A Reusable Isabelle/HOL Framework for Propositional Labelled Natural Deduction

Arthur Freitas Ramos David Barros Hulak
Ruy J. G. B. de Queiroz

June 25, 2026

Abstract

A reusable Isabelle/HOL framework presenting labelled natural deduction as a conservative refinement of ordinary natural deduction, with generic structural metatheory and several concrete label interpretations. We define ordinary intuitionistic propositional natural deduction, a parametric labelled refinement, prove erasure soundness (labelled derivations are ordinary derivations after erasing labels) and a controlled lifting theorem (ordinary derivations admit some labelling). Standard structural metatheory — weakening, exchange, substitution, cut admissibility — is established in both the ordinary and labelled systems.

The framework is instantiated three times. Unit labels recover ordinary ND as a special case via conservativity. Provenance labels carry sets of assumption indices and give both a sound over-approximation of assumption dependencies and a completeness result equating ordinary derivability with the existence of a provenance-labelled derivation whose filtered sub-context still derives the conclusion. A possible-world modal instance derives the K axiom schema and necessitation and is connected to Kripke validity; the modal fragment is presented as an example instantiation, not as a full labelled modal proof theory.

The development is fully constructive, contains no unfinished proof commands or oracle invocations, and is intended as a foundation other entries can cite.

Contents

1 What this entry proves

The main named results, with the theory in which each is established, are:

- `labelled_sound_under_erasure` (`Erasure`) — every labelled derivation erases to an ordinary derivation.
- `lifting_general`, `ordinary_derivations_admit_labelling` (`Label_Algebra`, `Lifting`) — ordinary derivations admit a labelling under the assumed lift-coherence law.
- `framework_adequacy`, `framework_adequacy_iff` (`Label_Algebra`) — the locale-level equivalence between labelled derivability up to a label and ordinary derivability after erasure.
- `unit_labels_conservative` (`Unit_Labels`) — the unit-labels instance recovers ordinary ND.
- `provenance_overapproximates_dependencies` (`Provenance_Labels`) — the provenance instance gives a sound over-approximation of assumption dependencies.
- `ordinary_admits_safe_provenance`, `ordinary_admits_tight_provenance`, `provenance_completeness` (`Provenance_Labels`) — the provenance instance is dependency-tracking complete: ordinary derivability is equivalent to the existence of a provenance-labelled derivation whose selected filtered context still derives the conclusion.
- `modal_K_schema`, `modal_necessitation`, `modal_K_kripke_valid` (`Modal_K_Schema`) — the worked modal instance, where the K axiom schema is derived uniformly and connected to Kripke validity.

2 Overview

Labelled deductive systems were introduced by Gabbay as a general way to attach structured information to formulae and rules [2]. The modal example in this entry follows the standard possible-world reading of modal labels, closely related to labelled proof-theoretic presentations such as Simpson's work on intuitionistic modal logic [3]. Labelled approaches to non-classical proof systems have been developed extensively by Viganò [4] and by Basin, Matthews, and Viganò [1]; our framework targets the simpler proof-theoretic core of propositional intuitionistic ND but recovers similar structural properties.

3 Label algebra and adequacy

The entry factors the parametric part of labelled derivability through a label-algebra locale. The locale records the proof obligations needed by any future label instance: that label constructors are uniform under erasure to the singleton type and that ordinary derivations can be lifted coherently over the labelled context. Its central theorem, `framework_adequacy`, states that labelled derivability up to a label is equivalent to ordinary derivability after erasing the context, with the chosen lift supplying the canonical labelled witness. The result is intentionally architectural rather than surprising: it packages the soundness/lifting obligations a new label instance must discharge, so that once those obligations are met the equivalence comes for free.

4 Modal K axiom and necessitation

The worked modal instance is presented as an example, not as a self-contained labelled modal proof theory, but it is strong enough to derive the K axiom schema uniformly in the formulae A, B, the accessibility relation R, and the world w. The theory `Modal_K_Schema.thy` records this as `modal_K_schema`, proves `modal_necessitation` under the standard side-condition that necessitation applies only to formulae derivable from the empty context at every world and relation, and connects the labelled derivation to Kripke semantics through `modal_K_kripke_valid`.

5 Provenance completeness

The provenance instance is the entry's main dependency-tracking result. It is adequate in a strong sense: every ordinary derivation lifts to a labelled derivation whose provenance is contained in the indices of the ordinary context and whose selected sub-context is still sufficient to re-derive the conclusion. The relevant named theorems are:

- `ordinary_admits_safe_provenance` — records the index-bound and well-formedness properties of the lifted provenance derivation.
- `ordinary_admits_tight_provenance` — strengthens this to a sufficient filtered sub-context.
- `provenance_completeness` — states the resulting equivalence between ordinary derivability and provenance-labelled derivability with a sufficient filtered context.

theory *Labelled-Formulas*
  imports *Main*
begin

This theory fixes the propositional formula language and the elementary operations that erase labels from labelled formulae, databases, and list contexts. Labels are deliberately represented by a product component, so the logical shape of a labelled formula is recovered by taking its second projection.

datatype *'a fm* =
    Atom 'a
  | *Bot*
  | *Imp 'a fm 'a fm*
  | *Con 'a fm 'a fm*
  | *Dis 'a fm 'a fm*

type-synonym *('l, 'a) lfm* = *'l* × *'a fm*

definition *erase-lfm* :: *('l, 'a) lfm* ⇒ *'a fm* **where**
  erase-lfm x = *snd x*

definition *erase-db* :: *('l, 'a) lfm set* ⇒ *'a fm set* **where**
  erase-db Γ = *snd* ‘ Γ

definition *erase-ctx* :: *('l, 'a) lfm list* ⇒ *'a fm list* **where**
  erase-ctx Γ = *map snd* Γ

lemma *erase-lfm-simps* [*simp*]:
  erase-lfm (*l*, *A*) = *A*
  ⟨*proof*⟩

lemma *erase-ctx-Nil* [*simp*]:
  erase-ctx [] = []
  ⟨*proof*⟩

lemma *erase-ctx-Cons* [*simp*]:
  erase-ctx (*x* # Γ) = *erase-lfm x* # *erase-ctx* Γ
  ⟨*proof*⟩

lemma *erase-ctx-append* [*simp*]:
  erase-ctx (Γ @ Δ) = *erase-ctx* Γ @ *erase-ctx* Δ
  ⟨*proof*⟩

lemma *erase-db-insert* [*simp*]:
  erase-db (*insert x* Γ) = *insert* (*erase-lfm x*) (*erase-db* Γ)
  ⟨*proof*⟩

lemma *erase-db-Un* [*simp*]:
  erase-db (Γ ∪ Δ) = *erase-db* Γ ∪ *erase-db* Δ

⟨*proof*⟩

end

theory *Natural-Deduction*
  imports *Labelled-Formulas*
begin

This theory defines ordinary intuitionistic propositional natural deduction over list contexts. The structural metatheory is stated in terms of the set of assumptions represented by a list, which keeps exchange and weakening explicit while avoiding dependence on a particular context order.

primrec *fm-subst* :: $('a \Rightarrow 'b\ fm) \Rightarrow 'a\ fm \Rightarrow 'b\ fm$ **where**
  fm-subst σ (*Atom* p) = $\sigma\ p$
| *fm-subst* σ *Bot* = *Bot*
| *fm-subst* σ (*Imp* $A\ B$) = *Imp* (*fm-subst* σ A) (*fm-subst* σ B)
| *fm-subst* σ (*Con* $A\ B$) = *Con* (*fm-subst* σ A) (*fm-subst* σ B)
| *fm-subst* σ (*Dis* $A\ B$) = *Dis* (*fm-subst* σ A) (*fm-subst* σ B)

inductive *derives* :: $'a\ fm\ list \Rightarrow 'a\ fm \Rightarrow bool$ (- ⊢ - [50, 50] 50)
where
  Assm: $A \in set\ \Gamma \Longrightarrow \Gamma \vdash A$
| *BotE*: $\Gamma \vdash Bot \Longrightarrow \Gamma \vdash A$
| *ImpI*: $A\ \#\ \Gamma \vdash B \Longrightarrow \Gamma \vdash Imp\ A\ B$
| *ImpE*: $\Gamma \vdash Imp\ A\ B \Longrightarrow \Gamma \vdash A \Longrightarrow \Gamma \vdash B$
| *ConI*: $\Gamma \vdash A \Longrightarrow \Gamma \vdash B \Longrightarrow \Gamma \vdash Con\ A\ B$
| *ConE1*: $\Gamma \vdash Con\ A\ B \Longrightarrow \Gamma \vdash A$
| *ConE2*: $\Gamma \vdash Con\ A\ B \Longrightarrow \Gamma \vdash B$
| *DisI1*: $\Gamma \vdash A \Longrightarrow \Gamma \vdash Dis\ A\ B$
| *DisI2*: $\Gamma \vdash B \Longrightarrow \Gamma \vdash Dis\ A\ B$
| *DisE*: $\Gamma \vdash Dis\ A\ B \Longrightarrow A\ \#\ \Gamma \vdash C \Longrightarrow B\ \#\ \Gamma \vdash C \Longrightarrow \Gamma \vdash C$

lemma *weakening*:
  assumes $\Gamma \vdash A$ **and** $set\ \Gamma \subseteq set\ \Delta$
  shows $\Delta \vdash A$
  ⟨*proof*⟩

lemma *weakening-cons*:
  assumes $\Gamma \vdash A$
  shows $B\ \#\ \Gamma \vdash A$
  ⟨*proof*⟩

lemma *exchange*:
  assumes $A\ \#\ B\ \#\ \Gamma \vdash C$
  shows $B\ \#\ A\ \#\ \Gamma \vdash C$
  ⟨*proof*⟩

lemma *cut-general*:

assumes $\Gamma \vdash A$ **and** $\Delta \vdash B$ **and** *set* $\Delta \subseteq$ *insert* A (*set* Γ)
shows $\Gamma \vdash B$
⟨*proof*⟩

lemma *cut*:
assumes $\Gamma \vdash A$ **and** $A \# \Gamma \vdash B$
shows $\Gamma \vdash B$
⟨*proof*⟩

theorem *deduction-theorem*:
$A \# \Gamma \vdash B \longleftrightarrow \Gamma \vdash Imp\ A\ B$
⟨*proof*⟩

lemma *substitution-atom*:
assumes $\Gamma \vdash A$
shows *map* (*fm-subst* σ) $\Gamma \vdash$ *fm-subst* $\sigma\ A$
⟨*proof*⟩

end

theory *Labelled-Natural-Deduction*
imports *Natural-Deduction*
begin

This theory introduces the parametric labelled natural-deduction kernel. A locale fixes inert label constructors for the logical rules, but imposes no equations on them; concrete label disciplines are supplied later by interpretation.

locale *label-structure* =
fixes *app* :: $'l \Rightarrow 'l \Rightarrow 'l$
and *lam* :: $'l \Rightarrow 'l \Rightarrow 'l$
and *pair* :: $'l \Rightarrow 'l \Rightarrow 'l$
and *fst-l* :: $'l \Rightarrow 'l$
and *snd-l* :: $'l \Rightarrow 'l$
and *abort* :: $'l \Rightarrow 'l$
and *inl* :: $'l \Rightarrow 'l$
and *inr* :: $'l \Rightarrow 'l$
and *cases* :: $'l \Rightarrow 'l \Rightarrow 'l \Rightarrow 'l \Rightarrow 'l \Rightarrow 'l$
begin

inductive *lderives* :: $('l, 'a)$ *lfm list* $\Rightarrow$ $('l, 'a)$ *lfm* $\Rightarrow$ *bool*
(- $\vdash L$ - [*50*, *50*] *50*)
where
LAssm: $x \in$ *set* $\Gamma \Longrightarrow \Gamma \vdash L\ x$
| *LBotE*: $\Gamma \vdash L\ (l, Bot) \Longrightarrow \Gamma \vdash L\ (abort\ l, A)$
| *LImpI*: $(l, A) \# \Gamma \vdash L\ (m, B) \Longrightarrow \Gamma \vdash L\ (lam\ l\ m, Imp\ A\ B)$
| *LImpE*: $\Gamma \vdash L\ (l, Imp\ A\ B) \Longrightarrow \Gamma \vdash L\ (m, A) \Longrightarrow \Gamma \vdash L\ (app\ l\ m, B)$
| *LConI*: $\Gamma \vdash L\ (l, A) \Longrightarrow \Gamma \vdash L\ (m, B) \Longrightarrow \Gamma \vdash L\ (pair\ l\ m, Con\ A\ B)$

| *LConE1*: $\Gamma \vdash L\ (l,\ Con\ A\ B) \Longrightarrow \Gamma \vdash L\ (fst\text{-}l\ l,\ A)$
| *LConE2*: $\Gamma \vdash L\ (l,\ Con\ A\ B) \Longrightarrow \Gamma \vdash L\ (snd\text{-}l\ l,\ B)$
| *LDisI1*: $\Gamma \vdash L\ (l,\ A) \Longrightarrow \Gamma \vdash L\ (inl\ l,\ Dis\ A\ B)$
| *LDisI2*: $\Gamma \vdash L\ (l,\ B) \Longrightarrow \Gamma \vdash L\ (inr\ l,\ Dis\ A\ B)$
| *LDisE*:
$\Gamma \vdash L\ (l,\ Dis\ A\ B) \Longrightarrow$
$(m,\ A)\ \#\ \Gamma \vdash L\ (n,\ C) \Longrightarrow$
$(p,\ B)\ \#\ \Gamma \vdash L\ (q,\ C) \Longrightarrow$
$\Gamma \vdash L\ (cases\ l\ m\ n\ p\ q,\ C)$

lemma *labelled-weakening*:
assumes $\Gamma \vdash L\ x$ **and** $set\ \Gamma \subseteq set\ \Delta$
shows $\Delta \vdash L\ x$
⟨*proof*⟩

lemma *labelled-weakening-cons*:
assumes $\Gamma \vdash L\ x$
shows $y\ \#\ \Gamma \vdash L\ x$
⟨*proof*⟩

lemma *labelled-exchange*:
assumes $x\ \#\ y\ \#\ \Gamma \vdash L\ z$
shows $y\ \#\ x\ \#\ \Gamma \vdash L\ z$
⟨*proof*⟩

lemma *labelled-cut-general*:
assumes $\Gamma \vdash L\ x$
and $\Delta \vdash L\ y$
and $set\ \Delta \subseteq insert\ x\ (set\ \Sigma)$
and $set\ \Gamma \subseteq set\ \Sigma$
shows $\Sigma \vdash L\ y$
⟨*proof*⟩

lemma *labelled-cut*:
assumes $\Gamma \vdash L\ x$ **and** $x\ \#\ \Delta \vdash L\ y$
shows $\Gamma\ @\ \Delta \vdash L\ y$
⟨*proof*⟩

lemma *assumption-substitution-lift*:
assumes $x \in set\ \Gamma$
shows $(fst\ x,\ fm\text{-}subst\ \sigma\ (snd\ x)) \in set\ (map\ (map\text{-}prod\ id\ (fm\text{-}subst\ \sigma))\ \Gamma)$
⟨*proof*⟩

lemma *labelled-substitution-atom-aux*:
assumes $\Gamma \vdash L\ x$
shows $map\ (map\text{-}prod\ id\ (fm\text{-}subst\ \sigma))\ \Gamma \vdash L\ (fst\ x,\ fm\text{-}subst\ \sigma\ (snd\ x))$
⟨*proof*⟩

lemma *labelled-substitution-atom*:

assumes $\Gamma \vdash L\ (l,\ A)$
shows *map* (*map-prod id* (*fm-subst* σ)) $\Gamma \vdash L\ (l,$ *fm-subst* $\sigma\ A)$
⟨*proof*⟩

end

end

theory *Erasure*
imports *Labelled-Natural-Deduction*
begin

This theory proves that labels are conservative annotations for the parametric kernel: every labelled derivation erases to an ordinary natural deduction derivation of the underlying formula from the erased context.

context *label-structure*
begin

theorem *labelled-sound-under-erasure*:
assumes $\Gamma \vdash L\ x$
shows *erase-ctx* $\Gamma \vdash$ *erase-lfm* x
⟨*proof*⟩

lemmas *erasure-sound* = *labelled-sound-under-erasure*

end

end

theory *Label-Algebra*
imports *Erasure*
begin

The label-algebra layer packages the two generic adequacy requirements used by the concrete interpretations. Label erasure targets *unit*, so every erasure map is extensionally the unique one. The lift operation may inspect the labelled context, which is needed for calculi whose assumption labels carry semantic data.

lemma *erase-ctx-mem-label*:
assumes $A \in$ *set* (*erase-ctx* Γ)
obtains l **where** $(l,\ A) \in$ *set* Γ
⟨*proof*⟩

context *label-structure*
begin

lemma *lifting-general-aux*:

assumes $\Delta \vdash A$ **and** *erase-ctx* $\Gamma = \Delta$
shows $\exists\, l.\ \Gamma \vdash L\ (l,\ A)$
⟨*proof*⟩

theorem *lifting-general*:
assumes *erase-ctx* $\Gamma \vdash A$
shows $\exists\, l.\ \Gamma \vdash L\ (l,\ A)$
⟨*proof*⟩

end

locale *label-algebra* =
fixes *app* :: $'l \Rightarrow 'l \Rightarrow 'l$
and *lam* :: $'l \Rightarrow 'l \Rightarrow 'l$
and *pair* :: $'l \Rightarrow 'l \Rightarrow 'l$
and *fst-l* :: $'l \Rightarrow 'l$
and *snd-l* :: $'l \Rightarrow 'l$
and *abort* :: $'l \Rightarrow 'l$
and *inl* :: $'l \Rightarrow 'l$
and *inr* :: $'l \Rightarrow 'l$
and *cases* :: $'l \Rightarrow 'l \Rightarrow 'l \Rightarrow 'l \Rightarrow 'l \Rightarrow 'l$
and *lderives* :: $('l,\ 'f)$ *lfm list* $\Rightarrow$ $('l,\ 'f)$ *lfm* $\Rightarrow$ *bool*
$(\text{-} \vdash L\ \text{-}\ [50,\ 50]\ 50)$
and *erase-label* :: $'l \Rightarrow$ *unit*
and *lift-label* :: $('l,\ 'f)$ *lfm list* $\Rightarrow$ $'f$ *fm* $\Rightarrow$ $'l$
assumes *erase-label-unique* [*simp*]: *erase-label* $l = ()$
and *erasure-sound-law*:
$\Gamma \vdash L\ x \Longrightarrow$ *erase-ctx* $\Gamma \vdash$ *erase-lfm* x
and *lift-coherence*:
erase-ctx $\Gamma \vdash A \Longrightarrow \Gamma \vdash L\ (\textit{lift-label}\ \Gamma\ A,\ A)$
begin

definition *erase-judgement* ::
$(('l,\ 'f)$ *lfm list* $\times$ $('l,\ 'f)$ *lfm*$) \Rightarrow (' f$ *fm list* $\times\ 'f$ *fm*$)$
where *erase-judgement* $J = (\textit{erase-ctx}\ (\textit{fst}\ J),\ \textit{erase-lfm}\ (\textit{snd}\ J))$

lemma *erase-judgement-simps* [*simp*]:
erase-judgement $(\Gamma,\ (l,\ A)) = (\textit{erase-ctx}\ \Gamma,\ A)$
⟨*proof*⟩

lemma *erasure-uniformity*:
erase-label $(\textit{app}\ l\ m) = ()$
erase-label $(\textit{lam}\ l\ m) = ()$
erase-label $(\textit{pair}\ l\ m) = ()$
erase-label $(\textit{fst-l}\ l) = ()$
erase-label $(\textit{snd-l}\ l) = ()$
erase-label $(\textit{abort}\ l) = ()$
erase-label $(\textit{inl}\ l) = ()$
erase-label $(\textit{inr}\ l) = ()$

erase-label (*cases* *l* *m* *n* *p* *q*) = ()
⟨*proof*⟩

lemma *erasure-uniformity-judgement*:
erase-judgement (Γ, (*app* *l* *m*, *A*)) = (*erase-ctx* Γ, *A*)
erase-judgement (Γ, (*lam* *l* *m*, *Imp* *A* *B*)) = (*erase-ctx* Γ, *Imp* *A* *B*)
erase-judgement (Γ, (*pair* *l* *m*, *Con* *A* *B*)) = (*erase-ctx* Γ, *Con* *A* *B*)
erase-judgement (Γ, (*fst-l* *l*, *A*)) = (*erase-ctx* Γ, *A*)
erase-judgement (Γ, (*snd-l* *l*, *B*)) = (*erase-ctx* Γ, *B*)
erase-judgement (Γ, (*abort* *l*, *A*)) = (*erase-ctx* Γ, *A*)
erase-judgement (Γ, (*inl* *l*, *Dis* *A* *B*)) = (*erase-ctx* Γ, *Dis* *A* *B*)
erase-judgement (Γ, (*inr* *l*, *Dis* *A* *B*)) = (*erase-ctx* Γ, *Dis* *A* *B*)
erase-judgement (Γ, (*cases* *l* *m* *n* *p* *q*, *C*)) = (*erase-ctx* Γ, *C*)
⟨*proof*⟩

theorem *framework-adequacy*:
fixes Γ*L* :: ('*l*, '*f*) *lfm list*
and *A* :: '*f fm*
shows (∃ *l*. Γ*L* ⊢*L* (*l*, *A*)) ⟷
(*erase-ctx* Γ*L* ⊢ *A* ∧ Γ*L* ⊢*L* (*lift-label* Γ*L* *A*, *A*))
⟨*proof*⟩

corollary *framework-adequacy-iff*:
fixes Γ*L* :: ('*l*, '*f*) *lfm list*
and *A* :: '*f fm*
shows (∃ *l*. Γ*L* ⊢*L* (*l*, *A*)) ⟷ *erase-ctx* Γ*L* ⊢ *A*
⟨*proof*⟩

end

end

theory *Lifting*
imports *Label-Algebra*
begin

This theory supplies a concrete datatype of proof labels and interprets the abstract label-structure locale with its constructors. With these labels, ordinary derivations can always be decorated to obtain a labelled derivation over any labelled context whose erasure is the ordinary context.

datatype *label* =
Hyp nat
| *App label label*
| *Lam label label*
| *Pair label label*
| *FstL label*
| *SndL label*
| *Inl label*

```
  | Inr label
  | Cases label label label label label
  | Abort label

interpretation default-labels: label-structure
  where app = App
    and lam = Lam
    and pair = Pair
    and fst-l = FstL
    and snd-l = SndL
    and abort = Abort
    and inl = Inl
    and inr = Inr
    and cases = Cases
  ⟨proof⟩

definition default-erase-label :: label ⇒ unit where
  default-erase-label l = ()

definition default-lift-label :: (label, 'a) lfm list ⇒ 'a fm ⇒ label where
  default-lift-label Γ A =
    (SOME l. default-labels.lderives Γ (l, A))

interpretation default-labels: label-algebra
  where app = App
    and lam = Lam
    and pair = Pair
    and fst-l = FstL
    and snd-l = SndL
    and abort = Abort
    and inl = Inl
    and inr = Inr
    and cases = Cases
    and lderives = default-labels.lderives
    and erase-label = default-erase-label
    and lift-label = default-lift-label
⟨proof⟩

lemma erase-ctx-mem-ex:
  assumes A ∈ set (erase-ctx Γ)
  shows ∃ l. (l, A) ∈ set Γ
  ⟨proof⟩

lemma erase-ctx-mem-witness:
  assumes A ∈ set (erase-ctx Γ)
  obtains l where (l, A) ∈ set Γ
  ⟨proof⟩

lemma ordinary-derivations-have-labelling-aux:
```

assumes $\Delta \vdash A$ **and** *erase-ctx* $\Gamma = \Delta$
shows $\exists\, l.$ *default-labels.lderives* Γ (l, A)
⟨*proof*⟩

theorem *ordinary-derivations-have-labelling*:
assumes *erase-ctx* $\Gamma \vdash A$
shows $\exists\, l.$ *default-labels.lderives* Γ (l, A)
⟨*proof*⟩

theorem *ordinary-derivations-admit-labelling*:
assumes *erase-ctx* $\Gamma \vdash A$
shows $\exists\, l.$ *default-labels.lderives* Γ (l, A)
⟨*proof*⟩

end

theory *Unit-Labels*
imports *Label-Algebra*
begin

This theory instantiates the labelled kernel with a singleton label type. Since all label constructors return the unique label, labelled derivability over the lifted context is equivalent to ordinary natural deduction.

datatype *unit-label* = *Star*

interpretation *unit-labels*: *label-structure*
where *app* = λ- -. *Star*
and *lam* = λ- -. *Star*
and *pair* = λ- -. *Star*
and *fst-l* = λ-. *Star*
and *snd-l* = λ-. *Star*
and *abort* = λ-. *Star*
and *inl* = λ-. *Star*
and *inr* = λ-. *Star*
and *cases* = λ- - - - -. *Star*
⟨*proof*⟩

definition *unit-erase-label* :: *unit-label* ⇒ *unit* **where**
unit-erase-label l = ()

definition *unit-lift-label* :: (*unit-label*, *'a*) *lfm list* ⇒ *'a fm* ⇒ *unit-label* **where**
unit-lift-label Γ *A* = *Star*

interpretation *unit-labels*: *label-algebra*
where *app* = λ- -. *Star*
and *lam* = λ- -. *Star*
and *pair* = λ- -. *Star*
and *fst-l* = λ-. *Star*

and *snd-l* = λ-. *Star*
and *abort* = λ-. *Star*
and *inl* = λ-. *Star*
and *inr* = λ-. *Star*
and *cases* = λ- - - - -. *Star*
and *lderives* = *unit-labels.lderives*
and *erase-label* = *unit-erase-label*
and *lift-label* = *unit-lift-label*
⟨*proof*⟩

definition *unit-lift* :: *'a fm list* ⇒ (*unit-label* × *'a fm*) *list* **where**
unit-lift Γ = *map* (λ*A*. (*Star*, *A*)) Γ

lemma *erase-ctx-unit-lift* [*simp*]:
erase-ctx (*unit-lift* Γ) = Γ
⟨*proof*⟩

lemma *unit-labels-from-ordinary*:
assumes Γ ⊢ *A*
shows *unit-labels.lderives* (*unit-lift* Γ) (*Star*, *A*)
⟨*proof*⟩

theorem *unit-labels-conservative*:
Γ ⊢ *A* ⟷ *unit-labels.lderives* (*unit-lift* Γ) (*Star*, *A*)
⟨*proof*⟩

end

theory *Provenance-Labels*
imports *Label-Algebra*
begin

This theory instantiates labels by sets of natural-number assumption indices. Application and pairing combine provenance by union, implication introduction removes the discharged assumption label, and disjunction elimination records the analysed disjunction together with the branch dependencies that survive discharge.

type-synonym *prov* = *nat set*

interpretation *provenance*: *label-structure*
where *app* = λ(*S*::*prov*) *T*. *S* ∪ *T*
and *lam* = λ(*S*::*prov*) *T*. *T* − *S*
and *pair* = λ(*S*::*prov*) *T*. *S* ∪ *T*
and *fst-l* = λ(*S*::*prov*). *S*
and *snd-l* = λ(*S*::*prov*). *S*
and *abort* = λ(*S*::*prov*). *S*
and *inl* = λ(*S*::*prov*). *S*
and *inr* = λ(*S*::*prov*). *S*

```
  and cases = λ(S::prov) (M::prov) (N::prov) (P::prov) (Q::prov).
    S ∪ (N − M) ∪ (Q − P)
 ⟨proof⟩

definition provenance-erase-label :: prov ⇒ unit where
 provenance-erase-label S = ()

definition provenance-lift-label :: (prov, 'a) lfm list ⇒ 'a fm ⇒ prov where
 provenance-lift-label Γ A =
   (SOME S. provenance.lderives Γ (S, A))

interpretation provenance: label-algebra
 where app = λ(S::prov) T. S ∪ T
   and lam = λ(S::prov) T. T − S
   and pair = λ(S::prov) T. S ∪ T
   and fst-l = λ(S::prov). S
   and snd-l = λ(S::prov). S
   and abort = λ(S::prov). S
   and inl = λ(S::prov). S
   and inr = λ(S::prov). S
   and cases = λ(S::prov) (M::prov) (N::prov) (P::prov) (Q::prov).
     S ∪ (N − M) ∪ (Q − P)
   and lderives = provenance.lderives
   and erase-label = provenance-erase-label
   and lift-label = provenance-lift-label
⟨proof⟩

fun indexed :: nat ⇒ 'a fm list ⇒ (prov × 'a fm) list where
 indexed n [] = []
| indexed n (A # Γ) = ({n}, A) # indexed (Suc n) Γ

definition ctx-prov :: (prov, 'a) lfm list ⇒ prov where
 ctx-prov Γ = ⋃(fst ‘ set Γ)

lemma ctx-prov-Cons [simp]:
 ctx-prov ((M, A) # Γ) = M ∪ ctx-prov Γ
 ⟨proof⟩

lemma erase-ctx-indexed [simp]:
 erase-ctx (indexed n Γ) = Γ
 ⟨proof⟩

lemma finite-ctx-prov-indexed [simp]:
 finite (ctx-prov (indexed n Γ))
 ⟨proof⟩

definition filter-indexed-by :: nat set ⇒ 'a fm list ⇒ 'a fm list where
 filter-indexed-by S Γ =
   map snd (filter (λ(i, A). i ∈ S) (zip [0..<length Γ] Γ))
```

lemma *filter-indexed-by-mono*:
assumes $S \subseteq T$
shows *set* (*filter-indexed-by* S Γ) $\subseteq$ *set* (*filter-indexed-by* T Γ)
⟨*proof*⟩

lemma *filter-indexed-by-subset-ctx*:
shows *set* (*filter-indexed-by* S Γ) $\subseteq$ *set* Γ
⟨*proof*⟩

lemma *filter-indexed-by-empty* [*simp*]:
filter-indexed-by {} Γ = []
⟨*proof*⟩

lemma *filter-indexed-by-union*:
set (*filter-indexed-by* ($S \cup T$) Γ) =
set (*filter-indexed-by* S Γ) $\cup$ *set* (*filter-indexed-by* T Γ)
⟨*proof*⟩

definition *filter-labeled-by* :: *prov* ⇒ (*prov* × *'a fm*) *list* ⇒ *'a fm list* **where**
filter-labeled-by S Γ =
map snd (*filter* (λ(M, A). $M \cap S \neq \{\}$) Γ)

definition *nonempty-prov-ctx* :: (*prov* × *'a fm*) *list* ⇒ *bool* **where**
nonempty-prov-ctx $\Gamma \longleftrightarrow$ (∀ $x \in$ *set* Γ. *fst* $x \neq \{\}$)

lemma *nonempty-prov-ctx-Cons* [*simp*]:
nonempty-prov-ctx ((S, A) # Γ) $\longleftrightarrow$ $S \neq \{\} \wedge$ *nonempty-prov-ctx* Γ
⟨*proof*⟩

lemma *nonempty-prov-ctx-indexed* [*simp*]:
nonempty-prov-ctx (*indexed* n Γ)
⟨*proof*⟩

lemma *filter-labeled-by-mono*:
assumes $S \subseteq T$
shows *set* (*filter-labeled-by* S Γ) $\subseteq$ *set* (*filter-labeled-by* T Γ)
⟨*proof*⟩

lemma *filter-labeled-by-union-left*:
set (*filter-labeled-by* S Γ) $\subseteq$ *set* (*filter-labeled-by* ($S \cup T$) Γ)
⟨*proof*⟩

lemma *filter-labeled-by-union-right*:
set (*filter-labeled-by* T Γ) $\subseteq$ *set* (*filter-labeled-by* ($S \cup T$) Γ)
⟨*proof*⟩

lemma *ctx-prov-contains-label*:
assumes (M, A) $\in$ *set* Γ

shows $M \subseteq$ *ctx-prov* Γ
⟨*proof*⟩

lemma *filter-labeled-by-minus-fresh*:
assumes $M \cap$ *ctx-prov* $\Gamma = \{\}$
shows *set* (*filter-labeled-by* $N\ \Gamma$) $\subseteq$ *set* (*filter-labeled-by* $(N - M)\ \Gamma$)
⟨*proof*⟩

lemma *filter-labeled-by-cons-minus-subset*:
assumes $M \cap$ *ctx-prov* $\Gamma = \{\}$
shows *set* (*filter-labeled-by* $N\ ((M, A) \# \Gamma)) \subseteq$
set $(A \#$ *filter-labeled-by* $(N - M)\ \Gamma)$
⟨*proof*⟩

lemma *filter-labeled-by-cons-minus-mono-subset*:
assumes $M \cap$ *ctx-prov* $\Gamma = \{\}$ **and** $N - M \subseteq U$
shows *set* (*filter-labeled-by* $N\ ((M, A) \# \Gamma)) \subseteq$
set $(A \#$ *filter-labeled-by* $U\ \Gamma)$
⟨*proof*⟩

lemma *filter-labeled-by-indexed-upt*:
filter-labeled-by S (*indexed* $n\ \Gamma$) =
map snd (*filter* $(\lambda(i, A).\ i \in S)$ (*zip* $[n..<n + \mathit{length}\ \Gamma]\ \Gamma$))
⟨*proof*⟩

lemma *filter-labeled-by-indexed-0*:
filter-labeled-by S (*indexed* $0\ \Gamma$) = *filter-indexed-by* $S\ \Gamma$
⟨*proof*⟩

inductive *prov-safe* :: (*prov* × *'a fm*) *list* ⇒ *prov* × *'a fm* ⇒ *bool*
(- ⊢*P* - [50, 50] 50)
where
PAssm:
$x \in$ *set* $\Gamma \Longrightarrow \Gamma \vdash P\ x$
| *PBotE*:
$\Gamma \vdash P\ (S, \mathit{Bot}) \Longrightarrow \Gamma \vdash P\ (S, A)$
| *PImpI*:
$(M, A) \# \Gamma \vdash P\ (N, B) \Longrightarrow$
$M \cap$ *ctx-prov* $\Gamma = \{\} \Longrightarrow$
$M \neq \{\} \Longrightarrow$
$\Gamma \vdash P\ (N - M, \mathit{Imp}\ A\ B)$
| *PImpE*:
$\Gamma \vdash P\ (S, \mathit{Imp}\ A\ B) \Longrightarrow$
$\Gamma \vdash P\ (T, A) \Longrightarrow$
$\Gamma \vdash P\ (S \cup T, B)$
| *PConI*:
$\Gamma \vdash P\ (S, A) \Longrightarrow$
$\Gamma \vdash P\ (T, B) \Longrightarrow$
$\Gamma \vdash P\ (S \cup T, \mathit{Con}\ A\ B)$

| *PConE1*:
  $\Gamma \vdash P\ (S,\ Con\ A\ B) \Longrightarrow \Gamma \vdash P\ (S,\ A)$
| *PConE2*:
  $\Gamma \vdash P\ (S,\ Con\ A\ B) \Longrightarrow \Gamma \vdash P\ (S,\ B)$
| *PDisI1*:
  $\Gamma \vdash P\ (S,\ A) \Longrightarrow \Gamma \vdash P\ (S,\ Dis\ A\ B)$
| *PDisI2*:
  $\Gamma \vdash P\ (S,\ B) \Longrightarrow \Gamma \vdash P\ (S,\ Dis\ A\ B)$
| *PDisE*:
  $\Gamma \vdash P\ (S,\ Dis\ A\ B) \Longrightarrow$
  $(M,\ A)\ \#\ \Gamma \vdash P\ (N,\ C) \Longrightarrow$
  $(P,\ B)\ \#\ \Gamma \vdash P\ (Q,\ C) \Longrightarrow$
  $M \cap ctx\text{-}prov\ \Gamma = \{\} \Longrightarrow M \neq \{\} \Longrightarrow$
  $P \cap ctx\text{-}prov\ \Gamma = \{\} \Longrightarrow P \neq \{\} \Longrightarrow$
  $M \cap P = \{\} \Longrightarrow$
  $\Gamma \vdash P\ (S \cup (N - M) \cup (Q - P),\ C)$

lemma *provenance-labels-in-context*:
  assumes *provenance.lderives* $\Gamma\ x$
  shows $fst\ x \subseteq ctx\text{-}prov\ \Gamma$
  ⟨*proof*⟩

lemma *ctx-prov-indexed-bound*:
  assumes $i \in ctx\text{-}prov\ (indexed\ n\ \Gamma)$
  shows $i < n + length\ \Gamma$
  ⟨*proof*⟩

theorem *provenance-overapproximates-dependencies*:
  assumes *provenance.lderives* $(indexed\ 0\ \Gamma 0)\ (S,\ A)$
  shows $\forall\, i.\ i \in S \longrightarrow i < length\ \Gamma 0$
⟨*proof*⟩

lemma *prov-safe-implies-lderives*:
  assumes $\Gamma \vdash P\ x$
  shows *provenance.lderives* $\Gamma\ x$
  ⟨*proof*⟩

lemma *provenance-drop-unused-general*:
  assumes $\Gamma \vdash P\ x$ **and** $nonempty\text{-}prov\text{-}ctx\ \Gamma$
  shows $filter\text{-}labeled\text{-}by\ (fst\ x)\ \Gamma \vdash snd\ x$
  ⟨*proof*⟩

theorem *provenance-drop-unused*:
  assumes $indexed\ 0\ \Gamma 0 \vdash P\ (S,\ A)$
  shows $filter\text{-}indexed\text{-}by\ S\ \Gamma 0 \vdash A$
⟨*proof*⟩

lemma *fresh-nat*:
  assumes *finite* F

obtains $i :: nat$ **where** $i \notin F$
⟨*proof*⟩

lemma *fresh-nat2*:
assumes *finite* F
obtains $i\ j :: nat$ **where** $i \notin F$ **and** $j \notin F$ **and** $i \neq j$
⟨*proof*⟩

lemma *ordinary-admits-safe-provenance-labeled*:
assumes $\Delta \vdash A$
and *erase-ctx* $\Gamma = \Delta$
and *nonempty-prov-ctx* Γ
and *finite* (*ctx-prov* Γ)
shows $\exists S.\ \Gamma \vdash P\ (S,\ A) \wedge S \subseteq$ *ctx-prov* Γ
⟨*proof*⟩

theorem *ordinary-admits-safe-provenance*:
assumes $\Gamma \vdash A$
shows $\exists S.$ (*indexed 0* $\Gamma \vdash P\ (S,\ A)$)
$\wedge\ S \subseteq \{0..<$*length* $\Gamma\}$
$\wedge$ *set* (*filter-indexed-by* $S\ \Gamma$) $\subseteq$ *set* Γ
⟨*proof*⟩

corollary *ordinary-admits-tight-provenance*:
assumes $\Gamma \vdash A$
shows $\exists S.$ (*indexed 0* $\Gamma \vdash P\ (S,\ A)$)
$\wedge\ S \subseteq \{0..<$*length* $\Gamma\}$
$\wedge$ *filter-indexed-by* $S\ \Gamma \vdash A$
⟨*proof*⟩

theorem *provenance-completeness*:
shows $(\Gamma \vdash A) \longleftrightarrow$
$(\exists S.$ (*indexed 0* $\Gamma \vdash P\ (S,\ A)) \wedge$ *filter-indexed-by* $S\ \Gamma \vdash A$)
⟨*proof*⟩

end

theory *Modal-Labels-Example*
imports *Label-Algebra*
begin

This theory instantiates the framework locale as *modal-labels* and then extends the resulting propositional labelled calculus with modal *BoxI* and *BoxE* rules. Modal labels carry possible-world annotations; the propositional constructors preserve the current world, while box elimination projects to an accessible world.

datatype *modal-label* =
MWorld nat

```
  | MApp modal-label modal-label
  | MLam modal-label modal-label
  | MPair modal-label modal-label
  | MFst modal-label
  | MSnd modal-label
  | MInl modal-label
  | MInr modal-label
  | MCases modal-label modal-label modal-label modal-label modal-label
  | MAbort modal-label
  | MBoxI modal-label
  | MBoxE modal-label

fun world-of :: modal-label ⇒ nat where
  world-of (MWorld w) = w
| world-of (MApp l m) = world-of l
| world-of (MLam l m) = world-of m
| world-of (MPair l m) = world-of l
| world-of (MFst l) = world-of l
| world-of (MSnd l) = world-of l
| world-of (MInl l) = world-of l
| world-of (MInr l) = world-of l
| world-of (MCases l m n p q) = world-of l
| world-of (MAbort l) = world-of l
| world-of (MBoxI l) = world-of l
| world-of (MBoxE l) = world-of l

interpretation modal-labels: label-structure
  where app = MApp
    and lam = MLam
    and pair = MPair
    and fst-l = MFst
    and snd-l = MSnd
    and abort = MAbort
    and inl = MInl
    and inr = MInr
    and cases = MCases
  ⟨proof⟩

definition modal-erase-label :: modal-label ⇒ unit where
  modal-erase-label l = ()

definition modal-lift-label :: (modal-label, 'a) lfm list ⇒ 'a fm ⇒ modal-label
where
  modal-lift-label Γ A =
    (SOME l. modal-labels.lderives Γ (l, A))

interpretation modal-labels: label-algebra
  where app = MApp
    and lam = MLam
```

and *pair* = *MPair*
and *fst-l* = *MFst*
and *snd-l* = *MSnd*
and *abort* = *MAbort*
and *inl* = *MInl*
and *inr* = *MInr*
and *cases* = *MCases*
and *lderives* = *modal-labels.lderives*
and *erase-label* = *modal-erase-label*
and *lift-label* = *modal-lift-label*
⟨*proof*⟩

datatype *'a mfm* =
MAtom 'a
| *MBot*
| *MImp 'a mfm 'a mfm*
| *MCon 'a mfm 'a mfm*
| *MDis 'a mfm 'a mfm*
| *Box 'a mfm*

primrec *modalise-fm* :: *'a fm* ⇒ *'a mfm* **where**
modalise-fm (*Atom p*) = *MAtom p*
| *modalise-fm Bot* = *MBot*
| *modalise-fm* (*Imp A B*) = *MImp* (*modalise-fm A*) (*modalise-fm B*)
| *modalise-fm* (*Con A B*) = *MCon* (*modalise-fm A*) (*modalise-fm B*)
| *modalise-fm* (*Dis A B*) = *MDis* (*modalise-fm A*) (*modalise-fm B*)

definition *modalise-lfm* :: *nat* ⇒ (*modal-label*, *'a*) *lfm* ⇒ *modal-label* × *'a mfm*
where
modalise-lfm w x = (*MWorld w*, *modalise-fm* (*snd x*))

primrec *modal-kripke-sat* ::
(*nat* ⇒ *'a* ⇒ *bool*) ⇒ (*nat* ⇒ *nat* ⇒ *bool*) ⇒ *nat* ⇒ *'a mfm* ⇒ *bool*
where
modal-kripke-sat V R w (*MAtom p*) = *V w p*
| *modal-kripke-sat V R w MBot* = *False*
| *modal-kripke-sat V R w* (*MImp A B*) =
(*modal-kripke-sat V R w A* ⟶ *modal-kripke-sat V R w B*)
| *modal-kripke-sat V R w* (*MCon A B*) =
(*modal-kripke-sat V R w A* ∧ *modal-kripke-sat V R w B*)
| *modal-kripke-sat V R w* (*MDis A B*) =
(*modal-kripke-sat V R w A* ∨ *modal-kripke-sat V R w B*)
| *modal-kripke-sat V R w* (*Box A*) =
(∀ *v*. *R w v* ⟶ *modal-kripke-sat V R v A*)

inductive *mlderives* ::
(*modal-label* × *'a mfm*) *list* ⇒
(*nat* ⇒ *nat* ⇒ *bool*) ⇒ (*modal-label* × *'a mfm*) ⇒ *bool*
where

MAssm: $x \in set\ \Gamma \Longrightarrow mlderives\ \Gamma\ R\ x$
| *MBotE*: $mlderives\ \Gamma\ R\ (l,\ MBot) \Longrightarrow mlderives\ \Gamma\ R\ (MAbort\ l,\ A)$
| *MImpI*:
$world\text{-}of\ l = world\text{-}of\ m \Longrightarrow$
$mlderives\ ((l,\ A)\ \#\ \Gamma)\ R\ (m,\ B) \Longrightarrow$
$mlderives\ \Gamma\ R\ (MLam\ l\ m,\ MImp\ A\ B)$
| *MImpE*:
$world\text{-}of\ l = world\text{-}of\ m \Longrightarrow$
$mlderives\ \Gamma\ R\ (l,\ MImp\ A\ B) \Longrightarrow$
$mlderives\ \Gamma\ R\ (m,\ A) \Longrightarrow$
$mlderives\ \Gamma\ R\ (MApp\ l\ m,\ B)$
| *MConI*:
$world\text{-}of\ l = world\text{-}of\ m \Longrightarrow$
$mlderives\ \Gamma\ R\ (l,\ A) \Longrightarrow$
$mlderives\ \Gamma\ R\ (m,\ B) \Longrightarrow$
$mlderives\ \Gamma\ R\ (MPair\ l\ m,\ MCon\ A\ B)$
| *MConE1*: $mlderives\ \Gamma\ R\ (l,\ MCon\ A\ B) \Longrightarrow mlderives\ \Gamma\ R\ (MFst\ l,\ A)$
| *MConE2*: $mlderives\ \Gamma\ R\ (l,\ MCon\ A\ B) \Longrightarrow mlderives\ \Gamma\ R\ (MSnd\ l,\ B)$
| *MDisI1*: $mlderives\ \Gamma\ R\ (l,\ A) \Longrightarrow mlderives\ \Gamma\ R\ (MInl\ l,\ MDis\ A\ B)$
| *MDisI2*: $mlderives\ \Gamma\ R\ (l,\ B) \Longrightarrow mlderives\ \Gamma\ R\ (MInr\ l,\ MDis\ A\ B)$
| *MDisE*:
$world\text{-}of\ l = world\text{-}of\ m \Longrightarrow$
$world\text{-}of\ l = world\text{-}of\ n \Longrightarrow$
$world\text{-}of\ l = world\text{-}of\ p \Longrightarrow$
$world\text{-}of\ l = world\text{-}of\ q \Longrightarrow$
$mlderives\ \Gamma\ R\ (l,\ MDis\ A\ B) \Longrightarrow$
$mlderives\ ((m,\ A)\ \#\ \Gamma)\ R\ (n,\ C) \Longrightarrow$
$mlderives\ ((p,\ B)\ \#\ \Gamma)\ R\ (q,\ C) \Longrightarrow$
$mlderives\ \Gamma\ R\ (MCases\ l\ m\ n\ p\ q,\ C)$
| *MBoxI*:
$(\bigwedge v.\ R\ (world\text{-}of\ l)\ v \Longrightarrow$
$\exists\, m.\ world\text{-}of\ m = v \wedge mlderives\ \Gamma\ R\ (m,\ A)) \Longrightarrow$
$mlderives\ \Gamma\ R\ (MBoxI\ l,\ Box\ A)$
| *MBoxE*:
$mlderives\ \Gamma\ R\ (l,\ Box\ A) \Longrightarrow$
$R\ (world\text{-}of\ l)\ v \Longrightarrow$
$mlderives\ \Gamma\ R\ (MBoxE\ (MWorld\ v),\ A)$

theorem *propositional-modal-sound*:
assumes *modal-labels.lderives* $\Gamma\ x$
shows $\exists\, l.\ world\text{-}of\ l = w \wedge$
$mlderives\ (map\ (modalise\text{-}lfm\ w)\ \Gamma)\ R\ (l,\ modalise\text{-}fm\ (snd\ x))$
⟨*proof*⟩

theorem *modal-labels-sound*:
assumes $mlderives\ \Gamma\ R\ x$
and $\forall\, (l,\ A) \in set\ \Gamma.\ modal\text{-}kripke\text{-}sat\ V\ R\ (world\text{-}of\ l)\ A$
shows $modal\text{-}kripke\text{-}sat\ V\ R\ (world\text{-}of\ (fst\ x))\ (snd\ x)$
⟨*proof*⟩

end

theory *Modal-K-Schema*
imports *Modal-Labels-Example*
begin

The modal instance derives the normal modal K axiom schema directly in the labelled calculus. The proof is uniform in the formulae, the accessibility relation, and the chosen world; necessitation is available only for formulae that are derivable from the empty context at every world and under every accessibility relation.

theorem *modal-K-schema*:
fixes *A B* :: *'a mfm*
and *R* :: *nat* ⇒ *nat* ⇒ *bool*
and *w* :: *nat*
shows
∃ *l*. *world-of l* = *w* ∧
mlderives [] *R* (*l*, *MImp* (*Box* (*MImp A B*)) (*MImp* (*Box A*) (*Box B*)))
⟨*proof*⟩

theorem *modal-necessitation*:
fixes *A* :: *'a mfm*
and *R* :: *nat* ⇒ *nat* ⇒ *bool*
and *w* :: *nat*
assumes *provable-everywhere*:
⋀*v R'*. ∃ *l*. *world-of l* = *v* ∧ *mlderives* [] *R'* (*l*, *A*)
shows
∃ *l*. *world-of l* = *w* ∧ *mlderives* [] *R* (*l*, *Box A*)
⟨*proof*⟩

corollary *modal-K-kripke-valid*:
fixes *A B* :: *'a mfm*
shows ∀ *V R w*. *modal-kripke-sat V R w*
(*MImp* (*Box* (*MImp A B*)) (*MImp* (*Box A*) (*Box B*)))
⟨*proof*⟩

corollary *modal-K-example*:
shows ∃ *l*. *world-of l* = *0* ∧ *mlderives* [] *R*
(*l*, *MImp* (*Box* (*MImp* (*MAtom* (*0*::*nat*)) (*MAtom 1*)))
(*MImp* (*Box* (*MAtom 0*)) (*Box* (*MAtom 1*))))
⟨*proof*⟩

end

theory *Examples*
imports *Provenance-Labels Modal-Labels-Example*

begin

This theory collects small derivations that exercise the labelled framework: provenance-labelled implicational combinators, labelled modus ponens, and a possible-world derivation of the modal K axiom.

lemma *K-axiom-labelled*:
provenance.lderives [] ({}, *Imp A* (*Imp B A*))
⟨*proof*⟩

lemma *S-axiom-labelled*:
provenance.lderives []
({}, *Imp* (*Imp A* (*Imp B C*)) (*Imp* (*Imp A B*) (*Imp A C*)))
⟨*proof*⟩

lemma *modus-ponens-labelled*:
provenance.lderives [({*0*}, *Imp A B*), ({*1*}, *A*)] ({*0*, *1*}, *B*)
⟨*proof*⟩

lemma *modal-K-axiom*:
mlderives [] *R*
(*MLam* (*MWorld w*) (*MLam* (*MWorld w*) (*MBoxI* (*MWorld w*))),
MImp (*Box* (*MImp A B*)) (*MImp* (*Box A*) (*Box B*)))
⟨*proof*⟩

end

References

[1] D. Basin, S. Matthews, and L. Viganò. Labelled propositional modal logics: theory and practice. *Journal of Logic and Computation*, 7(6):685–717, 1997.

[2] D. M. Gabbay. *Labelled Deductive Systems, Volume 1*. Oxford University Press, 1996.

[3] A. K. Simpson. *The Proof Theory and Semantics of Intuitionistic Modal Logic*. PhD thesis, University of Edinburgh, 1994.

[4] L. Viganò. *Labelled Non-Classical Logics*. Kluwer Academic Publishers, 2000.