Public Announcement Logic

Asta Halkjær From

September 13, 2023

Abstract

This work is a formalization of public announcement logic with countably many agents. It includes proofs of soundness and completeness for variants of the axiom system PA + DIST! + NEC! [1]. The completeness proofs build on the Epistemic Logic theory.

Contents

theory *PAL* **imports** *Epistemic-Logic.Epistemic-Logic* **begin**

1 Syntax

datatype *'i pfm*
= *FF* (‹$\bot_!$›)
| *Pro'* *id* (‹*Pro*$_!$›)
| *Dis* ‹*'i pfm*› ‹*'i pfm*› (**infixr** ‹$\vee_!$› *60*)
| *Con* ‹*'i pfm*› ‹*'i pfm*› (**infixr** ‹$\wedge_!$› *65*)
| *Imp* ‹*'i pfm*› ‹*'i pfm*› (**infixr** ‹$\longrightarrow_!$› *55*)
| *K'* *'i* ‹*'i pfm*› (‹*K*$_!$›)
| *Ann* ‹*'i pfm*› ‹*'i pfm*› (‹[-]$_!$ -› [*80*, *80*] *80*)

abbreviation *PIff* :: ‹*'i pfm* ⇒ *'i pfm* ⇒ *'i pfm*› (**infixr** ‹$\longleftrightarrow_!$› *55*) **where**
‹$p \longleftrightarrow_! q \equiv (p \longrightarrow_! q) \wedge_! (q \longrightarrow_! p)$›

abbreviation *PNeg* (‹$\neg_!$ -› [*70*] *70*) **where**
‹$\neg_! p \equiv p \longrightarrow_! \bot_!$›

abbreviation *PL* (‹*L*$_!$›) **where**
‹$L_! \; i \; p \equiv (\neg_! \; (K_! \; i \; (\neg_! \; p)))$›

primrec *anns* :: ‹*'i pfm* ⇒ *'i pfm set*› **where**
‹*anns* $\bot_!$ = {}›
| ‹*anns* (*Pro*$_!$ -) = {}›
| ‹*anns* ($p \vee_! q$) = (*anns* p ∪ *anns* q)›
| ‹*anns* ($p \wedge_! q$) = (*anns* p ∪ *anns* q)›
| ‹*anns* ($p \longrightarrow_! q$) = (*anns* p ∪ *anns* q)›
| ‹*anns* ($K_! \; i \; p$) = *anns* p›
| ‹*anns* ($[r]_! \; p$) = {r} ∪ *anns* r ∪ *anns* p›

2 Semantics

fun
psemantics :: ‹(*'i*, *'w*) *kripke* ⇒ *'w* ⇒ *'i pfm* ⇒ *bool*› (‹-, - $\models_!$ -› [*50*, *50*, *50*] *50*) **and**
restrict :: ‹(*'i*, *'w*) *kripke* ⇒ *'i pfm* ⇒ (*'i*, *'w*) *kripke*› (‹-[-!]› [*50*, *50*] *50*) **where**
‹$M, w \models_! \bot_! \longleftrightarrow$ *False*›
| ‹$M, w \models_!$ *Pro*$_!$ $x \longleftrightarrow \pi \; M \; w \; x$›
| ‹$M, w \models_! p \vee_! q \longleftrightarrow M, w \models_! p \vee M, w \models_! q$›
| ‹$M, w \models_! p \wedge_! q \longleftrightarrow M, w \models_! p \wedge M, w \models_! q$›
| ‹$M, w \models_! p \longrightarrow_! q \longleftrightarrow M, w \models_! p \longrightarrow M, w \models_! q$›
| ‹$M, w \models_! K_! \; i \; p \longleftrightarrow (\forall v \in \mathcal{W} \; M \cap \mathcal{K} \; M \; i \; w. \; M, v \models_! p)$›
| ‹$M, w \models_! [r]_! \; p \longleftrightarrow M, w \models_! r \longrightarrow M[r!], w \models_! p$›
| ‹$M[r!] = M$ (|$\mathcal{W}$:= {w. $w \in \mathcal{W} \; M \wedge M, w \models_! r$}|)›

abbreviation *validPStar* :: ‹(('*i*, '*w*) *kripke* ⇒ *bool*) ⇒ '*i pfm set* ⇒ '*i pfm* ⇒ *bool*›
(‹-; - ⊫!⋆ -› [*50*, *50*, *50*] *50*) **where**
‹*P*; *G* ⊫!⋆ *p* ≡ ∀ *M*. *P M* ⟶ (∀ *w* ∈ 𝒲 *M*. (∀ *q* ∈ *G*. *M*, *w* ⊨! *q*) ⟶ *M*, *w* ⊨! *p*)›

primrec *static* :: ‹'*i pfm* ⇒ *bool*› **where**
‹*static* ⊥! = *True*›
| ‹*static* (*Pro*! -) = *True*›
| ‹*static* (*p* ∨! *q*) = (*static p* ∧ *static q*)›
| ‹*static* (*p* ∧! *q*) = (*static p* ∧ *static q*)›
| ‹*static* (*p* ⟶! *q*) = (*static p* ∧ *static q*)›
| ‹*static* (*K*! *i p*) = *static p*›
| ‹*static* ([*r*]! *p*) = *False*›

primrec *lower* :: ‹'*i pfm* ⇒ '*i fm*› **where**
‹*lower* ⊥! = ⊥›
| ‹*lower* (*Pro*! *x*) = *Pro x*›
| ‹*lower* (*p* ∨! *q*) = (*lower p* ∨ *lower q*)›
| ‹*lower* (*p* ∧! *q*) = (*lower p* ∧ *lower q*)›
| ‹*lower* (*p* ⟶! *q*) = (*lower p* ⟶ *lower q*)›
| ‹*lower* (*K*! *i p*) = *K i* (*lower p*)›
| ‹*lower* ([*r*]! *p*) = *undefined*›

primrec *lift* :: ‹'*i fm* ⇒ '*i pfm*› **where**
‹*lift* ⊥ = ⊥!›
| ‹*lift* (*Pro x*) = *Pro*! *x*›
| ‹*lift* (*p* ∨ *q*) = (*lift p* ∨! *lift q*)›
| ‹*lift* (*p* ∧ *q*) = (*lift p* ∧! *lift q*)›
| ‹*lift* (*p* ⟶ *q*) = (*lift p* ⟶! *lift q*)›
| ‹*lift* (*K i p*) = *K*! *i* (*lift p*)›

lemma *lower-semantics*:
assumes ‹*static p*›
shows ‹(*M*, *w* ⊨ *lower p*) ⟷ (*M*, *w* ⊨! *p*)›
using *assms* **by** (*induct p arbitrary*: *w*) *simp-all*

lemma *lift-semantics*: ‹(*M*, *w* ⊨ *p*) ⟷ (*M*, *w* ⊨! *lift p*)›
by (*induct p arbitrary*: *w*) *simp-all*

lemma *lower-lift*: ‹*lower* (*lift p*) = *p*›
by (*induct p*) *simp-all*

lemma *lift-lower*: ‹*static p* ⟹ *lift* (*lower p*) = *p*›
by (*induct p*) *simp-all*

3 Soundness of Reduction

primrec *reduce'* :: ‹'*i pfm* ⇒ '*i pfm* ⇒ '*i pfm*› **where**

```
  ‹reduce′ r ⊥! = (r ⟶! ⊥!)›
| ‹reduce′ r (Pro! x) = (r ⟶! Pro! x)›
| ‹reduce′ r (p ∨! q) = (reduce′ r p ∨! reduce′ r q)›
| ‹reduce′ r (p ∧! q) = (reduce′ r p ∧! reduce′ r q)›
| ‹reduce′ r (p ⟶! q) = (reduce′ r p ⟶! reduce′ r q)›
| ‹reduce′ r (K! i p) = (r ⟶! K! i (reduce′ r p))›
| ‹reduce′ r ([p]! q) = undefined›

primrec reduce :: ‹'i pfm ⇒ 'i pfm› where
  ‹reduce ⊥! = ⊥!›
| ‹reduce (Pro! x) = Pro! x›
| ‹reduce (p ∨! q) = (reduce p ∨! reduce q)›
| ‹reduce (p ∧! q) = (reduce p ∧! reduce q)›
| ‹reduce (p ⟶! q) = (reduce p ⟶! reduce q)›
| ‹reduce (K! i p) = K! i (reduce p)›
| ‹reduce ([r]! p) = reduce′ (reduce r) (reduce p)›

lemma static-reduce′: ‹static p ⟹ static r ⟹ static (reduce′ r p)›
  by (induct p) simp-all

lemma static-reduce: ‹static (reduce p)›
  by (induct p) (simp-all add: static-reduce′)

lemma reduce′-semantics:
  assumes ‹static q›
  shows ‹(M, w ⊨! [p]! q) = (M, w ⊨! reduce′ p q)›
  using assms by (induct q arbitrary: w) auto

lemma reduce-semantics: ‹M, w ⊨! p ⟷ M, w ⊨! reduce p›
proof (induct p arbitrary: M w)
  case (Ann p q)
  then show ?case
    using reduce′-semantics static-reduce by fastforce
qed simp-all
```

4 Chains of Implications

```
primrec implyP :: ‹'i pfm list ⇒ 'i pfm ⇒ 'i pfm› (infixr ‹⇝!› 56) where
  ‹([] ⇝! q) = q›
| ‹(p # ps ⇝! q) = (p ⟶! ps ⇝! q)›

lemma lift-implyP: ‹lift (ps ⇝ q) = (map lift ps ⇝! lift q)›
  by (induct ps) auto

lemma reduce-implyP: ‹reduce (ps ⇝! q) = (map reduce ps ⇝! reduce q)›
  by (induct ps) auto
```

5 Proof System

primrec *peval* :: ‹(*id* ⇒ *bool*) ⇒ ('*i pfm* ⇒ *bool*) ⇒ '*i pfm* ⇒ *bool*› **where**
  ‹*peval* - - ⊥! = *False*›
| ‹*peval g* - (*Pro*! *x*) = *g x*›
| ‹*peval g h* (*p* ∨! *q*) = (*peval g h p* ∨ *peval g h q*)›
| ‹*peval g h* (*p* ∧! *q*) = (*peval g h p* ∧ *peval g h q*)›
| ‹*peval g h* (*p* ⟶! *q*) = (*peval g h p* ⟶ *peval g h q*)›
| ‹*peval* - *h* (*K*! *i p*) = *h* (*K*! *i p*)›
| ‹*peval* - *h* ([*r*]! *p*) = *h* ([*r*]! *p*)›

abbreviation ‹*ptautology p* ≡ ∀ *g h*. *peval g h p*›

inductive *PAK* :: ‹('*i pfm* ⇒ *bool*) ⇒ ('*i pfm* ⇒ *bool*) ⇒ '*i pfm* ⇒ *bool*›
  (‹-; - ⊢! -› [*50*, *50*, *50*] *50*)
  for *A B* :: ‹'*i pfm* ⇒ *bool*› **where**
    PA1: ‹*ptautology p* ⟹ *A*; *B* ⊢! *p*›
| *PA2*: ‹*A*; *B* ⊢! *K*! *i p* ∧! *K*! *i* (*p* ⟶! *q*) ⟶! *K*! *i q*›
| *PAx*: ‹*A p* ⟹ *A*; *B* ⊢! *p*›
| *PR1*: ‹*A*; *B* ⊢! *p* ⟹ *A*; *B* ⊢! *p* ⟶! *q* ⟹ *A*; *B* ⊢! *q*›
| *PR2*: ‹*A*; *B* ⊢! *p* ⟹ *A*; *B* ⊢! *K*! *i p*›
| *PAnn*: ‹*A*; *B* ⊢! *p* ⟹ *B r* ⟹ *A*; *B* ⊢! [*r*]! *p*›
| *PFF*: ‹*A*; *B* ⊢! [*r*]! ⊥! ⟷! (*r* ⟶! ⊥!)›
| *PPro*: ‹*A*; *B* ⊢! [*r*]! *Pro*! *x* ⟷! (*r* ⟶! *Pro*! *x*)›
| *PDis*: ‹*A*; *B* ⊢! [*r*]! (*p* ∨! *q*) ⟷! [*r*]! *p* ∨! [*r*]! *q*›
| *PCon*: ‹*A*; *B* ⊢! [*r*]! (*p* ∧! *q*) ⟷! [*r*]! *p* ∧! [*r*]! *q*›
| *PImp*: ‹*A*; *B* ⊢! [*r*]! (*p* ⟶! *q*) ⟷! ([*r*]! *p* ⟶! [*r*]! *q*)›
| *PK*: ‹*A*; *B* ⊢! [*r*]! *K*! *i p* ⟷! (*r* ⟶! *K*! *i* ([*r*]! *p*))›

abbreviation *PAK-assms* (‹-; -; - ⊢! -› [*50*, *50*, *50*, *50*] *50*) **where**
  ‹*A*; *B*; *G* ⊢! *p* ≡ ∃ *qs*. *set qs* ⊆ *G* ∧ (*A*; *B* ⊢! *qs* ⇝! *p*)›

lemma *eval-peval*: ‹*eval h* (*g o lift*) *p* = *peval h g* (*lift p*)›
  by (*induct p*) *simp-all*

lemma *tautology-ptautology*: ‹*tautology p* ⟹ *ptautology* (*lift p*)›
  using *eval-peval* **by** *blast*

theorem *AK-PAK*: ‹*A o lift* ⊢ *p* ⟹ *A*; *B* ⊢! *lift p*›
  by (*induct p rule*: *AK.induct*) (*auto intro*: *PAK.intros*(*1*−*5*) *simp*: *tautology-ptautology*)

abbreviation *validP*
  :: ‹(('*i*, '*i fm set*) *kripke* ⇒ *bool*) ⇒ '*i pfm set* ⇒ '*i pfm* ⇒ *bool*›
  (‹-; - ⊨! -› [*50*, *50*, *50*] *50*)
  where ‹*P*; *G* ⊨! *p* ≡ *P*; *G* ⊨!⋆ *p*›

lemma *set-map-inv*:
  assumes ‹*set xs* ⊆ *f* ‘ *X*›
  shows ‹∃ *ys*. *set ys* ⊆ *X* ∧ *map f ys* = *xs*›

using *assms*
proof (*induct xs*)
case (*Cons x xs*)
then obtain *ys* **where** ‹*set ys* ⊆ *X*› ‹*map f ys* = *xs*›
by *auto*
moreover obtain *y* **where** ‹*y* ∈ *X*› ‹*f y* = *x*›
using *Cons.prems* **by** *auto*
ultimately have ‹*set* (*y* # *ys*) ⊆ *X*› ‹*map f* (*y* # *ys*) = *x* # *xs*›
by *simp-all*
then show *?case*
by *meson*
qed *simp*

lemma *strong-static-completeness′*:
assumes ‹*static p*› **and** ‹∀ *q* ∈ *G*. *static q*› **and** ‹*P*; *G* ∥=! *p*›
and ‹*P*; *lower* ‘ *G* ∥=⋆ *lower p* ⟹ *A o lift*; *lower* ‘ *G* ⊢ *lower p*›
shows ‹*A*; *B*; *G* ⊢! *p*›
proof −
have ‹*P*; *lower* ‘ *G* ∥=⋆ *lower p*›
using *assms* **by** (*simp add*: *lower-semantics*)
then have ‹*A o lift*; *lower* ‘ *G* ⊢ *lower p*›
using *assms*(*4*) **by** *blast*
then obtain *qs* **where** ‹*set qs* ⊆ *G*› **and** ‹*A o lift* ⊢ *map lower qs* ↝ *lower p*›
using *set-map-inv* **by** *blast*
then have ‹*A*; *B* ⊢! *lift* (*map lower qs* ↝ *lower p*)›
using *AK-PAK* **by** *fast*
then have ‹*A*; *B* ⊢! *map lift* (*map lower qs*) ↝! *lift* (*lower p*)›
using *lift-implyP* **by** *metis*
then have ‹*A*; *B* ⊢! *map* (*lift o lower*) *qs* ↝! *lift* (*lower p*)›
by *simp*
then show *?thesis*
using *assms*(*1−2*) ‹*set qs* ⊆ *G*› *lift-lower*
by (*metis* (*mono-tags*, *lifting*) *comp-apply map-idI subset-eq*)
qed

theorem *strong-static-completeness*:
assumes ‹*static p*› **and** ‹∀ *q* ∈ *G*. *static q*› **and** ‹*P*; *G* ∥=! *p*›
and ‹⋀*G p*. *P*; *G* ∥= *p* ⟹ *A o lift*; *G* ⊢ *p*›
shows ‹*A*; *B*; *G* ⊢! *p*›
using *strong-static-completeness′ assms* **.**

corollary *static-completeness′*:
assumes ‹*static p*› **and** ‹*P*; {} ∥=!⋆ *p*›
and ‹*P*; {} ∥= *lower p* ⟹ *A o lift* ⊢ *lower p*›
shows ‹*A*; *B* ⊢! *p*›
using *assms strong-static-completeness′*[**where** *G*=‹{}› **and** *p*=*p*] **by** *simp*

corollary *static-completeness*:
assumes ‹*static p*› **and** ‹*P*; {} ∥=!⋆ *p*› **and** ‹⋀*p*. *P*; {} ∥= *p* ⟹ *A o lift* ⊢ *p*›

shows ‹A; $B \vdash_! p$›
using *static-completeness'* *assms* **.**

corollary
assumes ‹*static* p› ‹(λ-. *True*); {} $\Vdash_!$ p›
shows ‹A; $B \vdash_! p$›
using *assms* *static-completeness*[**where** P=‹λ-. *True*› **and** p=p] $completeness_A$
by *blast*

6 Soundness

lemma *peval-semantics*:
‹*peval* (*val* w) (λq. ($\mathcal{W}$ = W, $\mathcal{K}$ = r, π = *val*), $w \models_!$ q) p = (($\mathcal{W}$ = W, $\mathcal{K}$ = r, π = *val*), $w \models_!$ p)›
by (*induct* p) *simp-all*

lemma *ptautology*:
assumes ‹*ptautology* p›
shows ‹M, $w \models_!$ p›
proof −
from *assms* **have** ‹*peval* (g w) (λq. ($\mathcal{W}$ = W, $\mathcal{K}$ = r, π = g), $w \models_!$ q) p› **for** W g r
by *simp*
then have ‹($\mathcal{W}$ = W, $\mathcal{K}$ = r, π = g), $w \models_!$ p› **for** W g r
using *peval-semantics* **by** *fast*
then show ‹M, $w \models_!$ p›
by (*metis kripke.cases*)
qed

theorem $soundness_P$:
assumes
‹$\bigwedge M$ p w. A $p \Longrightarrow P$ $M \Longrightarrow w \in \mathcal{W}$ $M \Longrightarrow M$, $w \models_!$ p›
‹$\bigwedge M$ r. P $M \Longrightarrow B$ $r \Longrightarrow P$ (M[r!])›
shows ‹A; $B \vdash_!$ $p \Longrightarrow P$ $M \Longrightarrow w \in \mathcal{W}$ $M \Longrightarrow M$, $w \models_!$ p›
proof (*induct* p *arbitrary*: M w *rule*: *PAK.induct*)
case (*PAnn* p r)
then show *?case*
using *assms* **by** *simp*
qed (*simp-all add*: *assms ptautology*)

corollary ‹(λ-. *False*); $B \vdash_!$ $p \Longrightarrow w \in \mathcal{W}$ $M \Longrightarrow M$, $w \models_!$ p›
using $soundness_P$[**where** P=‹λ-. *True*›] **by** *metis*

7 Strong Soundness

lemma *ptautology-imply-superset*:
assumes ‹*set ps* $\subseteq$ *set qs*›
shows ‹*ptautology* (*ps* $\rightsquigarrow_!$ $r \longrightarrow_!$ *qs* $\rightsquigarrow_!$ r)›

proof (*rule ccontr*)
assume ‹¬ *?thesis*›
then obtain *g h* **where** ‹¬ *peval g h* (*ps* ⇝! *r* ⟶! *qs* ⇝! *r*)›
by *blast*
then have ‹*peval g h* (*ps* ⇝! *r*)› ‹¬ *peval g h* (*qs* ⇝! *r*)›
by *simp-all*
then consider (*np*) ‹∃ *p* ∈ *set ps*. ¬ *peval g h p*› | (*r*) ‹∀ *p* ∈ *set ps*. *peval g h p*›
‹*peval g h r*›
by (*induct ps*) *auto*
then show *False*
proof *cases*
case *np*
then have ‹∃ *p* ∈ *set qs*. ¬ *peval g h p*›
using ‹*set ps* ⊆ *set qs*› **by** *blast*
then have ‹*peval g h* (*qs* ⇝! *r*)›
by (*induct qs*) *simp-all*
then show *?thesis*
using ‹¬ *peval g h* (*qs* ⇝! *r*)› **by** *blast*
next
case *r*
then have ‹*peval g h* (*qs* ⇝! *r*)›
by (*induct qs*) *simp-all*
then show *?thesis*
using ‹¬ *peval g h* (*qs* ⇝! *r*)› **by** *blast*
qed
qed

lemma *PK-imply-weaken*:
assumes ‹*A*; *B* ⊢! *ps* ⇝! *q*› ‹*set ps* ⊆ *set ps′*›
shows ‹*A*; *B* ⊢! *ps′* ⇝! *q*›
proof −
have ‹*ptautology* (*ps* ⇝! *q* ⟶! *ps′* ⇝! *q*)›
using ‹*set ps* ⊆ *set ps′*› *ptautology-imply-superset* **by** *blast*
then have ‹*A*; *B* ⊢! *ps* ⇝! *q* ⟶! *ps′* ⇝! *q*›
using *PA1* **by** *blast*
then show *?thesis*
using ‹*A*; *B* ⊢! *ps* ⇝! *q*› *PR1* **by** *blast*
qed

lemma *implyP-append*: ‹(*ps* @ *ps′* ⇝! *q*) = (*ps* ⇝! *ps′* ⇝! *q*)›
by (*induct ps*) *simp-all*

lemma *PK-ImpI*:
assumes ‹*A*; *B* ⊢! *p* # *G* ⇝! *q*›
shows ‹*A*; *B* ⊢! *G* ⇝! (*p* ⟶! *q*)›
proof −
have ‹*set* (*p* # *G*) ⊆ *set* (*G* @ [*p*])›
by *simp*
then have ‹*A*; *B* ⊢! *G* @ [*p*] ⇝! *q*›

using *assms PK-imply-weaken* **by** *blast*
then have ‹A; $B \vdash_!$ $G \rightsquigarrow_!$ $[p] \rightsquigarrow_!$ q›
using *implyP-append* **by** *metis*
then show *?thesis*
by *simp*
qed

corollary *soundness-imply*$_P$:
assumes
‹$\bigwedge M\ p\ w.\ A\ p \Longrightarrow P\ M \Longrightarrow w \in \mathcal{W}\ M \Longrightarrow M,\ w \models_!\ p$›
‹$\bigwedge M\ r.\ P\ M \Longrightarrow B\ r \Longrightarrow P\ (M[r!])$›
shows ‹A; $B \vdash_!$ $qs \rightsquigarrow_!$ $p \Longrightarrow P\ M \Longrightarrow w \in \mathcal{W}\ M \Longrightarrow \forall\, q \in set\ qs.\ M,\ w \models_!\ q$ $\Longrightarrow M,\ w \models_!\ p$›
proof (*induct qs arbitrary*: p)
case *Nil*
then show *?case*
using *soundness*$_P$[*of A P B p M w*] *assms* **by** *simp*
next
case (*Cons q qs*)
then show *?case*
using *PK-ImpI* **by** *fastforce*
qed

theorem *strong-soundness*$_P$:
assumes
‹$\bigwedge M\ w\ p.\ A\ p \Longrightarrow P\ M \Longrightarrow w \in \mathcal{W}\ M \Longrightarrow M,\ w \models_!\ p$›
‹$\bigwedge M\ r.\ P\ M \Longrightarrow B\ r \Longrightarrow P\ (M[r!])$›
shows ‹A; B; $G \vdash_!$ $p \Longrightarrow P$; $G \Vdash_!\star$ p›
proof *safe*
fix *qs w* **and** M :: ‹('a, 'b) *kripke*›
assume ‹A; $B \vdash_!$ $qs \rightsquigarrow_!$ p›
moreover assume ‹*set qs* $\subseteq$ G› ‹$\forall\, q \in G.\ M,\ w \models_!\ q$›
then have ‹$\forall\, q \in set\ qs.\ M,\ w \models_!\ q$›
using ‹*set qs* $\subseteq$ G› **by** *blast*
moreover assume ‹$P\ M$› ‹$w \in \mathcal{W}\ M$›
ultimately show ‹$M,\ w \models_!\ p$›
using *soundness-imply*$_P$[*of A P B qs p*] *assms* **by** *blast*
qed

8 Completeness

lemma *ConE*:
assumes ‹A; $B \vdash_!$ $p \wedge_!$ q›
shows ‹A; $B \vdash_!$ p› ‹A; $B \vdash_!$ q›
using *assms* **by** (*metis PA1 PR1 peval.simps*(4−5))+

lemma *Iff-Dis*:
assumes ‹A; $B \vdash_!$ $p \longleftrightarrow_!$ p'› ‹A; $B \vdash_!$ $q \longleftrightarrow_!$ q'›
shows ‹A; $B \vdash_!$ $((p \vee_!\ q) \longleftrightarrow_!\ (p' \vee_!\ q'))$›

proof −
have ‹A; $B \vdash_!\ (p \longleftrightarrow_!\ p') \longrightarrow_!\ (q \longleftrightarrow_!\ q') \longrightarrow_!\ ((p \vee_!\ q) \longleftrightarrow_!\ (p' \vee_!\ q'))$›
by (*simp add*: *PA1*)
then show *?thesis*
using *assms PR1* **by** *blast*
qed

lemma *Iff-Con*:
assumes ‹A; $B \vdash_!\ p \longleftrightarrow_!\ p'$› ‹$A$; $B \vdash_!\ q \longleftrightarrow_!\ q'$›
shows ‹A; $B \vdash_!\ (p \wedge_!\ q) \longleftrightarrow_!\ (p' \wedge_!\ q')$›
proof −
have ‹A; $B \vdash_!\ (p \longleftrightarrow_!\ p') \longrightarrow_!\ (q \longleftrightarrow_!\ q') \longrightarrow_!\ ((p \wedge_!\ q) \longleftrightarrow_!\ (p' \wedge_!\ q'))$›
by (*simp add*: *PA1*)
then show *?thesis*
using *assms PR1* **by** *blast*
qed

lemma *Iff-Imp*:
assumes ‹A; $B \vdash_!\ p \longleftrightarrow_!\ p'$› ‹$A$; $B \vdash_!\ q \longleftrightarrow_!\ q'$›
shows ‹A; $B \vdash_!\ ((p \longrightarrow_!\ q) \longleftrightarrow_!\ (p' \longrightarrow_!\ q'))$›
proof −
have ‹A; $B \vdash_!\ (p \longleftrightarrow_!\ p') \longrightarrow_!\ (q \longleftrightarrow_!\ q') \longrightarrow_!\ ((p \longrightarrow_!\ q) \longleftrightarrow_!\ (p' \longrightarrow_!\ q'))$›
by (*simp add*: *PA1*)
then show *?thesis*
using *assms PR1* **by** *blast*
qed

lemma *Iff-sym*: ‹$(A;\ B \vdash_!\ p \longleftrightarrow_!\ q) = (A;\ B \vdash_!\ q \longleftrightarrow_!\ p)$›
proof −
have ‹A; $B \vdash_!\ (p \longleftrightarrow_!\ q) \longleftrightarrow_!\ (q \longleftrightarrow_!\ p)$›
by (*simp add*: *PA1*)
then show *?thesis*
using *PR1 ConE* **by** *blast*
qed

lemma *Iff-Iff*:
assumes ‹A; $B \vdash_!\ p \longleftrightarrow_!\ p'$› ‹$A$; $B \vdash_!\ p \longleftrightarrow_!\ q$›
shows ‹A; $B \vdash_!\ p' \longleftrightarrow_!\ q$›
proof −
have ‹*ptautology* $((p \longleftrightarrow_!\ p') \longrightarrow_!\ (p \longleftrightarrow_!\ q) \longrightarrow_!\ (p' \longleftrightarrow_!\ q))$›
by (*metis peval.simps*(*4*−*5*))
with *PA1* **have** ‹A; $B \vdash_!\ (p \longleftrightarrow_!\ p') \longrightarrow_!\ (p \longleftrightarrow_!\ q) \longrightarrow_!\ (p' \longleftrightarrow_!\ q)$› **.**
then show *?thesis*
using *assms PR1* **by** *blast*
qed

lemma *K'-A2'*: ‹A; $B \vdash_!\ K_!\ i\ (p \longrightarrow_!\ q) \longrightarrow_!\ K_!\ i\ p \longrightarrow_!\ K_!\ i\ q$›
proof −

have ‹A; $B \vdash_!\ K_!\ i\ p \wedge_!\ K_!\ i\ (p \longrightarrow_!\ q) \longrightarrow_!\ K_!\ i\ q$›
using *PA2* **by** *fast*
moreover have ‹A; $B \vdash_!\ (P \wedge_!\ Q \longrightarrow_!\ R) \longrightarrow_!\ (Q \longrightarrow_!\ P \longrightarrow_!\ R)$› **for** $P\ Q\ R$
by (*simp add*: *PA1*)
ultimately show *?thesis*
using *PR1* **by** *fast*
qed

lemma *K′-map*:
assumes ‹A; $B \vdash_!\ p \longrightarrow_!\ q$›
shows ‹A; $B \vdash_!\ K_!\ i\ p \longrightarrow_!\ K_!\ i\ q$›
proof −
note ‹A; $B \vdash_!\ p \longrightarrow_!\ q$›
then have ‹A; $B \vdash_!\ K_!\ i\ (p \longrightarrow_!\ q)$›
using *PR2* **by** *fast*
moreover have ‹A; $B \vdash_!\ K_!\ i\ (p \longrightarrow_!\ q) \longrightarrow_!\ K_!\ i\ p \longrightarrow_!\ K_!\ i\ q$›
using *K′-A2′* **by** *fast*
ultimately show *?thesis*
using *PR1* **by** *fast*
qed

lemma *ConI*:
assumes ‹A; $B \vdash_!\ p$› ‹A; $B \vdash_!\ q$›
shows ‹A; $B \vdash_!\ p \wedge_!\ q$›
proof −
have ‹A; $B \vdash_!\ p \longrightarrow_!\ q \longrightarrow_!\ p \wedge_!\ q$›
by (*simp add*: *PA1*)
then show *?thesis*
using *assms PR1* **by** *blast*
qed

lemma *Iff-wk*:
assumes ‹A; $B \vdash_!\ p \longleftrightarrow_!\ q$›
shows ‹A; $B \vdash_!\ (r \longrightarrow_!\ p) \longleftrightarrow_!\ (r \longrightarrow_!\ q)$›
proof −
have ‹A; $B \vdash_!\ (p \longleftrightarrow_!\ q) \longrightarrow_!\ ((r \longrightarrow_!\ p) \longleftrightarrow_!\ (r \longrightarrow_!\ q))$›
by (*simp add*: *PA1*)
then show *?thesis*
using *assms PR1* **by** *blast*
qed

lemma *Iff-reduce′*:
assumes ‹*static* p›
shows ‹A; $B \vdash_!\ [r]_!\ p \longleftrightarrow_!$ *reduce′* $r\ p$›
using *assms*
proof (*induct p rule*: *pfm.induct*)
case *FF*
then show *?case*

by (*simp add*: *PFF*)
next
case (*Pro′ x*)
then show *?case*
by (*simp add*: *PPro*)
next
case (*Dis p q*)
then have ‹*A*; *B* ⊢! [*r*]! *p* ∨! [*r*]! *q* ⟷! *reduce′ r* (*p* ∨! *q*)›
using *Iff-Dis* **by** *fastforce*
moreover have ‹*A*; *B* ⊢! ([*r*]! *p* ∨! [*r*]! *q*) ⟷! ([*r*]! (*p* ∨! *q*))›
using *PDis Iff-sym* **by** *fastforce*
ultimately show *?case*
using *PA1 PR1 Iff-Iff* **by** *blast*
next
case (*Con p q*)
then have ‹*A*; *B* ⊢! [*r*]! *p* ∧! [*r*]! *q* ⟷! *reduce′ r* (*p* ∧! *q*)›
using *Iff-Con* **by** *fastforce*
moreover have ‹*A*; *B* ⊢! ([*r*]! *p* ∧! [*r*]! *q*) ⟷! ([*r*]! (*p* ∧! *q*))›
using *PCon Iff-sym* **by** *fastforce*
ultimately show *?case*
using *PA1 PR1 Iff-Iff* **by** *blast*
next
case (*Imp p q*)
then have ‹*A*; *B* ⊢! ([*r*]! *p* ⟶! [*r*]! *q*) ⟷! *reduce′ r* (*p* ⟶! *q*)›
using *Iff-Imp* **by** *fastforce*
moreover have ‹*A*; *B* ⊢! ([*r*]! *p* ⟶! [*r*]! *q*) ⟷! ([*r*]! (*p* ⟶! *q*))›
using *PImp Iff-sym* **by** *fastforce*
ultimately show *?case*
using *PA1 PR1 Iff-Iff* **by** *blast*
next
case (*K′ i p*)
then have ‹*A*; *B* ⊢! [*r*]! *p* ⟷! *reduce′ r p*›
by *simp*
then have ‹*A*; *B* ⊢! *K*! *i* ([*r*]! *p*) ⟷! *K*! *i* (*reduce′ r p*)›
using *K′-map ConE ConI* **by** *metis*
moreover have ‹*A*; *B* ⊢! [*r*]! *K*! *i p* ⟷! *r* ⟶! *K*! *i* ([*r*]! *p*)›
using *PK* **.**
ultimately have ‹*A*; *B* ⊢! [*r*]! *K*! *i p* ⟷! *r* ⟶! *K*! *i* (*reduce′ r p*)›
by (*meson Iff-Iff Iff-sym Iff-wk*)
then show *?case*
by *simp*
next
case (*Ann r p*)
then show *?case*
by *simp*
qed

lemma *Iff-Ann1*:
assumes *r*: ‹*A*; *B* ⊢! *r* ⟷! *r′*› **and** ‹*static p*›

shows ‹A; $B \vdash_!$ $[r]_!$ $p \longleftrightarrow_!$ $[r']_!$ p›
using *assms*(2−)
proof (*induct p*)
case *FF*
have ‹A; $B \vdash_!$ $(r \longleftrightarrow_!$ $r') \longrightarrow_!$ $((r \longrightarrow_!$ $\bot_!) \longleftrightarrow_!$ $(r' \longrightarrow_!$ $\bot_!))$›
by (*auto intro*: *PA1*)
then have ‹A; $B \vdash_!$ $(r \longrightarrow_!$ $\bot_!) \longleftrightarrow_!$ $(r' \longrightarrow_!$ $\bot_!)$›
using *r PR1* **by** *blast*
then show *?case*
by (*meson PFF Iff-Iff Iff-sym*)
next
case (*Pro′ x*)
have ‹A; $B \vdash_!$ $(r \longleftrightarrow_!$ $r') \longrightarrow_!$ $((r \longrightarrow_!$ $Pro_!$ $x) \longleftrightarrow_!$ $(r' \longrightarrow_!$ $Pro_!$ $x))$›
by (*auto intro*: *PA1*)
then have ‹A; $B \vdash_!$ $(r \longrightarrow_!$ $Pro_!$ $x) \longleftrightarrow_!$ $(r' \longrightarrow_!$ $Pro_!$ $x)$›
using *r PR1* **by** *blast*
then show *?case*
by (*meson PPro Iff-Iff Iff-sym*)
next
case (*Dis p q*)
then have ‹A; $B \vdash_!$ $[r]_!$ $p \vee_!$ $[r]_!$ $q \longleftrightarrow_!$ $[r']_!$ $p \vee_!$ $[r']_!$ q›
by (*simp add*: *Iff-Dis*)
then show *?case*
by (*meson PDis Iff-Iff Iff-sym*)
next
case (*Con p q*)
then have ‹A; $B \vdash_!$ $[r]_!$ $p \wedge_!$ $[r]_!$ $q \longleftrightarrow_!$ $[r']_!$ $p \wedge_!$ $[r']_!$ q›
by (*simp add*: *Iff-Con*)
then show *?case*
by (*meson PCon Iff-Iff Iff-sym*)
next
case (*Imp p q*)
then have ‹A; $B \vdash_!$ $([r]_!$ $p \longrightarrow_!$ $[r]_!$ $q) \longleftrightarrow_!$ $([r']_!$ $p \longrightarrow_!$ $[r']_!$ $q)$›
by (*simp add*: *Iff-Imp*)
then show *?case*
by (*meson PImp Iff-Iff Iff-sym*)
next
case (*K′ i p*)
then have ‹A; $B \vdash_!$ $[r]_!$ $p \longleftrightarrow_!$ $[r']_!$ p›
by *simp*
then have ‹A; $B \vdash_!$ $K_!$ i $([r]_!$ $p) \longleftrightarrow_!$ $K_!$ i $([r']_!$ $p)$›
using *K′-map ConE ConI* **by** *metis*
then show *?case*
by (*meson Iff-Iff Iff-Imp Iff-sym PK r*)
next
case (*Ann s p*)
then show *?case*
by *simp*
qed

lemma *Iff-Ann2*:
assumes ‹A; $B \vdash_!\ p \longleftrightarrow_!\ p'$› **and** ‹$B\ r$›
shows ‹A; $B \vdash_!\ [r]_!\ p \longleftrightarrow_!\ [r]_!\ p'$›
using *assms PAnn ConE ConI PImp PR1* **by** *metis*

lemma *Iff-reduce*:
assumes ‹$\forall\, r \in anns\ p.\ B\ r$›
shows ‹A; $B \vdash_!\ p \longleftrightarrow_!\ reduce\ p$›
using *assms*
proof (*induct p*)
case (*Dis p q*)
then show *?case*
by (*simp add*: *Iff-Dis*)
next
case (*Con p q*)
then show *?case*
by (*simp add*: *Iff-Con*)
next
case (*Imp p q*)
then show *?case*
by (*simp add*: *Iff-Imp*)
next
case (*K' i p*)
then have
‹A; $B \vdash_!\ K_!\ i\ p \longrightarrow_!\ K_!\ i\ (reduce\ p)$›
‹A; $B \vdash_!\ K_!\ i\ (reduce\ p) \longrightarrow_!\ K_!\ i\ p$›
using *K'-map ConE* **by** *fastforce+*
then have ‹A; $B \vdash_!\ K_!\ i\ p \longleftrightarrow_!\ K_!\ i\ (reduce\ p)$›
using *ConI* **by** *blast*
then show *?case*
by *simp*
next
case (*Ann r p*)
then have ‹$B\ r$›
by *simp*
have ‹A; $B \vdash_!\ [reduce\ r]_!\ reduce\ p \longleftrightarrow_!\ reduce'\ (reduce\ r)\ (reduce\ p)$›
using *Iff-reduce' static-reduce* **by** *blast*
moreover have ‹A; $B \vdash_!\ [r]_!\ reduce\ p \longleftrightarrow_!\ [reduce\ r]_!\ reduce\ p$›
using *Ann Iff-Ann1 static-reduce* **by** *fastforce*
ultimately have ‹A; $B \vdash_!\ [r]_!\ reduce\ p \longleftrightarrow_!\ reduce'\ (reduce\ r)\ (reduce\ p)$›
using *Iff-Iff Iff-sym* **by** *blast*
moreover have ‹$\forall\, r \in anns\ p.\ B\ r$›
using *Ann.prems* **by** *simp*
then have ‹A; $B \vdash_!\ p \longleftrightarrow_!\ reduce\ p$›
using *Ann.hyps*(*2*) **by** *blast*
then have ‹A; $B \vdash_!\ [r]_!\ reduce\ p \longleftrightarrow_!\ [r]_!\ p$›
using ‹$B\ r$› *Iff-Ann2 Iff-sym* **by** *blast*
ultimately have ‹A; $B \vdash_!\ [r]_!\ p \longleftrightarrow_!\ reduce'\ (reduce\ r)\ (reduce\ p)$›

using *Iff-Iff* **by** *blast*
then show *?case*
by *simp*
qed (*simp-all add*: *PA1*)

lemma *anns-implyP* [*simp*]:
‹*anns* (*ps* $\rightsquigarrow_!$ *q*) = *anns q* ∪ ($\bigcup$ *p* ∈ *set ps*. *anns p*)›
by (*induct ps*) *auto*

lemma *strong-completeness*$_P$′:
assumes ‹*P*; *G* $\models_!$ *p*›
and ‹∀ *r* ∈ *anns p*. *B r*› ‹∀ *q* ∈ *G*. ∀ *r* ∈ *anns q*. *B r*›
and ‹*P*; *lower* ‘ *reduce* ‘ *G* $\models\star$ *lower* (*reduce p*) ⟹
A o lift; *lower* ‘ *reduce* ‘ *G* ⊢ *lower* (*reduce p*)›
shows ‹*A*; *B*; *G* $\vdash_!$ *p*›
proof −
have ‹*P*; *reduce* ‘ *G* $\models_!\star$ *reduce p*›
using *assms*(*1*) *reduce-semantics* **by** *fast*
moreover have ‹*static* (*reduce p*)› ‹∀ *q* ∈ *reduce* ‘ *G*. *static q*›
using *static-reduce* **by** *fast*+
ultimately have ‹*A*; *B*; *reduce* ‘ *G* $\vdash_!$ *reduce p*›
using *assms*(*4*) *strong-static-completeness*′[**where** *G*=‹*reduce* ‘ *G*› **and** *p*=‹*reduce p*›]
by *presburger*
then have ‹∃ *qs*. *set qs* ⊆ *G* ∧ (*A*; *B* $\vdash_!$ *map reduce qs* $\rightsquigarrow_!$ *reduce p*)›
using *set-map-inv* **by** *fast*
then obtain *qs* **where** *qs*: ‹*set qs* ⊆ *G*› **and** ‹*A*; *B* $\vdash_!$ *map reduce qs* $\rightsquigarrow_!$ *reduce p*›
by *blast*
then have ‹*A*; *B* $\vdash_!$ *reduce* (*qs* $\rightsquigarrow_!$ *p*)›
using *reduce-implyP* **by** *metis*
moreover have ‹∀ *r* ∈ *anns* (*qs* $\rightsquigarrow_!$ *p*). *B r*›
using *assms*(*2*−*3*) *qs* **by** *auto*
then have ‹*A*; *B* $\vdash_!$ *qs* $\rightsquigarrow_!$ *p* $\longleftrightarrow_!$ *reduce* (*qs* $\rightsquigarrow_!$ *p*)›
using *Iff-reduce* **by** *blast*
ultimately have ‹*A*; *B* $\vdash_!$ *qs* $\rightsquigarrow_!$ *p*›
using *ConE*(*2*) *PR1* **by** *blast*
then show *?thesis*
using *qs* **by** *blast*
qed

theorem *strong-completeness*$_P$:
assumes ‹*P*; *G* $\models_!$ *p*›
and ‹∀ *r* ∈ *anns p*. *B r*› ‹∀ *q* ∈ *G*. ∀ *r* ∈ *anns q*. *B r*›
and ‹$\bigwedge$*G p*. *P*; *G* $\models\star$ *p* ⟹ *A o lift*; *G* ⊢ *p*›
shows ‹*A*; *B*; *G* $\vdash_!$ *p*›
using *strong-completeness*$_P$′ *assms* **.**

theorem *main*$_P$:

assumes ‹$\bigwedge M\ w\ p.\ A\ p \Longrightarrow P\ M \Longrightarrow w \in \mathcal{W}\ M \Longrightarrow M,\ w \models_!\ p$›
and ‹$\bigwedge M\ r.\ P\ M \Longrightarrow B\ r \Longrightarrow P\ (M[r!])$›
and ‹$\forall\, r \in \mathit{anns}\ p.\ B\ r$› ‹$\forall\, q \in G.\ \forall\, r \in \mathit{anns}\ q.\ B\ r$›
and ‹$\bigwedge G\ p.\ P;\ G \Vdash\!\!\!\models\star\ p \Longrightarrow A\ o\ \mathit{lift};\ G \vdash p$›
shows ‹$P;\ G \Vdash\!\!\!\models_!\ p \longleftrightarrow A;\ B;\ G \vdash_!\ p$›
using *strong-soundness*$_P$[*of* $A\ P\ B\ G\ p$] *strong-completeness*$_P$[*of* $P\ G\ p\ B\ A$] *assms* **by** *blast*

corollary *strong-completeness*$_{PB}$:
assumes ‹$P;\ G \Vdash\!\!\!\models_!\ p$›
and ‹$\bigwedge G\ p.\ P;\ G \Vdash\!\!\!\models\star\ p \Longrightarrow A\ o\ \mathit{lift};\ G \vdash p$›
shows ‹$A;\ (\lambda\text{-}.\ \mathit{True});\ G \vdash_!\ p$›
using *strong-completeness*$_P$[**where** B=‹$\lambda\text{-}.\ \mathit{True}$›] *assms* **by** *blast*

corollary *completeness*$_P{}'$:
assumes ‹$P;\ \{\} \Vdash\!\!\!\models_!\ p$›
and ‹$\forall\, r \in \mathit{anns}\ p.\ B\ r$›
and ‹$\bigwedge p.\ P;\ \{\} \Vdash\!\!\!\models \mathit{lower}\ p \Longrightarrow A\ o\ \mathit{lift} \vdash \mathit{lower}\ p$›
shows ‹$A;\ B \vdash_!\ p$›
using *assms strong-completeness*$_P{}'$[**where** P=P **and** G=‹$\{\}$›] **by** *simp*

corollary *completeness*$_P$:
assumes ‹$P;\ \{\} \Vdash\!\!\!\models_!\ p$›
and ‹$\forall\, r \in \mathit{anns}\ p.\ B\ r$›
and ‹$\bigwedge p.\ P;\ \{\} \Vdash\!\!\!\models p \Longrightarrow A\ o\ \mathit{lift} \vdash p$›
shows ‹$A;\ B \vdash_!\ p$›
using *completeness*$_P{}'$ *assms* **.**

corollary *completeness*$_{PA}$:
assumes ‹$(\lambda\text{-}.\ \mathit{True});\ \{\} \Vdash\!\!\!\models_!\ p$›
shows ‹$A;\ (\lambda\text{-}.\ \mathit{True}) \vdash_!\ p$›
using *assms completeness*$_P$[*of* ‹$\lambda\text{-}.\ \mathit{True}$› p ‹$\lambda\text{-}.\ \mathit{True}$›] *completeness*$_A$ **by** *blast*

9 System PAL + K

abbreviation *SystemPK* (‹- $\vdash_{!K}$ -› [50, 50] 50) **where**
‹$G \vdash_{!K}\ p \equiv (\lambda\text{-}.\ \mathit{False});\ (\lambda\text{-}.\ \mathit{True});\ G \vdash_!\ p$›

lemma *strong-soundness*$_{PK}$: ‹$G \vdash_{!K}\ p \Longrightarrow (\lambda\text{-}.\ \mathit{True});\ G \Vdash\!\!\!\models_!\star\ p$›
using *strong-soundness*$_P$[*of* ‹$\lambda\text{-}.\ \mathit{False}$› ‹$\lambda\text{-}.\ \mathit{True}$›] **by** *fast*

abbreviation *validPK* (‹- $\Vdash\!\!\!\models_{!K}$ -› [50, 50] 50) **where**
‹$G \Vdash\!\!\!\models_{!K}\ p \equiv (\lambda\text{-}.\ \mathit{True});\ G \Vdash\!\!\!\models_!\ p$›

lemma *strong-completeness*$_{PK}$:
assumes ‹$G \Vdash\!\!\!\models_{!K}\ p$›
shows ‹$G \vdash_{!K}\ p$›
using *strong-completeness*$_{PB}$ *assms strong-completeness*$_K$ **unfolding** *comp-apply*
.

theorem $main_{PK}$: ‹$G \Vdash_{!K} p \longleftrightarrow G \vdash_{!K} p$›
using $strong\text{-}soundness_{PK}$[*of* G p] $strong\text{-}completeness_{PK}$[*of* G p] **by** *fast*

corollary ‹$G \Vdash_{!K} p \Longrightarrow (\lambda\text{-}.\ True);\ G \Vdash_{!\star} p$›
using $strong\text{-}soundness_{PK}$[*of* G p] $strong\text{-}completeness_{PK}$[*of* G p] **by** *fast*

10 System PAL + T

Also known as System PAL + M

inductive $AxPT$ :: ‹$'i\ pfm \Rightarrow bool$› **where**
‹$AxPT\ (K_!\ i\ p \longrightarrow_!\ p)$›

abbreviation $SystemPT$ (‹- $\vdash_{!T}$ -› [50, 50] 50) **where**
‹$G \vdash_{!T} p \equiv AxPT;\ (\lambda\text{-}.\ True);\ G \vdash_!\ p$›

lemma $soundness\text{-}AxPT$: ‹$AxPT\ p \Longrightarrow reflexive\ M \Longrightarrow w \in \mathcal{W}\ M \Longrightarrow M,\ w \models_!\ p$›
unfolding $reflexive\text{-}def$ **by** (*induct p rule*: $AxPT.induct$) *simp*

lemma $reflexive\text{-}restrict$: ‹$reflexive\ M \Longrightarrow reflexive\ (M[r!])$›
unfolding $reflexive\text{-}def$ **by** *simp*

lemma $strong\text{-}soundness_{PT}$: ‹$G \vdash_{!T} p \Longrightarrow reflexive;\ G \Vdash_{!\star} p$›
using $strong\text{-}soundness_P$[*of* $AxPT$ $reflexive$ ‹$\lambda\text{-}.\ True$› G p]
$soundness\text{-}AxPT$ $reflexive\text{-}restrict$ **by** *fast*

lemma $AxT\text{-}AxPT$: ‹$AxT = AxPT\ o\ lift$›
unfolding $comp\text{-}apply$ **using** $lower\text{-}lift$
by (*metis* $AxPT.simps$ $AxT.simps$ $lift.simps(5-6)$ $lower.simps(5-6)$)

abbreviation $validPT$ (‹- $\Vdash_{!T}$ -› [50, 50] 50) **where**
‹$G \Vdash_{!T} p \equiv reflexive;\ G \Vdash_!\ p$›

lemma $strong\text{-}completeness_{PT}$:
assumes ‹$G \Vdash_{!T} p$›
shows ‹$G \vdash_{!T} p$›
using $strong\text{-}completeness_{PB}$ $assms$ $strong\text{-}completeness_T$ **unfolding** $AxT\text{-}AxPT$
.

theorem $main_{PT}$: ‹$G \Vdash_{!T} p \longleftrightarrow G \vdash_{!T} p$›
using $strong\text{-}soundness_{PT}$[*of* G p] $strong\text{-}completeness_{PT}$[*of* G p] **by** *fast*

corollary ‹$G \Vdash_{!T} p \Longrightarrow reflexive;\ G \Vdash_{!\star} p$›
using $strong\text{-}soundness_{PT}$[*of* G p] $strong\text{-}completeness_{PT}$[*of* G p] **by** *fast*

11 System PAL + KB

inductive *AxPB* :: ‹*'i pfm* ⇒ *bool*› **where**
‹*AxPB* (*p* ⟶$_!$ *K*$_!$ *i* (*L*$_!$ *i p*))›

abbreviation *SystemPKB* (‹- ⊢$_{!KB}$ -› [*50*, *50*] *50*) **where**
‹*G* ⊢$_{!KB}$ *p* ≡ *AxPB*; (λ-. *True*); *G* ⊢$_!$ *p*›

lemma *soundness-AxPB*: ‹*AxPB p* ⟹ *symmetric M* ⟹ *w* ∈ 𝒲 *M* ⟹ *M*, *w* ⊨$_!$ *p*›
unfolding *symmetric-def* **by** (*induct p rule*: *AxPB.induct*) *auto*

lemma *symmetric-restrict*: ‹*symmetric M* ⟹ *symmetric* (*M*[*r*!])›
unfolding *symmetric-def* **by** *simp*

lemma *strong-soundness*$_{PKB}$: ‹*G* ⊢$_{!KB}$ *p* ⟹ *symmetric*; *G* ⊨$_{!}^{\star}$ *p*›
using *strong-soundness*$_P$[*of AxPB symmetric* ‹λ-. *True*› *G p*]
soundness-AxPB symmetric-restrict **by** *fast*

lemma *AxB-AxPB*: ‹*AxB* = *AxPB o lift*›
proof
fix *p* :: ‹*'i fm*›
show ‹*AxB p* = (*AxPB* ∘ *lift*) *p*›
unfolding *comp-apply* **using** *lower-lift*
by (*smt* (*verit*, *best*) *AxB.simps AxPB.simps lift.simps*(*1*, *5*−*6*) *lower.simps*(*5*−*6*))
qed

abbreviation *validPKB* (‹- ⊨$_{!KB}$ -› [*50*, *50*] *50*) **where**
‹*G* ⊨$_{!KB}$ *p* ≡ *symmetric*; *G* ⊨$_!$ *p*›

lemma *strong-completeness*$_{PKB}$:
assumes ‹*G* ⊨$_{!KB}$ *p*›
shows ‹*G* ⊢$_{!KB}$ *p*›
using *strong-completeness*$_{PB}$ *assms strong-completeness*$_{KB}$ **unfolding** *AxB-AxPB*
.

theorem *main*$_{PKB}$: ‹*G* ⊨$_{!KB}$ *p* ⟷ *G* ⊢$_{!KB}$ *p*›
using *strong-soundness*$_{PKB}$[*of G p*] *strong-completeness*$_{PKB}$[*of G p*] **by** *fast*

corollary ‹*G* ⊨$_{!KB}$ *p* ⟹ *symmetric*; *G* ⊨$_{!}^{\star}$ *p*›
using *strong-soundness*$_{PKB}$[*of G p*] *strong-completeness*$_{PKB}$[*of G p*] **by** *fast*

12 System PAL + K4

inductive *AxP4* :: ‹*'i pfm* ⇒ *bool*› **where**
‹*AxP4* (*K*$_!$ *i p* ⟶$_!$ *K*$_!$ *i* (*K*$_!$ *i p*))›

abbreviation *SystemPK4* (‹- ⊢$_{!K4}$ -› [*50*, *50*] *50*) **where**
‹*G* ⊢$_{!K4}$ *p* ≡ *AxP4*; (λ-. *True*); *G* ⊢$_!$ *p*›

lemma *pos-introspection*:
assumes ‹*transitive M*› ‹$w \in \mathcal{W}\ M$›
shows ‹$M,\ w \models_!\ (K_!\ i\ p \longrightarrow_!\ K_!\ i\ (K_!\ i\ p))$›
proof −
{ assume ‹$M,\ w \models_!\ K_!\ i\ p$›
then have ‹$\forall\, v \in \mathcal{W}\ M \cap \mathcal{K}\ M\ i\ w.\ M,\ v \models_!\ p$›
by *simp*
then have ‹$\forall\, v \in \mathcal{W}\ M \cap \mathcal{K}\ M\ i\ w.\ \forall\, u \in \mathcal{W}\ M \cap \mathcal{K}\ M\ i\ v.\ M,\ u \models_!\ p$›
using ‹*transitive M*› ‹$w \in \mathcal{W}\ M$› **unfolding** *transitive-def* **by** *blast*
then have ‹$\forall\, v \in \mathcal{W}\ M \cap \mathcal{K}\ M\ i\ w.\ M,\ v \models_!\ K_!\ i\ p$›
by *simp*
then have ‹$M,\ w \models_!\ K_!\ i\ (K_!\ i\ p)$›
by *simp* **}**
then show *?thesis*
by *fastforce*
qed

lemma *soundness-AxP4*: ‹*AxP4 p* $\Longrightarrow$ *transitive M* $\Longrightarrow$ $w \in \mathcal{W}\ M \Longrightarrow M,\ w \models_!$
p›
by (*induct p rule*: *AxP4.induct*) (*metis pos-introspection*)

lemma *transitive-restrict*: ‹*transitive M* $\Longrightarrow$ *transitive* (M[*r*!])›
unfolding *transitive-def* **by** (*cases M*) (*metis* (*no-types*, *lifting*) *frame.select-convs*(*1*−*2*)
frame.update-convs(*1*) *mem-Collect-eq restrict.simps*)

lemma *strong-soundness*$_{PK4}$: ‹$G \vdash_{!K4} p \Longrightarrow$ *transitive*; $G \Vdash_{!\star} p$›
using *strong-soundness*$_P$[*of AxP4 transitive* ‹λ-. *True*› $G\ p$]
soundness-AxP4 transitive-restrict **by** *fast*

lemma *Ax4-AxP4*: ‹*Ax4* = *AxP4 o lift*›
proof
fix p :: ‹$'i\ fm$›
show ‹*Ax4* p = (*AxP4* ∘ *lift*) p›
unfolding *comp-apply* **using** *lower-lift*
by (*smt* (*verit*, *best*) *Ax4.simps AxP4.simps lift.simps*(*1*, *5*−*6*) *lower.simps*(*5*−*6*))
qed

abbreviation *validPK4* (‹- $\Vdash_{!K4}$ -› [*50*, *50*] *50*) **where**
‹$G \Vdash_{!K4} p \equiv$ *transitive*; $G \Vdash_!\ p$›

lemma *strong-completeness*$_{PK4}$:
assumes ‹$G \Vdash_{!K4} p$›
shows ‹$G \vdash_{!K4} p$›
using *strong-completeness*$_{PB}$ *assms strong-completeness*$_{K4}$ **unfolding** *Ax4-AxP4*
.

theorem *main*$_{PK4}$: ‹$G \Vdash_{!K4} p \longleftrightarrow G \vdash_{!K4} p$›
using *strong-soundness*$_{PK4}$[*of G p*] *strong-completeness*$_{PK4}$[*of G p*] **by** *fast*

corollary ‹$G \Vdash\models_{!K4} p \Longrightarrow$ *transitive*; $G \Vdash\models_{!\star} p$›
using *strong-soundness*$_{PK4}$[*of* G p] *strong-completeness*$_{PK4}$[*of* G p] **by** *fast*

13 System PAL + K5

inductive *AxP5* :: ‹*'i pfm* ⇒ *bool*› **where**
‹*AxP5* ($L_!$ *i p* ⟶$_!$ $K_!$ *i* ($L_!$ *i p*))›

abbreviation *SystemPK5* (‹- ⊢$_{!K5}$ -› [*50*, *50*] *50*) **where**
‹$G \vdash_{!K5} p \equiv$ *AxP5*; (λ-. *True*); $G \vdash_!$ *p*›

lemma *soundness-AxP5*: ‹*AxP5 p* ⟹ *Euclidean M* ⟹ $w \in \mathcal{W}$ *M* ⟹ *M*, *w* ⊨$_!$ *p*›
by (*induct p rule*: *AxP5.induct*) (*unfold Euclidean-def psemantics.simps*, *blast*)

lemma *Euclidean-restrict*: ‹*Euclidean M* ⟹ *Euclidean* (*M*[*r*!])›
unfolding *Euclidean-def* **by** *auto*

lemma *strong-soundness*$_{PK5}$: ‹$G \vdash_{!K5} p \Longrightarrow$ *Euclidean*; $G \Vdash\models_{!\star} p$›
using *strong-soundness*$_P$[*of AxP5 Euclidean* ‹λ-. *True*› G *p*]
soundness-AxP5 Euclidean-restrict **by** *fast*

lemma *Ax5-AxP5*: ‹*Ax5* = *AxP5 o lift*›
proof
fix *p* :: ‹*'i fm*›
show ‹*Ax5 p* = (*AxP5* ∘ *lift*) *p*›
unfolding *comp-apply* **using** *lower-lift*
by (*smt* (*verit*, *best*) *Ax5.simps AxP5.simps lift.simps*(*1*, *5*−*6*) *lower.simps*(*5*−*6*))
qed

abbreviation *validPK5* (‹- ⊨⊨$_{!K5}$ -› [*50*, *50*] *50*) **where**
‹$G \Vdash\models_{!K5} p \equiv$ *Euclidean*; $G \Vdash\models_!$ *p*›

lemma *strong-completeness*$_{PK5}$:
assumes ‹$G \Vdash\models_{!K5} p$›
shows ‹$G \vdash_{!K5} p$›
using *strong-completeness*$_{PB}$ *assms strong-completeness*$_{K5}$ **unfolding** *Ax5-AxP5*
.

theorem *main*$_{PK5}$: ‹$G \Vdash\models_{!K5} p \longleftrightarrow G \vdash_{!K5} p$›
using *strong-soundness*$_{PK5}$[*of* G p] *strong-completeness*$_{PK5}$[*of* G p] **by** *fast*

corollary ‹$G \Vdash\models_{!K5} p \Longrightarrow$ *Euclidean*; $G \Vdash\models_{!\star} p$›
using *strong-soundness*$_{PK5}$[*of* G p] *strong-completeness*$_{PK5}$[*of* G p] **by** *fast*

14 System PAL + S4

abbreviation *SystemPS4* (‹- $\vdash_{!S4}$ -› [*50*, *50*] *50*) **where**
‹$G \vdash_{!S4} p \equiv AxPT \oplus AxP4$; (λ-. *True*); $G \vdash_{!} p$›

lemma *soundness-AxPT4*: ‹(*AxPT* ⊕ *AxP4*) *p* ⟹ *refltrans M* ⟹ *w* ∈ 𝒲 *M* ⟹ *M*, *w* $\models_{!}$ *p*›
using *soundness-AxPT soundness-AxP4* **by** *fast*

lemma *refltrans-restrict*: ‹*refltrans M* ⟹ *refltrans* (*M*[*r*!])›
using *reflexive-restrict transitive-restrict* **by** *blast*

lemma *strong-soundness*$_{PS4}$: ‹$G \vdash_{!S4} p \Longrightarrow$ *refltrans*; $G \Vdash_{!\star} p$›
using *strong-soundness*$_{P}$[*of* ‹*AxPT* ⊕ *AxP4*› *refltrans* ‹λ-. *True*› *G p*]
soundness-AxPT4 refltrans-restrict **by** *fast*

lemma *AxT4-AxPT4*: ‹(*AxT* ⊕ *Ax4*) = (*AxPT* ⊕ *AxP4*) *o lift*›
using *AxT-AxPT Ax4-AxP4* **unfolding** *comp-apply* **by** *metis*

abbreviation *validPS4* (‹- $\Vdash_{!S4}$ -› [*50*, *50*] *50*) **where**
‹$G \Vdash_{!S4} p \equiv$ *refltrans*; $G \Vdash_{!} p$›

theorem *strong-completeness*$_{PS4}$:
assumes ‹$G \Vdash_{!S4} p$›
shows ‹$G \vdash_{!S4} p$›
using *strong-completeness*$_{PB}$ *assms strong-completeness*$_{S4}$ **unfolding** *AxT4-AxPT4*
.

theorem *main*$_{PS4}$: ‹$G \Vdash_{!S4} p \longleftrightarrow G \vdash_{!S4} p$›
using *strong-soundness*$_{PS4}$[*of G p*] *strong-completeness*$_{PS4}$[*of G p*] **by** *fast*

corollary ‹$G \Vdash_{!S4} p \Longrightarrow$ *refltrans*; $G \Vdash_{!\star} p$›
using *strong-soundness*$_{PS4}$[*of G p*] *strong-completeness*$_{PS4}$[*of G p*] **by** *fast*

15 System PAL + S5

abbreviation *SystemPS5* (‹- $\vdash_{!S5}$ -› [*50*, *50*] *50*) **where**
‹$G \vdash_{!S5} p \equiv AxPT \oplus AxPB \oplus AxP4$; (λ-. *True*); $G \vdash_{!} p$›

abbreviation *AxPTB4* :: ‹'*i pfm* ⇒ *bool*› **where**
‹*AxPTB4* ≡ *AxPT* ⊕ *AxPB* ⊕ *AxP4*›

lemma *soundness-AxPTB4*: ‹*AxPTB4 p* ⟹ *equivalence M* ⟹ *w* ∈ 𝒲 *M* ⟹ *M*, *w* $\models_{!}$ *p*›
using *soundness-AxPT soundness-AxPB soundness-AxP4* **by** *fast*

lemma *equivalence-restrict*: ‹*equivalence M* ⟹ *equivalence* (*M*[*r*!])›
using *reflexive-restrict symmetric-restrict transitive-restrict* **by** *blast*

lemma *strong-soundness*$_{PS5}$: ‹$G \vdash_{!S5} p \Longrightarrow$ *equivalence*; $G \Vdash_{!\star} p$›
using *strong-soundness*$_P$[*of AxPTB4 equivalence* ‹λ-. *True*› G p]
soundness-AxPTB4 equivalence-restrict **by** *fast*

lemma *AxTB4-AxPTB4*: ‹*AxTB4* = *AxPTB4 o lift*›
using *AxT-AxPT AxB-AxPB Ax4-AxP4* **unfolding** *comp-apply* **by** *metis*

abbreviation *validPS5* (‹- $\Vdash_{!S5}$ -› [*50*, *50*] *50*) **where**
‹$G \Vdash_{!S5} p \equiv$ *equivalence*; $G \Vdash_!$ p›

theorem *strong-completeness*$_{PS5}$:
assumes ‹$G \Vdash_{!S5} p$›
shows ‹$G \vdash_{!S5} p$›
using *strong-completeness*$_{PB}$ *assms strong-completeness*$_{S5}$ **unfolding** *AxTB4-AxPTB4*
.

theorem *main*$_{PS5}$: ‹$G \Vdash_{!S5} p \longleftrightarrow G \vdash_{!S5} p$›
using *strong-soundness*$_{PS5}$[*of* G p] *strong-completeness*$_{PS5}$[*of* G p] **by** *fast*

corollary ‹$G \Vdash_{!S5} p \Longrightarrow$ *equivalence*; $G \Vdash_{!\star} p$›
using *strong-soundness*$_{PS5}$[*of* G p] *strong-completeness*$_{PS5}$[*of* G p] **by** *fast*

16 System PAL + S5'

abbreviation *SystemPS5'* (‹- $\vdash_{!S5}$'' -› [*50*, *50*] *50*) **where**
‹$G \vdash_{!S5}' p \equiv$ *AxPT* $\oplus$ *AxP5*; (λ-. *True*); $G \vdash_!$ p›

abbreviation *AxPT5* :: ‹'*i pfm* $\Rightarrow$ *bool*› **where**
‹*AxPT5* $\equiv$ *AxPT* $\oplus$ *AxP5*›

lemma *soundness-AxPT5*: ‹*AxPT5* $p \Longrightarrow$ *equivalence* $M \Longrightarrow w \in \mathcal{W}$ $M \Longrightarrow M$, $w \models_!$ p›
using *soundness-AxPT soundness-AxPT soundness-AxP5 symm-trans-Euclid* **by** *fast*

lemma *strong-soundness*$_{PS5}'$: ‹$G \vdash_{!S5}' p \Longrightarrow$ *equivalence*; $G \Vdash_{!\star} p$›
using *strong-soundness*$_P$[*of AxPT5 equivalence* ‹λ-. *True*› G p]
soundness-AxPT5 equivalence-restrict **by** *fast*

lemma *AxT5-AxPT5*: ‹*AxT5* = *AxPT5 o lift*›
using *AxT-AxPT Ax5-AxP5* **unfolding** *comp-apply* **by** *metis*

theorem *strong-completeness*$_{PS5}'$:
assumes ‹$G \Vdash_{!S5} p$›
shows ‹$G \vdash_{!S5}' p$›
using *strong-completeness*$_{PB}$ *assms strong-completeness*$_{S5}'$ **unfolding** *AxT5-AxPT5*
.

theorem *main*$_{PS5}'$: ‹$G \Vdash_{!S5} p \longleftrightarrow G \vdash_{!S5}' p$›

using *strong-soundness*$_{PS5}$'[*of G p*] *strong-completeness*$_{PS5}$'[*of G p*] **by** *fast*

end

References

[1] Y. Wang and Q. Cao. On axiomatizations of public announcement logic. *Synthese*, 190(Supplement-1):103–134, 2013.