdot]$  from labelled formulas and relational atoms into polarized first-order formulas as  $[x : A] = [A]_x$  and  $[xRy] = R(x, y)$ . Predicates of the form  $P(x)$  and  $R(x, y)$  are assigned positive polarity. This translation will be used for all the formalisms considered in the paper.

It is easy to notice that each proof in  $LMF_*$  is also a proof of  $LMF_m$  (just ignore the present of a sequent, as well as the present and future of formulas). Furthermore, a proof in the multi-focused system  $LMF_m$  can always be reproduced in  $LMF$ , by breaking a multi-focused bipole into a chain of single-focused bipoles. Finally, in Miller and Volpe (2015), it has been shown that a strict correspondence between proofs in  $LMF$  and proofs in  $LKF$  (restricted to the correspondence language) exists. This chain of correspondences allows us to state the following theoretical result, on which the adequacy of the implementation proposed in next section relies.

**Theorem 3.2.** Let  $X$  range over  $\{LS, PT, OS, NS\}$ . There exists a proof  $\Pi$  of  $A$  in the proof system  $X$  iff there exists a proof  $\Pi'$  of  $[A]_x$  in *LKF*, for any  $x$ . Moreover, for each application of a rule  $r$  in  $\Pi$ , there is a sequence of bipoles in  $\Pi'$  corresponding to  $r$ .

#### 4. Certification of modal proofs

This section describes the implementation of a general framework for the certification of modal proofs and shows how this framework can be used in order to certify proofs from different proof systems. We will rely here on the theoretical results of Section 3.

The implementation discussed in this paper is freely accesible on Github<sup>§</sup> or on Zenodo<sup>¶</sup>. More information on the implementation is given in Section 4.6.

FPC (see Section 2.2 for details) form a rich language for the certification of any proof object. This flexibility stems from connecting a trusted kernel with arbitrary  $\lambda$ Prolog programs (called FPC specifications). The richness of the language, however, has the downside that defining a new set of FPC specifications is, in general, a complex task – it involves the encoding of the semantics of a system over another (represented by the kernel). The complexity of supporting a new proof format is not unique to ProofCert. There are but a few general proof certification tools and the effort to enable the certification of a particular proof system is non-trivial.

<sup>§</sup> <https://github.com/proofcert/checkers/tree/dalefest>

<sup>¶</sup> <https://zenodo.org/record/1325924#.W2IWPVfgWM>

Our aim in this paper is to create a certification framework which enjoys generality and ease of use. Taking, for example, the *OS* system given in Figure 3, Theorem 3.2 has established the existence of a functional transformation from *OS* proofs into  $LMF_{OS}$ , which is a restriction of  $LMF^*$ . In Miller and Volpe (2015), the existence of a functional transformation from *LMF* proofs into *LKF* was shown to exist.

A simple approach to the certification of *OS* proofs would then be the direct translation of these proofs into *LKF*. Such a translation would amount to a soundness proof of the *OS* calculus and would violate our two criteria mentioned above. Generalization would be violated since the translation would target *OS* proofs only. Moreover, since such translations are generally very complex and include many technical details that are hidden in the theoretical proofs, they are not easy to implement.

In this paper, we present an approach based on the general proof checker presented in Section 2.2. We attempt to encode the semantics of different proof systems using logical programs (predicates), a trusted kernel and proof guidance and search. But, while defining the semantics using logical predicates is easier than using a functional translation, we are still left with the complexity of defining these predicates. In addition, it seems that by writing the predicates for a particular calculus, we compromise on generality.

A way to amend the generalization problem of the two approaches above is to consider a framework in which many different proofs can be certified. Such general frameworks are often cumbersome to use. Therefore, in order to make the certification as easy as possible, we would like to require the framework semantics to be as close to the semantics of the different proof calculi as possible.

The two properties stated above seem contradictory to each other. Being general and supporting different proof calculi and formats necessarily mean making it harder to implement any specific calculus and format. We try to circumvent this problem by introducing different layers in our framework. Some layers will be very general but harder to use, while others will be simpler but would not be able to support as many formats. In addition, the layers will be build on top of each other in such a way that using an upper layer necessarily means using also a lower one. As we will see, the simplest and lowest layer will be *LKF*. The remaining layers correspond to the systems which were introduced in Sections 3.1 and 3.2.

It should be noted that, as long as we always use the same lower level kernel, at no point trust is being compromised. Both the functional and the logical approaches are based on the reconstruction of a proof in the classical first-order sequent calculus. If such a proof is constructed for a certain (translation of the) original formula, we are assured that the formula is valid, up to the correctness of the first-order certifier as well as of the adequacy of the translation.

A simple layered approach would consist in using a separate kernel for each layer, but that would compromise some of the trust we can place in the proof certifier as well as violate the universality of the certification process, since certifications over different kernels cannot be combined into one, foundational, proof. For this reason, we will stick to one concrete kernel (*LKF*) and will simulate the different layers on top of it.

$$\frac{\frac{\frac{\frac{\frac{\overline{\vdash P, \neg P, Q} \quad \overline{\vdash \neg Q, \neg P, Q}}{\vdash P \wedge \neg Q, \neg P, Q}}{\vdash \diamond(P \wedge \neg Q), \diamond \neg P, \square Q}}{\vdash \diamond(P \wedge \neg Q), \diamond \neg P \vee \square Q}}{\vdash \diamond(P \wedge \neg Q) \vee (\diamond \neg P \vee \square Q)}}$$

Fig. 8. An OS proof of the axiom K.

4.1. *Introducing the framework*

In this section, we present an implementation of a proof certification framework based on the general proof checker from Section 2.2 and which uses *LKF* as its sole kernel. As mentioned above, such an approach enjoys the highest amount of trust.

The layers of our architecture will correspond to the implementation of systems that were described in the previous sections and that we briefly recall here:

1. *LKF* : a focused sequent calculus for the first-order classical logic; presented in Liang and Miller (2009) – see Section 2.3;
2. *LMF* : a focused labelled sequent calculus for the modal logic *K*; presented in Miller and Volpe (2015) – see Section 3.1;
3. *LMF<sub>m</sub>* : a multi-focused variant of *LMF* – see Section 3.1;
4. *LMF\** : a framework, based on *LMF<sub>m</sub>*, for the emulation of modal calculi; presented in Marin et al. (2016) – see Section 3.2.

Given, for example, the proof evidence for the OS proof in Figure 8, such a layered architecture allows us to certify the evidence over *LKF*, while defining the FPC over another layer, which is closer to the semantics of OS.

First, let us examine the information in the proof. The information contained in each inference step of an OS proof can be summarized as follows:

- The (main) formula occurrence to which the inference is applied (in case of  $\square_{OS}$ , we consider this to be the  $\square$ -formula introduced; in case of *init<sub>OS</sub>*, we consider this to be the positive literal in the couple of complementary literals).
- A possible additional list of formula occurrences:
  - in case of the rule  $\square_{OS}$ , all the  $\diamond$ -formula occurrences that are introduced by the rule;
  - in case of the rule *init<sub>OS</sub>*, the complementary negative literal.

Therefore, an adequate tree-shaped proof evidence for the above OS proof is the one shown in Figure 9.

In the next parts of this section, we will use the above example in order to discuss the implementation details of different aspects of the framework.

4.2. *Formula indices*

Our first challenge is to be able to refer to specific formula occurrences inside a proof whose conclusion contains a single formula *A*. The role of the indices will be to identify

- $\diamond(P \wedge \neg Q) \vee (\diamond\neg P \vee \Box Q)$
- $\diamond\neg P \vee \Box Q$
- $\Box Q$  with additional information  $\diamond(P \wedge \neg Q)$  and  $\diamond\neg P$
- $P \wedge \neg Q$ 
  - $P$  and the additional information  $\neg P$
  - $Q$  and the additional information  $\neg Q$

Fig. 9. The OS proof evidence for a proof of the axiom *K*.

- $\emptyset$
- $\text{right}(\emptyset)$
- $\text{right}(\text{right}(\emptyset))$  with the additional information  $\text{left}(\emptyset)$  and  $\text{left}(\text{right}(\emptyset))$
- $\text{left}(\text{left}(\emptyset))$ 
  - $\text{left}(\text{left}(\text{left}(\emptyset)))$  with the additional information  $\text{left}(\text{left}(\text{right}(\emptyset)))$
  - $\text{left}(\text{right}(\text{right}(\emptyset)))$  with the additional information  $\text{left}(\text{right}(\text{left}(\emptyset)))$

Fig. 10. OS proof evidence using basic indexing.

a specific subformula of *A*. For example, given an index *I* for a conjunctive formula, the index of the left conjunct can be defined as  $\text{left}(I)$ .

**Definition 4.1 (Basic indexing).** Given a formula *F* which has an index *I*, we define the following indices for the subformulas of *F*:

- If  $F = A \wedge B$  or  $F = A \vee B$ , we assign the index  $\text{left}(I)$  to *A* and  $\text{right}(I)$  to *B*.
- If  $F = \Box A$  or  $F = \diamond A$ , we assign the index  $\text{left}(I)$  to *A*.

Using the above basic indexing scheme and denoting the index of the theorem to prove by  $\emptyset$ , the proof from Figure 8 will be represented by the evidence in Figure 10.

While sufficient for an OS calculus for the system *K*, this indexing mechanism falls short for most other systems and calculi. The reason for that is represented by the implicit contractions of formulas which take place in such systems (for example, in *LS*, in the introduction rule for  $\diamond$ ). In our representation of proofs, this amounts to the need for indexing differently possible distinct occurrences of direct subformulas of a given  $\diamond$ -formula. In order to distinguish between them, we define a correspondence between  $\diamond$ -formulas and  $\Box$ -formulas inside a given proof. This will also allow us to capture the idea behind labels and to therefore omit explicit label information in our framework<sup>||</sup>.

**Definition 4.2 (Modal correspondence for LMF\*).** Let  $\Pi$  be an *LMF\** proof. We say that a labelled formula  $x : \diamond A$  corresponds in  $\Pi$  to a labelled formula  $y : \Box B$  iff

- there is a  $\Box_F$  rule application in  $\Pi$  whose conclusion is a sequent containing  $y : \Box B$  and whose premise is a sequent containing the formula  $z : B$ ; and

<sup>||</sup> In fact, in a (single-sided) labelled system, the correspondence would rather be between a  $\diamond$  introduction rule application and a specific label. However, at least in the case of the logic *K*, the only way to introduce (bottom-up) a new label is by means of a  $\Box$  introduction rule. It is thus possible to identify each label with the  $\Box$ -formula that introduces it.



- $\emptyset$
- $\text{right}(\emptyset)$
- $\text{right}(\text{right}(\emptyset))$  with the additional information  $\text{left}(\emptyset)$  and  $\text{left}(\text{right}(\emptyset))$
- $\text{diaind}(\text{left}(\emptyset), \text{right}(\text{right}(\emptyset)))$ 
  - $\text{left}(\text{diaind}(\text{left}(\emptyset), \text{right}(\text{right}(\emptyset))))$   
with the additional information  $\text{diaind}(\text{left}(\text{right}(\emptyset), \text{right}(\text{right}(\emptyset)))$
  - $\text{left}(\text{right}(\text{right}(\emptyset)))$   
with the additional information  $\text{right}(\text{diaind}(\text{left}(\emptyset), \text{right}(\text{right}(\emptyset))))$

Fig. 11. OS proof evidence using indexing.

- there is a  $\diamond_F$  rule application in  $\Pi$  whose conclusion is a sequent containing the formula  $x : \diamond A$  and whose premise is a sequent containing the formula  $z : A$ .

This definition can be extended to non-labelled systems such as OS and NS. For example, upon the application of the OS inference rule  $\square_{OS}$ , all  $\diamond$ -formulas in the lower sequent corresponds to the  $\square$ -formula. Similarly, for nested sequents,  $\diamond$  inference rules result in formulas being added to nested sequents which are associated with  $\square$ -formulas.

**Definition 4.3 (Modal correspondence for ordinary sequents).** Let  $\Pi$  be an OS proof. A formula  $\diamond A$  corresponds in  $\Pi$  to a formula  $\square B$  iff

- the conclusion of a  $\square_{OS}$  rule application in  $\Pi$  contains both formulas.

**Definition 4.4 (Modal correspondence for nested sequents).** Let  $\Pi$  be an NS proof. A formula  $\diamond A$  corresponds in  $\Pi$  to a formula  $\square B$  iff

- there is a  $\square_{NS}$  rule application in  $\Pi$  whose conclusion contains  $\mathcal{N}\{\square B\}$  and whose premise contains  $\mathcal{N}\{[B]\}$ ; and
- there is a  $\diamond_{NS}$  rule application in  $\Pi$  whose conclusion contains  $\mathcal{N}\{\diamond A, [\mathcal{M}]\}$  and whose premise contains  $\mathcal{N}\{[A, \mathcal{M}]\}$ , where  $\mathcal{M}$  contains  $B$ .

We omit the definitions for LS and PT, which can be easily inferred from the one given for LMF\* and are anyway not used in the examples of the paper.

Using modal correspondence, we can define the indices of  $\diamond$ -formulas.

**Definition 4.5 (Indexing).** Given a formula  $F$  which has an index  $I$ , we refine the definition of basic indexing (4.1) and replace the indexing of direct subformulas of  $\diamond$ -formulas as follows:

- If we apply the diamond inference rule to a formula  $F = \diamond A$  corresponding to a  $\square$ -formula at index  $J$ , we assign the index  $\text{diaind}(I, J)$  to  $A$ .

Using the indexing mechanism above on our example, we get the evidence in Figure 11.

### 4.3. Layered architecture

As mentioned in the introduction to this section, our aim is to implement a layered framework. On the one hand, the upper you go, the more restricted system you reach in which it is easier to define complex semantics of proof calculi. On the other hand, as you

go down, the layer syntax and definition are simpler and might be a better fit for some proof calculi.

No matter which layer we are using, we still wish our certification to be over our most trusted kernel – *LKF*. Let us consider the layer just above it – *LMF*. Given a proof evidence denoted in terms of the system of this layer, it is straightforward to define its FPC specification over *LKF*. When we consider one layer above – *LMF<sub>m</sub>*, it becomes more complex to define it over *LKF* but a relatively simpler matter to define it in terms of *LMF*.

**4.3.1. The *LMF* system layer.** In the previous examples, we have seen that modal proof evidences normally contain information about the worlds associated with  $\Box$  and  $\Diamond$  inference rules. Our first layer is capable, therefore, of accepting proof evidences which contain the following information:

1. At each step, on which formula we apply a rule of the *LMF* calculus.
2. In the case of a  $\Diamond$ -formula, with respect to which label (or, equivalently, as explained in the previous section, with respect to which  $\Box$ -formula) we apply the rule.
3. In the case of an initial rule, with respect to which complementary literal we apply it.

For this reason, we define the proof evidence of this layer as a tree describing the original proof. Each node is decorated by a pair containing (i) the index of the formula on which a rule is applied, as explained in (1), together with (ii) a (possibly null) further index carrying additional information, to be used in cases (2) and (3) above. Formulas in the tree will drive the construction (bottom-up) of the *LKF* derivation, in the sense that, by starting from the root, at each step, the *LKF* kernel will decide on the given formula and proceed, constrained by properly defined clerks and experts, along a synchronous and an asynchronous phase. The results in Libal and Volpe (2016) guarantee that at the end of a bipole, we will be in a situation which is equivalent to that of the corresponding step in the original proof.

As described in item (2) above, if we are applying an  $\exists$ -rule in *LKF*, then we need further information specifying with respect to which term we apply the rule. This term can be found in the additional index supplied. Similarly, in the case of an initial (3), the additional information in the node will specify the index of the complementary literal.

This definition permits the usage of different types of nodes in the same tree, which will allow us to smoothly move between the layers.

Using these definitions, we can now denote our example from Figure 11 in terms of the *LMF* layer, as can be seen in Figure 12. The corresponding *LMF* proof is shown in Figure 13. One can observe that each item in the proof evidence of Figure 12 corresponds to a block in the derivation of Figure 13 that starts, reading the proof bottom-up, with a *decide<sub>K</sub>* application and possibly uses, along the bipole, the additional information contained in the evidence. Note that, for shortness, we use in this proof (as well as in other examples below) derived rules for  $\partial^+$  and  $\partial^-$  introduction. They can be easily derived from other rules in the calculi.

```

—  $\emptyset$ 
— right( $\emptyset$ )
— right(right( $\emptyset$ ))
— left( $\emptyset$ ) with the additional information right(right( $\emptyset$ ))
— left(right( $\emptyset$ )) with the additional information right(right( $\emptyset$ ))
— diaind(left( $\emptyset$ ),right(right( $\emptyset$ )))
  — left(diaind(left( $\emptyset$ ),right(right( $\emptyset$ ))))
    with the additional information diaind(left(right( $\emptyset$ ),right(right( $\emptyset$ )))
  — left(right(right( $\emptyset$ )))
    with the additional information right(diaind(left( $\emptyset$ ),right(right( $\emptyset$ )))

```

Fig. 12. The *LMF* proof evidence for a proof of the axiom *K*.

The implementation<sup>††</sup> of the *LMF* layer over the *LKF* kernel in our system will mainly do the following:

- Use the index in each node of the tree to choose the right formula to decide on.
- Use the extra information for the  $\diamond$ -formula when applying an  $\exists$  rule.
- Use the extra information for the literals when applying an `init` rule.

A simplified version (omitting some technical details) of a part of the implementation is given in Figure 14. Here, a definition consists of a predicate name, e.g., `orNeg_c`, and a list of arguments. The predicates correspond to those introduced in *LKF<sup>a</sup>* (Figure 5), and can be divided into clerks (ending with `_c`) and experts (ending with `_e`).

The role of the arguments is also the same as in Figure 5. Typically, the first and last arguments of each predicate are the ‘input’ and ‘output’ evidence of the inference rule, i.e., the evidence of the conclusion and the evidence of the premise, respectively. In some cases, further arguments are present, e.g., in the `some_e` predicate of the example, a second argument containing information about the witness term to be used in the  $\exists$ -introduction application. In the implementation, an evidence of the *LMF* layer is a term named `lmf_cert` which has two arguments: (i) a state (holding additional information, as described below) and (ii) a tree. The tree is a regular inductively defined tree node `I 0 List` where `List` is a list of sub-trees of the current node and `I` and `0` are indices describing the current node in terms of the formula to decide on and the optional additional information.

The first definition, for the negative disjunction, just skips the root node of the proof tree in the evidence. Since we are in an asynchronous phase, the information in the evidence is not required. On the other hand, the information transmitted to the clerk from the kernel – the principal formula – is ignored by the clerk since it is not required later. We also remark on the existence of a state variable. The state is being used in order to propagate some information between the rule applications. It is not used in this FPC definition.

The second definition, for the universal quantifier, does also skip the current node, similarly to the previous definition. It does need, though, to record the eigenvariable used

<sup>††</sup> `src/fpc/modal/lmf-singlefoc.mod`

$$\begin{array}{c}
 \frac{\frac{\frac{\overline{\{xRy\} \vdash \Sigma_7 \Downarrow y : P} \text{init}_K}{\overline{\{xRy\} \vdash \Sigma_7 \equiv \Sigma_6 \cup \{y : P\} \Uparrow} \cdot} \text{decide}_K}{\overline{\{xRy\} \vdash \Sigma_6 \Uparrow y : P} \cdot} \text{store}_K}{\overline{\{xRy\} \vdash \Sigma_6 \Uparrow y : P} \cdot} \text{decide}_K \quad \frac{\frac{\frac{\overline{\{xRy\} \vdash \Sigma_8 \Uparrow y : Q} \text{init}_K}{\overline{\{xRy\} \vdash \Sigma_8 \equiv \Sigma_6 \cup \{y : \neg Q\} \Uparrow} \cdot} \text{decide}_K}{\overline{\{xRy\} \vdash \Sigma_6 \Uparrow y : \neg Q} \cdot} \text{store}_K}{\overline{\{xRy\} \vdash \Sigma_6 \Uparrow y : \neg Q} \cdot} \text{decide}_K}{\overline{\{xRy\} \vdash \Sigma_6 \Uparrow y : P \wedge \neg Q} \cdot} \wedge_K \\
 \frac{\frac{\frac{\overline{\{xRy\} \vdash \Sigma_6 \Downarrow y : P \wedge \neg Q} \text{release}_K}{\overline{\{xRy\} \vdash \Sigma_6 \Downarrow y : \partial^+(P \wedge \neg Q)} \cdot} \partial^+}{\overline{\{xRy\} \vdash \Sigma_6 \equiv \Sigma_5 \cup \{y : \neg P\} \Uparrow} \cdot} \text{decide}_K}{\overline{\{xRy\} \vdash \Sigma_5 \Uparrow y : \neg P} \cdot} \text{store}_K \\
 \frac{\frac{\frac{\overline{\{xRy\} \vdash \Sigma_5 \Uparrow y : \neg P} \partial^-}{\overline{\{xRy\} \vdash \Sigma_5 \Uparrow y : \partial^-(\neg P)} \cdot} \text{decide}_K}{\overline{\{xRy\} \vdash \Sigma_5 \Downarrow y : \partial^-(\neg P)} \cdot} \text{release}_K}{\overline{\{xRy\} \vdash \Sigma_5 \Downarrow y : \partial^-(\neg P)} \cdot} \text{decide}_K}{\overline{\{xRy\} \vdash \Sigma_5 \Downarrow x : \diamond \partial^-(\neg P)} \cdot} \diamond_K \\
 \frac{\frac{\frac{\overline{\{xRy\} \vdash \Sigma_5 \equiv \Sigma_4 \cup \{y : \partial^+(P \wedge \neg Q)\} \Uparrow} \cdot} \text{decide}_K}{\overline{\{xRy\} \vdash \Sigma_4 \Uparrow y : \partial^+(P \wedge \neg Q)} \cdot} \text{store}_K}{\overline{\{xRy\} \vdash \Sigma_4 \Uparrow y : \partial^-(\partial^+(P \wedge \neg Q))} \cdot} \partial^-}{\overline{\{xRy\} \vdash \Sigma_4 \Downarrow y : \partial^-(\partial^+(P \wedge \neg Q))} \cdot} \text{release}_K}{\overline{\{xRy\} \vdash \Sigma_4 \Downarrow x : \diamond(\partial^-(\partial^+(P \wedge \neg Q)))} \cdot} \diamond_K}{\overline{\{xRy\} \vdash \Sigma_4 \equiv \Sigma_3 \cup \{y : Q\} \Uparrow} \cdot} \text{decide}_K}{\overline{\{xRy\} \vdash \Sigma_3 \Uparrow y : Q} \cdot} \text{store}_K \\
 \frac{\frac{\frac{\overline{\{xRy\} \vdash \Sigma_3 \Uparrow y : Q} \square_K}{\overline{\emptyset \vdash \Sigma_3 \Uparrow x : \square Q} \cdot} \text{release}_K}{\overline{\emptyset \vdash \Sigma_3 \Downarrow x : \square Q} \cdot} \partial^+}{\overline{\emptyset \vdash \Sigma_3 \Downarrow x : \partial^+(\square Q)} \cdot} \text{decide}_K}{\overline{\emptyset \vdash \Sigma_3 \equiv \Sigma_2 \cup \{x : \partial^+(\square Q)\} \Uparrow} \cdot} \text{store}_K}{\overline{\emptyset \vdash \Sigma_2 \equiv \Sigma_1 \cup \{x : \diamond \partial^-(\neg P)\} \Uparrow x : \partial^+(\square Q)} \cdot} \text{store}_K}{\overline{\emptyset \vdash \Sigma_1 \Uparrow x : \diamond \partial^-(\neg P), x : \partial^+(\square Q)} \cdot} \text{store}_K}{\overline{\emptyset \vdash \Sigma_1 \Uparrow x : \diamond \partial^-(\neg P) \vee \partial^+(\square Q)} \cdot} \vee_K}{\overline{\emptyset \vdash \Sigma_1 \Downarrow x : \diamond \partial^-(\neg P) \vee \partial^+(\square Q)} \cdot} \text{release}_K}{\overline{\emptyset \vdash \Sigma_1 \Downarrow x : \partial^+(\diamond \partial^-(\neg P) \vee \partial^+(\square Q))} \cdot} \partial^+}{\overline{\emptyset \vdash \Sigma_1 \equiv \Sigma \cup \{x : \partial^+(\diamond \partial^-(\neg P) \vee \partial^+(\square Q))\} \Uparrow} \cdot} \text{decide}_K}{\overline{\emptyset \vdash \Sigma \equiv \{x : \varphi, x : \diamond(\partial^-(\partial^+(P \wedge \neg Q)))\} \Uparrow x : \partial^+(\diamond \partial^-(\neg P) \vee \partial^+(\square Q))} \cdot} \text{store}_K}{\overline{\emptyset \vdash x : \varphi \Uparrow x : \diamond(\partial^-(\partial^+(P \wedge \neg Q))), x : \partial^+(\diamond \partial^-(\neg P) \vee \partial^+(\square Q))} \cdot} \text{store}_K}{\overline{\emptyset \vdash x : \varphi \Uparrow x : \varphi} \cdot} \vee_K \\
 \frac{\frac{\frac{\overline{\emptyset \vdash x : \varphi \Uparrow x : \varphi} \text{release}_K}{\overline{\emptyset \vdash x : \varphi \Downarrow x : \varphi} \cdot} \text{decide}_K}{\overline{\emptyset \vdash x : \varphi \Uparrow} \cdot} \text{decide}_K}
 \end{array}$$

Fig. 13. The LMF proof corresponding to the proof evidence of Figure 12, where  $\varphi \equiv \diamond(\partial^-(\partial^+(P \wedge \neg Q))) \vee (\partial^+(\diamond \partial^-(\neg P) \vee \partial^+(\square Q)))$  is a polarized version of the axiom  $K$ , obtained according to the translation in Section 3.3.

```

orNeg_c
  (lmf-cert State (node I 0 [H|T]))
  (lmf-cert State H).

all_c
  (lmf-cert (state M) (node I 0 [H|T]))
  (X \ lmf-cert (state [pair I X|M]) H).

some_e
  (lmf-cert (state M) (node I 0 [H|T]))
  X
  (lmf-cert (state M) H) :-
    member (pair 0 X) M.

```

Fig. 14. A simplified version of the implementation of three FPC definitions.

and stores it in a mapping in the state. This mapping associates the index of the node, i.e., the index of the corresponding  $\square$ -formula, to the actual eigenvariable introduced by the kernel. We store these values in the state using a list of pairs of indices and eigenvariables. Since the predicate cannot intervene directly with the kernel in order to obtain the eigenvariable, we have defined the second argument of this predicate as a function from eigenvariables to proof evidences. The kernel is then responsible for applying this function to the eigenvariable.

The last definition, which is an expert FPC definition, selects the previously stored eigenvariable and returns it as the term witness. In order to find the correct eigenvariable, we check for the membership of a pair of the known index and the required eigenvariable in the list which forms the state.

4.3.2. *The  $LMF_m$  system layer.* One can immediately see that the  $LMF$  layer is not the most suitable for describing the semantics of  $OS$ . The reason is that the order in which  $\diamond$ -formulas are decided on, which is explicit in  $LMF$ , is not always relevant in the corresponding  $OS$  proof.

We would like to have a layer which allows us to decide simultaneously on different formulas in the sequent. This is obtained by multi-focusing.

The  $LMF_m$  layer allows us to simulate a multi-focusing step in the kernel (which is non-multi-focused) and corresponds to the multi-focused version of  $LMF$  defined in Section 3.1. Our system will simulate multi-focusing by relating each inference with a number. This number will force all inferences labelled the same to occur sequentially. We observe that this does not simulate multi-focusing adequately in the general case; however, for the modal proof calculi considered in this paper and due to the fact that we restrict to the logic  $K$ , we are ensured that this simple mechanism is enough for encoding multi-focusing in our case. We note here that in order to support multi-focusing in logics other than  $K$ , we would need to support a multi-focused version of  $LKF$  as our kernel.

- $\emptyset$  with the additional information multi-focus value 1
- $\text{right}(\emptyset)$  with the additional information multi-focus value 2
- $\text{right}(\text{right}(\emptyset))$  with the additional information multi-focus value 3
- $\text{left}(\emptyset)$  with the additional information  $\text{right}(\text{right}(\emptyset))$  and multi-focus value 4
- $\text{left}(\text{right}(\emptyset))$  with the additional information  $\text{right}(\text{right}(\emptyset))$  and multi-focus value 4
- $\text{diaind}(\text{left}(\emptyset), \text{right}(\text{right}(\emptyset)))$  with the additional information multi-focus value 5
  - $\text{left}(\text{diaind}(\text{left}(\emptyset), \text{right}(\text{right}(\emptyset))))$  with the additional information  $\text{diaind}(\text{left}(\text{right}(\emptyset), \text{right}(\text{right}(\emptyset)))$  and multi-focus value 6
  - $\text{left}(\text{right}(\text{right}(\emptyset)))$  with the additional information  $\text{right}(\text{diaind}(\text{left}(\emptyset), \text{right}(\text{right}(\emptyset))))$  and multi-focus value 7

Fig. 15. The  $LMF_m$  proof evidence for a proof of the axiom  $K$ .

A proof evidence for our running example in  $LMF_m$  can be seen in Figure 15. The multi-focus value which appears there is just an integer which is used to group simultaneous focusing. The corresponding  $LMF_m$  proof is shown in Figure 16. With respect to the  $LMF$  proof of Figure 13, we can notice that here we have a *decide* step in which the focus is put on two  $\diamond$ -formulas at the same time; such formulas correspond to those indices having the same multi-focus value in Figure 15.

4.3.3. *The  $LMF_*$  system layer.* The most expressive layer is  $LMF_*$ . This layer extends the previous one with information about worlds which are currently active (the present) and the possible futures of formulas.

Going back to our running example, we see that we still cannot simulate properly the semantics of  $OS$  calculus. There is no mechanism in our  $LMF_m$  layer which enforces all  $\diamond$ -formulas to ‘go’ to the same world (the one introduced by the corresponding  $\square$ -formula). Essentially, we want to forbid proof evidences where  $\diamond$ -formulas belonging to the same multi-focusing step correspond to different  $\square$ -formulas, because we know that in such a case our kernel would not be simulating an  $OS$  proof. We can impose additional restrictions based on the tools from Section 3.2, which will ensure such cases cannot happen.

Our running example proof evidence will now look like the one in Figure 17. The nodes of a proof evidence, as defined in Section 3.2, contain, in addition to the information required in the previous layer, also information about the new present of the sequent and future of formulas, denoted by labels<sup>‡‡</sup>. Essentially, the new features of this layer help us further restrict the proof search over the previous layer by giving us the ability to avoid

<sup>‡‡</sup> We remark that, for simplicity and since it is equivalent in the case of the systems considered, in the implementation we attach the information concerning the future to the whole sequent, rather than to single formulas as described in Section 3.2.

$$\begin{array}{c}
 \frac{\frac{\overline{\{xRy\} \vdash \Sigma_7 \Downarrow y : P} \text{init}_K}{\overline{\{xRy\} \vdash \Sigma_7 \equiv \Sigma_6 \cup \{y : P\} \Uparrow} \cdot} \text{decide}_K}{\overline{\{xRy\} \vdash \Sigma_6 \Uparrow y : P} \text{store}_K} \quad \frac{\frac{\overline{\{xRy\} \vdash \Sigma_8 \Uparrow y : Q} \text{init}_K}{\overline{\{xRy\} \vdash \Sigma_8 \equiv \Sigma_6 \cup \{y : \neg Q\} \Uparrow} \cdot} \text{decide}_K}{\overline{\{xRy\} \vdash \Sigma_6 \Uparrow y : \neg Q} \text{store}_K} \\
 \frac{\overline{\{xRy\} \vdash \Sigma_6 \Uparrow y : P \wedge \neg Q} \text{release}_K}{\overline{\{xRy\} \vdash \Sigma_6 \Downarrow y : P \wedge \neg Q} \text{decide}_K} \wedge_K \\
 \frac{\overline{\{xRy\} \vdash \Sigma_6 \Downarrow y : \partial^+(P \wedge \neg Q)} \partial^+}{\overline{\{xRy\} \vdash \Sigma_6 \equiv \Sigma_5 \cup \{y : \neg P\} \Uparrow} \cdot} \text{decide}_K \\
 \frac{\overline{\{xRy\} \vdash \Sigma_5 \Uparrow y : \neg P} \text{store}_K}{\overline{\{xRy\} \vdash \Sigma_5 \equiv \Sigma_4 \cup \{y : \partial^+(P \wedge \neg Q)\} \Uparrow y : \partial^-(\neg P)} \partial^- \\
 \frac{\overline{\{xRy\} \vdash \Sigma_4 \Uparrow y : \partial^+(P \wedge \neg Q), y : \partial^-(\neg P)} \partial^-}{\overline{\{xRy\} \vdash \Sigma_4 \Uparrow y : \partial^-(\partial^+(P \wedge \neg Q)), y : \partial^-(\neg P)} \text{release}_K} \\
 \frac{\overline{\{xRy\} \vdash \Sigma_4 \Downarrow y : \partial^-(\partial^+(P \wedge \neg Q)), y : \partial^-(\neg P)} \diamond_K}{\overline{\{xRy\} \vdash \Sigma_4 \Downarrow y : \partial^-(\partial^+(P \wedge \neg Q)), x : \diamond \partial^-(\neg P)} \diamond_K} \\
 \frac{\overline{\{xRy\} \vdash \Sigma_4 \Downarrow x : \diamond(\partial^-(\partial^+(P \wedge \neg Q))), x : \diamond \partial^-(\neg P)} \text{decide}_K}{\overline{\{xRy\} \vdash \Sigma_4 \equiv \Sigma_3 \cup \{y : Q\} \Uparrow} \cdot} \text{store}_K \\
 \frac{\overline{\{xRy\} \vdash \Sigma_3 \Uparrow y : Q} \square_K}{\overline{\emptyset \vdash \Sigma_3 \Uparrow x : \square Q} \text{release}_K} \\
 \frac{\overline{\emptyset \vdash \Sigma_3 \Downarrow x : \square Q} \partial^+}{\overline{\emptyset \vdash \Sigma_3 \Downarrow x : \partial^+(\square Q)} \text{decide}_K} \\
 \frac{\overline{\emptyset \vdash \Sigma_3 \equiv \Sigma_2 \cup \{x : \partial^+(\square Q)\} \Uparrow} \cdot} \text{store}_K \\
 \frac{\overline{\emptyset \vdash \Sigma_2 \equiv \Sigma_1 \cup \{x : \diamond \partial^-(\neg P)\} \Uparrow x : \partial^+(\square Q)} \text{store}_K}{\overline{\emptyset \vdash \Sigma_1 \Uparrow x : \diamond \partial^-(\neg P), x : \partial^+(\square Q)} \text{store}_K} \\
 \frac{\overline{\emptyset \vdash \Sigma_1 \Uparrow x : \diamond \partial^-(\neg P) \vee \partial^+(\square Q)} \sqrt{K}}{\overline{\emptyset \vdash \Sigma_1 \Downarrow x : \diamond \partial^-(\neg P) \vee \partial^+(\square Q)} \text{release}_K} \\
 \frac{\overline{\emptyset \vdash \Sigma_1 \Downarrow x : \partial^+(\diamond \partial^-(\neg P) \vee \partial^+(\square Q))} \partial^+}{\overline{\emptyset \vdash \Sigma_1 \equiv \Sigma \cup \{x : \partial^+(\diamond \partial^-(\neg P) \vee \partial^+(\square Q))\} \Uparrow} \cdot} \text{decide}_K \\
 \frac{\overline{\emptyset \vdash \Sigma \equiv \{x : \varphi, x : \diamond(\partial^-(\partial^+(P \wedge \neg Q)))\} \Uparrow x : \partial^+(\diamond \partial^-(\neg P) \vee \partial^+(\square Q))} \text{store}_K}{\overline{\emptyset \vdash x : \varphi \Uparrow x : \diamond(\partial^-(\partial^+(P \wedge \neg Q))), x : \partial^+(\diamond \partial^-(\neg P) \vee \partial^+(\square Q))} \text{store}_K} \\
 \frac{\overline{\emptyset \vdash x : \varphi \Uparrow x : \varphi} \text{release}_K}{\overline{\emptyset \vdash x : \varphi \Downarrow x : \varphi} \text{decide}_K} \sqrt{K} \\
 \overline{\emptyset \vdash x : \varphi \Uparrow} \cdot
 \end{array}$$

Fig. 16. The  $LMF_m$  proof corresponding to the proof evidence of Figure 15, where  $\varphi \equiv \diamond(\partial^-(\partial^+(P \wedge \neg Q))) \vee (\partial^+(\diamond \partial^-(\neg P) \vee \partial^+(\square Q)))$  is a polarized version of the axiom  $K$ , obtained according to the translation in Section 3.3.

applying the *decide* and  $\diamond$ -introduction rules in some cases. In Figure 18, we show the  $LMF_*$  proof corresponding to the proof evidence of Figure 17. The proof has the same structure as the one in Figure 16 and the way the additional ‘decorations’ (denoting the present and the future) are used, respect the restrictions that characterize  $OS$  and make it indeed an  $LMF_{OS}$  proof.

- $\emptyset$  with the additional information multi-focus value 1, present  $\{\emptyset\}$  and an empty future
- $\text{right}(\emptyset)$  with the additional information multi-focus value 2, present  $\{\emptyset\}$  and an empty future
- $\text{right}(\text{right}(\emptyset))$  with the additional information multi-focus value 3, present  $\{\emptyset\}$  and an empty future
- $\text{left}(\emptyset)$  with the additional information  $\text{right}(\text{right}(\emptyset))$ , multi-focus value 4, present  $\{\text{right}(\text{right}(\emptyset))\}$  and future  $\text{right}(\text{right}(\emptyset))$
- $\text{left}(\text{right}(\emptyset))$  with the additional information  $\text{right}(\text{right}(\emptyset))$ , multi-focus value 4, present  $\{\text{right}(\text{right}(\emptyset))\}$  and future  $\text{right}(\text{right}(\emptyset))$
- $\text{diaind}(\text{left}(\emptyset), \text{right}(\text{right}(\emptyset)))$  with the additional information multi-focus value 5, present  $\{\text{right}(\text{right}(\emptyset))\}$  and an empty future
  - $\text{left}(\text{diaind}(\text{left}(\emptyset), \text{right}(\text{right}(\emptyset))))$  with the additional information  $\text{diaind}(\text{left}(\text{right}(\emptyset), \text{right}(\text{right}(\emptyset)))$ , multi-focus value 6, present  $\{\text{right}(\text{right}(\emptyset))\}$  and an empty future
  - $\text{left}(\text{right}(\text{right}(\emptyset)))$  with the additional information  $\text{right}(\text{diaind}(\text{left}(\emptyset), \text{right}(\text{right}(\emptyset))))$  and multi-focus value 7, present  $\{\text{right}(\text{right}(\emptyset))\}$  and an empty future

Fig. 17. The  $LMF_*$  proof evidence for a proof of the axiom  $K$ .

#### 4.4. Polymorphic proof evidence

As we mentioned in previous sections, one of the goals of our framework is ease of use. When implementing the integration of specific modal systems, we would like to use the layer which is closest to the semantics of the system, e.g., the  $LMF_*$  layer in the case of  $OS$ . When considering the proof evidences of these two systems (Figures 11 and 17), we notice that they are not that similar.

In this section, we present the first approach for amending this problem – ease of use while preserving trust and universality – which we call *polymorphic proof evidence*. The name stems from the fact that in order to use a layer and be able to certify over the  $LKF$  kernel, we consider different proof evidences as being polymorphic, i.e., belonging to different proof calculi and layers.

In the example of  $OS$ , by polymorphic proof evidence, we mean that these proof evidences, from the implementation point of view, are both  $OS$  proofs and  $LMF_*$  proofs. In a similar way, we can define an  $LMF_*$  proof as being both an  $LMF_*$  proof and an  $LMF_m$  one. By following this approach, we will obtain that the ordinary proof evidence, being built on top of our topmost layer, is a proof evidence of all mentioned systems.

From the certification point of view, this will allow us to certify the  $OS$  proof evidence ‘out of the box’ by just considering it as an  $LMF_*$  evidence. From the implementation point of view, we need to be able to define the evidence as polymorphic.  $\lambda$ Prolog, being a logic programming language, does not support polymorphism on the object level (it does, though, support type polymorphism similar to the one in functional programming languages).

We overcome that by using programs which translates the evidences across layers.

The first set of programs are based on the fact that each layer is built on top of the one below it. We will include the proof evidence of the lower layer within the proof evidence



$$\begin{array}{c}
 \frac{\overline{\{xRy\} \vdash_{\{y\}} \Sigma_7 \Downarrow y : P} \text{ init}_K}{\overline{\{xRy\} \vdash_{\{y\}} \Sigma_7 \equiv \Sigma_6 \cup \{y : P\} \Uparrow} \cdot} \text{decide}_K \quad \frac{\overline{\{xRy\} \vdash_{\{y\}} \Sigma_8 \Uparrow y : Q} \text{ init}_K}{\overline{\{xRy\} \vdash_{\{y\}} \Sigma_8 \equiv \Sigma_6 \cup \{y : \neg Q\} \Uparrow} \cdot} \text{decide}_K \\
 \frac{}{\overline{\{xRy\} \vdash_{\{y\}} \Sigma_6 \Uparrow y : P} \text{ store}_K} \quad \frac{}{\overline{\{xRy\} \vdash_{\{y\}} \Sigma_6 \Uparrow y : \neg Q} \text{ store}_K} \\
 \frac{}{\overline{\{xRy\} \vdash_{\{y\}} \Sigma_6 \Uparrow y : P \wedge \neg Q} \wedge_K} \\
 \frac{}{\overline{\{xRy\} \vdash_{\{y\}} \Sigma_6 \Downarrow y : P \wedge \neg Q} \text{ release}_K} \\
 \frac{}{\overline{\{xRy\} \vdash_{\{y\}} \Sigma_6 \Downarrow y : \partial^+(P \wedge \neg Q)} \partial^+} \\
 \frac{}{\overline{\{xRy\} \vdash_{\{y\}} \Sigma_6 \equiv \Sigma_5 \cup \{y : \neg P\} \Uparrow} \cdot} \text{decide}_K \\
 \frac{}{\overline{\{xRy\} \vdash_{\{y\}} \Sigma_5 \Uparrow y : \neg P} \text{ store}_K} \\
 \frac{}{\overline{\{xRy\} \vdash_{\{y\}} \Sigma_5 \equiv \Sigma_4 \cup \{y : \partial^+(P \wedge \neg Q)\} \Uparrow y : \partial^-(\neg P)} \partial^-} \\
 \frac{}{\overline{\{xRy\} \vdash_{\{y\}} \Sigma_4 \Uparrow y : \partial^+(P \wedge \neg Q), y : \partial^-(\neg P)} \text{ store}_K} \\
 \frac{}{\overline{\{xRy\} \vdash_{\{y\}} \Sigma_4 \Uparrow y : \partial^-(\partial^+(P \wedge \neg Q)), y : \partial^-(\neg P)} \partial^-} \\
 \frac{}{\overline{\{xRy\} \vdash_{\{y\}} \Sigma_4 \Downarrow y : \partial^-(\partial^+(P \wedge \neg Q)), y : \partial^-(\neg P)} \text{ release}_K} \\
 \frac{}{\overline{\{xRy\} \vdash_{\{y\}} \Sigma_4 \Downarrow y : \partial^-(\partial^+(P \wedge \neg Q)), xy : \diamond \partial^-(\neg P)} \diamond_K} \\
 \frac{}{\overline{\{xRy\} \vdash_{\{y\}} \Sigma_4 \Downarrow xy : \diamond(\partial^-(\partial^+(P \wedge \neg Q))), xy : \diamond \partial^-(\neg P)} \diamond_K} \\
 \frac{}{\overline{\{xRy\} \vdash_{\{x\}} \Sigma_4 \equiv \Sigma_3 \cup \{y : Q\} \Uparrow} \cdot} \text{decide}_K \\
 \frac{}{\overline{\{xRy\} \vdash_{\{x\}} \Sigma_3 \Uparrow y : Q} \text{ store}_K} \\
 \frac{}{\overline{\emptyset \vdash_{\{x\}} \Sigma_3 \Uparrow x : \square Q} \square_K} \\
 \frac{}{\overline{\emptyset \vdash_{\{x\}} \Sigma_3 \Downarrow x : \square Q} \text{ release}_K} \\
 \frac{}{\overline{\emptyset \vdash_{\{x\}} \Sigma_3 \Downarrow x : \partial^+(\square Q)} \partial^+} \\
 \frac{}{\overline{\emptyset \vdash_{\{x\}} \Sigma_3 \equiv \Sigma_2 \cup \{x : \partial^+(\square Q)\} \Uparrow} \cdot} \text{decide}_K \\
 \frac{}{\overline{\emptyset \vdash_{\{x\}} \Sigma_2 \equiv \Sigma_1 \cup \{x : \diamond \partial^-(\neg P)\} \Uparrow x : \partial^+(\square Q)} \text{ store}_K} \\
 \frac{}{\overline{\emptyset \vdash_{\{x\}} \Sigma_2 \equiv \Sigma_1 \cup \{x : \diamond \partial^-(\neg P)\} \Uparrow x : \partial^+(\square Q)} \text{ store}_K} \\
 \frac{}{\overline{\emptyset \vdash_{\{x\}} \Sigma_1 \Uparrow x : \diamond \partial^-(\neg P), x : \partial^+(\square Q)} \sqrt{K}} \\
 \frac{}{\overline{\emptyset \vdash_{\{x\}} \Sigma_1 \Downarrow x : \diamond \partial^-(\neg P) \vee \partial^+(\square Q)} \text{ release}_K} \\
 \frac{}{\overline{\emptyset \vdash_{\{x\}} \Sigma_1 \Downarrow x : \partial^+(\diamond \partial^-(\neg P) \vee \partial^+(\square Q))} \partial^+} \\
 \frac{}{\overline{\emptyset \vdash_{\{x\}} \Sigma_1 \equiv \Sigma \cup \{x : \partial^+(\diamond \partial^-(\neg P) \vee \partial^+(\square Q))\} \Uparrow} \cdot} \text{decide}_K \\
 \frac{}{\overline{\emptyset \vdash_{\{x\}} \Sigma \equiv \{x : \varphi, x : \diamond(\partial^-(\partial^+(P \wedge \neg Q)))\} \Uparrow x : \partial^+(\diamond \partial^-(\neg P) \vee \partial^+(\square Q))} \text{ store}_K} \\
 \frac{}{\overline{\emptyset \vdash_{\{x\}} x : \varphi \Uparrow x : \diamond(\partial^-(\partial^+(P \wedge \neg Q))), x : \partial^+(\diamond \partial^-(\neg P) \vee \partial^+(\square Q))} \text{ store}_K} \\
 \frac{}{\overline{\emptyset \vdash_{\{x\}} x : \varphi \Uparrow x : \varphi} \sqrt{K}} \\
 \frac{}{\overline{\emptyset \vdash_{\{x\}} x : \varphi \Downarrow x : \varphi} \text{ release}_K} \\
 \frac{}{\overline{\emptyset \vdash_{\{x\}} x : \varphi \Uparrow} \cdot} \text{decide}_K
 \end{array}$$

Fig. 18. The  $LMF_*$  proof corresponding to the proof evidence of Figure 17, where  $\varphi \equiv \diamond(\partial^-(\partial^+(P \wedge \neg Q))) \vee (\partial^+(\diamond \partial^-(\neg P) \vee \partial^+(\square Q)))$  is a polarized version of the axiom  $K$ , obtained according to the translation in Section 3.3.

of the upper one. Polymorphism will be obtained by considering, in each layer, only the relevant component.

The second is based on the fact that a proof evidence, like that for an  $OS$  proof, can be denoted in terms of the proof evidence expected in one of the layers. Polymorphism is

```

star_to_multi-foc (star_cert S (multi-foc_cert M))
  (multi-foc_cert M) S.

multi-foc_to_star (multi-foc_cert M) S
  (star_cert S (multi-foc_cert M)).

orNeg_c Cert Form Cert' :-
  star_to_multi-foc Cert Cert-m S,
  % call matching a definition in the multi-foc layer
  orNeg_c Cert-m Form Cert-m',
  multi-foc_to_star Cert-m' S Cert'.

```

Fig. 19. Proof evidence transformation between two layers.

obtained via a logical program which translates the evidence back and forth between the original proof and the one of the relevant layer.

By explicitly defining these small programs, we allow users to certify proof evidence, whose semantics are defined using one of the layers, on top of the *LKF* kernel.

An example of using a translation between an  $LMF_*$  and  $LMF_m$  proof evidences is given in Figure 19.

A program which enables this ‘polymorphic’ behaviour for *OS* over our framework is given in Figure 20. In order to apply the FPC specification of the  $LMF_*$  layer, we convert the proof evidence to the one expected by the  $LMF_*$  layer and recursively apply the predicate. The logic programming mechanism will use the transformed proof evidence in order to locate the proper predicate to apply. We can, therefore, simulate polymorphism over the proof evidences.

We can see that the essential information about the index of the formula a rule is applied to, as well as the optional additional index (denoted *I* and *OI* in the figure) are copied between the evidences. The information which is not part of the *OS* evidence – the multi-focus index, the future and the set of presents – is being stored in a state like data structure and is copied to the  $LMF_*$  evidence. The state of the *OS* evidence is initially empty and is being updated by the *OS* FPC specification.

Here, we have shown the relatively simple (abstraction over the) definition of the `orNeg_c` FPC specification. The one for `allNeg_c`, for example, includes a simulation of the application of  $\diamond$  inference rules.

#### 4.5. Certification of different proof formats

Given the different layers in the proof system defined in the previous section, we can easily write FPC specifications for different popular proof formats.

The process is always the same. The FPC specifications translate the evidence into the evidence of a particular layer, as explained in Section 4.4.

In the next sections, we describe in more detail how the framework is used in order to support specific proof formats. In all cases, in order to perform the proof reconstruction inside *LKF*, the modal formula to be proved is translated according to the translation [.] of Section 3.3.

```

ordinary_to_star
  (ordinary_cert (ordinary_state H F M) I OI)
  (star_cert H F (multi-foc_cert M (single_cert I OI))).

star_to_ordinary
  (star_cert H F (multi-foc_cert M (single_cert I OI)))
  (ordinary_cert (ordinary_state H F M) I OI).

orNeg_c Cert Form Cert' :-
  ordinary_to_star Cert Cert-s,
  orNeg_c Cert-s Form Cert-s',
  star_to_ordinary Cert-s' Cert'.

```

Fig. 20. Proof evidence transformation between *OS* and *LMF\**.

4.5.1. *Labelled sequents.* The treatment of labelled systems (*LS*) (Negri 2005) was already implemented in the previous version of *Checkers*, described in Libal and Volpe (2016). In order to get emulation of *LS*, we require a very simple use of the framework *LMF\**, where at each sequent the present corresponds to the set of all the labels occurring in the proof, no use of multi-focusing is required and the future of a formula is set, in the case of  $\diamond$ -formulas, to the index of the corresponding  $\square$ -formula. For simplicity, since this is enough in the case of *K*, in our implementation, we rely on the lower layer *LMF*. Please refer to Libal and Volpe (2016).

4.5.2. *Prefixed tableaux.* The popular *PT* proof format (Fitting 1972), which is used by various automated theorem provers (for example, Beckert and Goré (1997)), is, in the case of *K*, very close to that of *LS*. Therefore, support for it can be obtained in a very similar way. Its implementation, which has been described in Libal and Volpe (2016), also relies on *LMF* and mainly consists in inverting, with respect to *LS* the role of boxes and diamonds in the FPC and in letting tableau closure rules behave as sequent initial rules. This inversion is related to the fact that, despite *LS*, *PT* is a refutation method.

4.5.3. *Ordinary sequents.* As described in Section 2.1.5, *OS* systems differ in several ways from the previous systems. First, they do not have labels and second, they treat both  $\square$  and  $\diamond$ -formulas inside a single inference rule. For these reasons, the case of *OS* illustrates the use of the features of the framework *LMF\** in a more significant way already for the logic *K*.

In particular, the modal rule, which applies to all  $\diamond$ -formulas at once, can be emulated in our system by using multi-focusing. In addition, the relationship between the modal operators can be used in order to restrict the futures allowed: given a modal rule, all the  $\diamond$ -formulas occurring there are assigned the same future, which corresponds to the index of the only  $\square$ -formula.

$$\begin{array}{c}
 \frac{\frac{\frac{\frac{\frac{\frac{\frac{\diamond(P \wedge \neg Q), \diamond \neg P, [-Q, \neg P, Q]}{\diamond(P \wedge \neg Q), \diamond \neg P, [P \wedge \neg Q, \neg P, Q]}{\diamond(P \wedge \neg Q), \diamond \neg P, [\neg P, Q]}{\diamond(P \wedge \neg Q), \diamond \neg P, [Q]}{\diamond(P \wedge \neg Q), \diamond \neg P, \Box Q}}{\diamond(P \wedge \neg Q), \diamond \neg P \vee \Box Q}}{\diamond(P \wedge \neg Q) \vee (\diamond \neg P \vee \Box Q)}}{\diamond(P \wedge \neg Q), \diamond \neg P, [-Q, \neg P, Q]} \quad \diamond(P \wedge \neg Q), \diamond \neg P, [P, \neg P, Q]}
 \end{array}$$

Fig. 21. Nested sequent proof of axiom K.

The information required in an *OS* proof evidence is therefore

- for each application of a  $\Box_F$  inference rule, a list of indices of all affected  $\diamond$ -formulas;
- for the application of the initial rule, the corresponding index of the complementary literal.

The program which translates between *OS* and *LMF\** proof evidence needs to compute from the above information the relevant multi-focus indices as well as the present and future of sequents and formulas.

An *OS* node contains its index as well as a list of indices. This list is empty for all inference rules except for the modal rule, where it specifies the indices of all the  $\diamond$ -formulas that are affected, as well as for the initial rule, in which case the list contains a single index denoting the complementary literal.

In more technical terms, upon reaching the application of a modal rule in the *OS* proof evidence, the FPC program generates a proof evidence in the *LMF\** layer which contains a new inference for each  $\diamond$  formula in the *OS* proof evidence. This is required since there are no  $\diamond$  inference rules in *OS*. It then populates them with the same multi-focusing values as well as with the correct futures and presents.

Section 4.4 contains more information about the FPC specification.

4.5.4. *Nested sequents.* A more challenging example of using our framework is supporting nested sequent proof evidences. Here, we will also demonstrate how layers other than *LMF\** can be used in order to support proof formats.

Figure 21 shows a proof of the same theorem we have seen so far, this time by using the *NS* calculus which was mentioned in Section 2.1.6.

Unlike *OS* proofs, we can see a closer relationship to the *LS* calculus, which forms the basis of our framework. While the *OS* calculus has one modal rule, the *NS* calculus rules correspond, more or less to the ones in *LS*. This allows us to use directly the *LMF* layer for the certification of *NS* proofs.

We note though two differences between *NS* and *LS*. Nested sequents do not use labels and in order to index them, we need more than just the symbols *right*, *left*, *diaind* and  $\emptyset$ . We now index formulas using two separate indices: The first one is just the location of the sub-formula as before, while the second is the index of the nested sequent, as will be explained next.

$$\frac{\frac{(q)^{(ns \ (lind \ root) \ zb)}, ([p])^{(ns \ (rind \ root) \ (chld \ 1 \ zb))}}{(q)^{(ns \ (lind \ root) \ zb)}, (\Box p)^{(ns \ (rind \ root) \ zb)}}}{(q \vee \Box p)^{(ns \ root \ zb)}}$$

Fig. 22. An example of a nested sequent derivation and the corresponding indices.

We remind the reader that our framework is not based on labels but on correspondences between indices. Therefore, the translation between *NS* and *LMF* will only require a consistent mapping between the indices defined next and those of the *LMF* layer.

**Definition 4.6 (Indexing a nested sequents).** Indices of nested sequents within a sequent are defined recursively by

- *zb* is an index (of the top level nested sequent);
- if *ind* is an index of a nested sequent containing the nested sequents  $S_1, \dots, S_n$ , then  $(chld \ i \ ind)$  is an index denoting the nested sequent at position *i*.

The index of a sequent in *NS* will then be composed of a pair containing our regular index and the index which was just defined.

Figure 22 gives an example of a nested sequent derivation and the indices of sub-formulas.

In order to certify nested sequent proofs in our framework, we will use, as mentioned above, the *LMF* layer. We note that the *NS*  $\Box$  rule creates a new nested sequent, while the *NS*  $\Diamond$  rule adds the formula into an existing nested sequent. This is similar to the correspondence between  $\Box$  and  $\Diamond$  formulas we have defined in Definition 4.2. Our FPC specification for *NS* will indeed exploit this similarity and will translate between the indices of the two systems, *NS* and *LMF*.

Figure 23 shows the idea behind this translation. In order to keep track of the index translations, we add to the proof evidence an empty structure which denotes its state. This state is being updated along the certification process by the FPC specification.

In general, supporting nested sequent proof evidence for *K* is straightforward and does not require any knowledge of *LKF*. The only thing required is to be able to translate between the indices. Our use of the ‘polymorphic’ approach means that understanding of the *LKF* inference rules is still required. In Section 5, we will discuss an alternative approach that eliminates the need to understand the *LKF* calculus.

#### 4.6. Examples

In this section, we explain how our program can be executed on different examples. The examples described in the paper and others can be found in the testing section of the Checkers proof certifier. Checkers can be obtained on Github<sup>§§</sup>, or on Zenodo<sup>¶¶</sup>. It depends on the  $\lambda$ Prolog interpreter Teyjus (<http://teyjus.cs.umn.edu/>) and can be executed by running in a bash terminal:

<sup>§§</sup> <https://github.com/proofcert/checkers/tree/dalefest>.

<sup>¶¶</sup> <https://zenodo.org/record/1325924#.W2IWPHVfgWM>

```

nested_to_single
  (nested_cert (nested_state Map) I OI)
  (single_cert I' OI') Map :-
map_index Map I I',
map_index Map OI OI'.

single_to_nested Map
  (single_cert I' OI')
  (nested_cert (nested_state Map) I OI) :-
map_index Map I I',
map_index Map OI OI'.

orNeg_c Cert Form Cert' :-
  nested_to_single Cert Cert-s Map,
  orNeg_c Cert-s Form Cert-s',
  single_to_nested Map Cert-s' Cert'.

```

Fig. 23. Proof evidence transformation between *NS* and *LMF*.

```
$ ./prover-teyjus.sh arg
```

where the argument is the name of the  $\lambda$ Prolog module denoting the proof evidence one wishes to check.

We are currently also supporting the ELPI implementation of  $\lambda$ Prolog<sup>|||</sup>. For running examples using ELPI, please use

```
$ ./prover-elpi.sh arg
```

Unfortunately, some bugs in the implementations of  $\lambda$ Prolog forced us to associate the different examples to specific implementations. The examples starting with *ex-* can be executed with Teyjus, while the rest are better executed with ELPI.

## 5. Discussion and conclusion

In this paper, we have presented the implementation of a framework for certifying propositional modal logic proofs. The framework has been developed by following the general principles of the project ProofCert and as a module of the concrete implementation provided by Checkers. Our approach is based on the use of a layered architecture, which allowed us to design a modular framework capable of supporting many different proof formats. While layers provide a high level of flexibility, we observe that the need to translate between proof evidences in order to support proof evidence polymorphism gets increasingly more complex, the farther we get from the kernel. In addition, FPC specifications are still defined in terms of the *LKF* kernel, which requires an implementer to be familiar with *LKF*. A different approach would consist in having a distinct (concrete) kernel for each layer. However, this could compromise the trust one can place in such

<sup>|||</sup> <https://github.com/LPCIC/elpi>

```

orNeg_LMFc
(nested-cert node I 0 [H|_]))
(nested-cert (node H)).

```

Fig. 24. Proof evidence transformation between *NS* and *LMF*, by using virtual kernels.

layers and would go against the general principle of having kernels based on simple, well-known and low-level calculi. In addition, since different proof calculi might use different kernels, the property of proof universality would be lost.

In Chihani et al. (2017), it has been shown how it is possible to ‘host’ the classical calculus  $LKF^a$  on an intuitionistic focused kernel, by using a translation between the two logics. Along the lines of what has been proposed there, we are investigating the possibility of representing the different layers as ‘virtual’ kernels, built on top of a lower level kernel. We briefly illustrate the idea by showing how  $LMF^a$  (a version of *LMF* augmented with proper clerks and experts) can be ‘hosted’ on  $LKF^a$ . The calculus  $LMF^a$  is shown in Figure 25. The augmentation leading from *LMF* to  $LMF^a$  is similar in spirit to the one going from *LKF* to  $LKF^a$  and is obtained by adding control predicates to the base system. We remark that in the case of relational formulas, we do not need to store them with a significant index as we will never focus on them again. Now, we can provide a definition of the clerks and experts of  $LKF^a$  in terms of those given for  $LMF^a$ , as shown in Figure 26. Like in the approach of Section 4, and more generally in any attempt to find a classical first-order proof for a propositional modal formula, an adequate translation is required. The definition of Figure 26 goes therefore together with a simple translation from the polarized propositional modal language to the polarized first-order language, which basically maps each classical connective into the corresponding connective (by also preserving polarity) and translates  $\Box$  and  $\Diamond$  as in Section 3.3. We omit a full type declaration and just remark that in Figure 26,  $C$ ,  $C'$  and  $C''$  stand for *LMF* evidences, while  $\text{mod}$  and  $\text{tns}$  are constructors that, applied to an *LMF* evidence, produce an  $LKF$  evidence. Intuitively, we use them to distinguish between two phases along the construction of a proof: a phase dealing with connectives introduced by the translation of  $\Box$  or  $\Diamond$  (denoted by  $\text{tns}$ ) and a ‘normal’ one that does not involve the translation of modalities (denoted by  $\text{mod}$ ). By using such an inter-definition, an external user interested in certifying her proofs over *LMF* can assume that a kernel based on  $LMF^a$  indeed exists and only needs to define  $LMF^a$  clerks and experts. In the context of our framework, by relying on the same idea, we could base each kernel on the immediately lower one. This solution would allow for keeping only one trusted kernel –  $LKF^a$  – but would, at the same time, provide virtual kernels which can be used in order to write simpler FPC specifications for the different proof formats. As an example, Figure 24 shows how an FPC specification can be written for the system *NS*, in a framework that uses virtual kernels, and should be compared with the specification in Figure 23. As one can see, we can now use directly the clerks and experts of the *LMF* layer.

Besides further investigation in this direction, there are several ways in which this work can be extended. The design of the parametric devices of the framework has been driven by the ambition of being as comprehensive as possible in terms of formalisms captured. The modularity and parameterizability of the whole approach should make

ASYNCHRONOUS INTRODUCTION RULES

$$\frac{\Xi', \mathcal{G} \vdash \Theta \uparrow x : A, \Gamma \quad \Xi'', \mathcal{G} \vdash \Theta \uparrow x : B, \Gamma \quad \text{andNeg}_c(\Xi, \Xi', \Xi'')}{\Xi, \mathcal{G} \vdash \Theta \uparrow x : A \wedge B, \Gamma}$$

$$\frac{\Xi', \mathcal{G} \vdash \Theta \uparrow x : A, x : B, \Gamma \quad \text{orNeg}_c(\Xi, \Xi') \quad (\Xi' y), \mathcal{G} \cup \{\langle \text{relind}, R(x, y) \rangle\} \vdash \Theta \uparrow y : B, \Gamma \quad \text{box}_c(\Xi, \Xi')}{\Xi, \mathcal{G} \vdash \Theta \uparrow x : A \vee B, \Gamma} \quad \frac{\Xi', \mathcal{G} \cup \{\langle \text{relind}, R(x, y) \rangle\} \vdash \Theta \uparrow y : B, \Gamma \quad \text{box}_c(\Xi, \Xi')}{\Xi, \mathcal{G} \vdash \Theta \uparrow x : \Box B, \Gamma} \dagger$$

SYNCHRONOUS INTRODUCTION RULES

$$\frac{\Xi', \mathcal{G} \vdash \Theta \downarrow x : B_1 \quad \Xi'', \mathcal{G} \vdash \Theta \downarrow x : B_2 \quad \text{andPos}_e(\Xi, \Xi', \Xi'')}{\Xi, \mathcal{G} \vdash \Theta \downarrow x : B_1 \wedge^+ B_2}$$

$$\frac{\Xi', \mathcal{G} \vdash \Theta \downarrow x : B_i \quad \text{orPos}_e(\Xi, \Xi', i)}{\Xi, \mathcal{G} \vdash \Theta \downarrow x : B_1 \vee^+ B_2} \quad \frac{\Xi', \mathcal{G} \cup \{\langle \text{relind}, R(x, y) \rangle\} \vdash \Theta \downarrow y : B \quad \text{dia}_e(\Xi, y, \Xi')}{\Xi, \mathcal{G} \cup \{\langle \text{relind}, R(x, y) \rangle\} \vdash \Theta \downarrow x : \Diamond B}$$

IDENTITY RULES

$$\frac{\langle l, x : \neg B \rangle \in \Theta \quad \text{initial}_e(\Xi, l)}{\Xi, \mathcal{G} \vdash \Theta \downarrow x : B} \text{init}$$

STRUCTURAL RULES

$$\frac{\Xi', \mathcal{G} \vdash \Theta \uparrow x : B \quad \text{release}_e(\Xi, \Xi')}{\Xi, \mathcal{G} \vdash \Theta \downarrow x : B} \text{release} \quad \frac{\Xi', \mathcal{G} \vdash \Theta, \langle l, x : B \rangle \uparrow \Gamma \quad \text{store}_c(\Xi, x : B, l, \Xi')}{\Xi, \mathcal{G} \vdash \Theta \uparrow x : B, \Gamma} \text{store}$$

$$\frac{\Xi', \mathcal{G} \vdash \Theta \downarrow x : B \quad \langle l, x : B \rangle \in \Theta \quad \text{decide}_e(\Xi, l, \Xi')}{\Xi, \mathcal{G} \vdash \Theta \uparrow} \text{decide}$$

In *decide*,  $B$  is positive; in *release*,  $B$  is negative; in *store*,  $B$  is a positive formula or a negative literal; in *init*,  $B$  is a positive literal. The proviso  $\dagger$  specifies that  $y$  is different from  $x$  and does not occur in  $\Theta$  nor in  $\mathcal{G}$ .

Fig. 25.  $LMF^a$ : a focused labelled proof system for the modal logic  $K$ .

```

andNeg_LKFc (mod C) (mod C') (mod C'') :- andNeg_LMFc C C' C''.
orNeg_LKFc (tns C) (tns C').
orNeg_LKFc (mod C) (mod C') :- orNeg_LMFc C C'.
all_LKFc (mod C) (X \ tns (C' X)) :- box_LMFc C C'.

andPos_LKFe (tns C) (tns C) (mod C).
andPos_LKFe (mod C) (mod C') (mod C'') :- andPos_LMFe C C' C''.
orPos_LKFe (mod C) (mod C') LeftRight :- orPos_LMFe C C' LeftRight.
some_LKFe (mod C) Term (tns C') :- dia_LMFe C Term C'.

initial_LKFe (mod C) Index :- initial_LMFe C Index.
initial_LKFe (tns C) relind.

release_LKFe (mod C) (mod C') :- release_LMFe C C'.
store_LKFc (tns C) relind (mod C).
store_LKFc (mod C) Index (mod C') :- store_LMFc C Index C'.
decide_LKFe (mod C) Index (mod C') :- decide_LMFe C Index C'.

```

Fig. 26. The definition of  $LKF^a$  clerks and experts based on those given for  $LMF^a$ .



it possible, in fact, to consider other related approaches to modal proof theory, like hypersequent calculi (Avron 1996), e.g., by using a *present* parameter that is a multiset, representing external structural rules as operations on such a present, and viewing modal communication rules as a combination of relational and modal rules. The focused nature of the approach should also allow for certifying proofs coming from focused proof systems for modal logics, like the ones in Lellmann and Pimentel (2015), Chaudhuri et al. (2016), possibly by using a different polarization of formulas.

Orthogonally, we also aim at extending the approach to variants of the logic  $K$ . By applying the results in Marin et al. (2016), the extension to the logics characterized by the so-called geometric frames seems to be not too complex, although it might require some modifications to the current implementation (e.g., in the case of a logic defined, amongst the others, by the axiom of seriality  $D$ , a more complex notion of modal correspondence between  $\diamond$ -formulas and  $\Box$ -formulas, or labels, would be required).

We also notice that while this work was inspired by certification consisting in a strict emulation of original proofs, it is sometimes the case that only partial information about the proof to be checked is provided. We plan to complement the current implementation with a ‘relaxed’ version of the FPCs, such that it can also deal with incomplete proof evidences, similarly to what has been done in Libal and Volpe (2016) in order to check, e.g., free-variable tableau (Beckert and Goré 1997) proofs.

The kind of investigation done in this work also suggests us new directions to explore, more generally, in the context of the *Checkers* project. For instance, it seems interesting to consider the application of *Checkers* to objects composed from proofs coming from different proof calculi. One example of such objects are coalesced proofs, described in Doligez et al. (2014). In this work, a proof evidence is created by using modal theorem provers alongside first-order ones. A universal proof certifier such as *Checkers*, which is based on *LKF*, can attempt to use the different components in order to find a composed formal proof of the original theorem. One can also compare this goal to the one achieved by different ‘hammers’ (Blanchette and Paulson 2018), where specialized theorem provers are indirectly used in order to find a formal proof in a different calculus. The success and popularity of the different hammers makes a case in favour of further extensions of *Checkers*. An example of such an extension, beyond first-order and propositional modal proofs, is the support of proofs based on theories, such as arithmetic.

## Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions.

The work presented in this paper was partially funded by the ERC Advanced Grant ProofCert. Part of the work was carried out while the authors were at INRIA Saclay.

## References

- Andreoli, J. M. (1992). Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation* **2** (3) 297–347.

- Avron, A. (1996). The method of hypersequents in the proof theory of propositional non-classical logics. In: *Logic: From Foundations to Applications, European Logic Colloquium*, Oxford University Press, 1–32.
- Beckert, B. and Gor, R. (1997). Free-variable tableaux for propositional modal logics. *Studia Logica* **69** (1) 59–96.
- Benzmüller, C. and Woltzenlogel Paleo, B. (2015). Interacting with modal logics in the coq proof assistant. In: *Proceedings of the 10th International Computer Science Symposium in Russia*, Lecture Notes in Computer Science, vol. 9139, Springer, 398–411.
- Blackburn, P. and Van Benthem, J. (2007). Modal logic: A semantic perspective. In: Wolter, F., Blackburn, P. and van Benthem, J. (eds.) *Handbook of Modal Logic*, Elsevier, 1–82.
- Blanchette, J. C. and Paulson, L. C. (2018). ‘Hammering Away’. A Users Guide to Sledgehammer for Isabelle. Available at: <https://isabelle.in.tum.de/dist/doc/sledgehammer.pdf>
- Brünnler, K. (2009). Deep sequent systems for modal logic. *Archive for Mathematical Logic* **48** (6) 551–577.
- Chaudhuri, K., Marin, S. and Straßburger, L. (2016). Focused and synthetic nested sequents. In: Jacobs, B. and Löding, C. (eds.) *Foundations of Software Science and Computation Structures*, Lecture Notes in Computer Science, vol. 9634, Springer, 390–407.
- Cihani, Z., Libal, T. and Reis, G. (2015). The proof certifier checkers. In: *Automated Reasoning with Analytic Tableaux and Related Methods - 24th International Conference*, Lecture Notes in Computer Science **9323**, 201–210.
- Cihani, Z., Miller, D. and Renaud, F. (2017). A semantic framework for proof evidence. *Journal of Automated Reasoning* **59** (3) 287–330.
- de Bruijn, N.G. (1970). The mathematical language AUTOMATH, its usage, and some of its extensions. In: *Proceedings of the Symposium on Automatic Demonstration*. Lecture Notes in Mathematics 125, Springer, 29–61.
- Doligez, D., Kriener, J., Lamport, L., Libal, T. and Merz, S. (2014). Coalescing: Syntactic abstraction for reasoning in first-order modal logics. In: *Proceedings of “Automated Reasoning in Quantified Non-Classical Logics”*, EPiC Series in Computing, vol. 33, 1–16.
- Fitting, M. (1972). Tableau methods of proof for modal logics. *Notre Dame Journal of Formal Logic* **13** (2) 237–247.
- Fitting, M. (2007). Modal proof theory. In: Blackburn, P., van Benthem, J. and Wolter, F. (eds.) *Handbook of Modal Logic*, Elsevier, 85–138.
- Fitting, M. (2012). Prefixed tableaux and nested sequents. *Annals of Pure and Applied Logic* **163** (3) 291–313.
- Gabbay, D. (1996). *Labelled Deductive Systems*, Clarendon Press.
- Goré, R. and Ramanayake, R. (2012). Labelled tree sequents, tree hypersequents and nested (deep) sequents. In: *Proceedings of the 9th conference on “Advances in Modal Logic”*, College Publications, 279–299.
- Indrzejczak, A. (2010). *Natural Deduction, Hybrid Systems and Modal Logics*, Springer.
- Kashima, R. (1994). Cut-free sequent calculi for some tense logics. *Studia Logica* **53** (1) 119–136.
- Lellmann, B. (2015). Linear nested sequents, 2-sequents and hypersequents. In: *Proceedings of the 24th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, Lecture Notes in Computer Science, vol. 9323, Springer, 135–150.
- Lellmann, B. and Pimentel, E. (2015). Proof search in nested sequent calculi. In: *Proceedings of the 20th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, Lecture Notes in Computer Science, vol. 9450, Springer, 558–574.
- Liang, C. and Miller, D. (2009). Focusing and polarization in linear, intuitionistic, and classical logics. *Theoretical Computer Science* **410** (46) 4747–4768.

- Libal, T. and Volpe, M. (2016). Certification of prefixed tableau proofs for modal logic. In: *Proceedings of the 7th International Symposium on Games, Automata, Logics and Formal Verification*, EPTCS Series, vol. 226, 257–271.
- Marin, S., Miller, D. and Volpe, M. (2016). A focused framework for emulating modal proof systems. In: *Proceedings of the 11th Conference on 'Advances in Modal Logic'*, College Publications, 469–488.
- Miller, D. (2011). Proofcert: Broad spectrum proof certificates. An ERC Advanced Grant funded for the five years 2012–2016. Technical description, available online at: <http://www.lix.polytechnique.fr/Labo/Dale.Miller/ProofCert/ProofCert.pdf>
- Miller, D. and Nadathur, G. (2012). *Programming With Higher-Order Logic*, Cambridge University Press.
- Miller, D. and Volpe, M. (2015). Focused labeled proof systems for modal logic. In: *Proceedings of the 20th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, Lecture Notes in Computer Science, vol. 9450, Springer, 266–280.
- Negri, S. (2005). Proof analysis in modal logic. *Journal of Philosophical Logic* **34** (5–6) 507–544.
- Poggiolesi, F. (2011). *Gentzen Calculi for Modal Propositional Logic*, Springer.
- Stewart, C. and Stouppa, P. (2004). A systematic proof theory for several modal logics. In: *Proceedings of the 5th Conference on 'Advances in Modal logic'*, King's College Publications, 309–333.