

Public Announcement Logic

Asta Halkjær From

March 17, 2025

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 $\text{PAL} + \text{DIST!} + \text{NEC!}$ [1]. The completeness proofs build on the Epistemic Logic theory.

Contents

1	Syntax	3
2	Semantics	3
3	Soundness of Reduction	4
4	Chains of Implications	5
5	Proof System	6
6	Soundness	8
7	Strong Soundness	8
8	Completeness	10
9	System $\text{PAL} + \text{K}$	17
10	System $\text{PAL} + \text{T}$	18
11	System $\text{PAL} + \text{KB}$	19
12	System $\text{PAL} + \text{K4}$	19
13	System $\text{PAL} + \text{K5}$	21
14	System $\text{PAL} + \text{S4}$	22

15 System PAL + S5	22
16 System PAL + S5'	23

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

1 Syntax

datatype *'i pfm*
 $= FF \langle \perp_! \rangle$
 $| Pro' id \langle Pro_! \rangle$
 $| Dis \langle 'i pfm \rangle \langle 'i pfm \rangle \text{ (infixr } \langle \vee_! \rangle 60)$
 $| Con \langle 'i pfm \rangle \langle 'i pfm \rangle \text{ (infixr } \langle \wedge_! \rangle 65)$
 $| Imp \langle 'i pfm \rangle \langle 'i pfm \rangle \text{ (infixr } \langle \longrightarrow_! \rangle 55)$
 $| K' 'i \langle 'i pfm \rangle \langle K_! \rangle$
 $| Ann \langle 'i pfm \rangle \langle 'i pfm \rangle \langle [-]_! \rightarrow [80, 80] 80)$

abbreviation *PIff* $:: \langle 'i pfm \Rightarrow 'i pfm \Rightarrow 'i pfm \rangle \text{ (infixr } \langle \longleftrightarrow_! \rangle 55)$ **where**
 $\langle p \longleftrightarrow_! q \equiv (p \longrightarrow_! q) \wedge_! (q \longrightarrow_! p) \rangle$

abbreviation *PNeg* $\langle \neg_! \rightarrow [70] 70)$ **where**
 $\langle \neg_! p \equiv p \longrightarrow_! \perp_! \rangle$

abbreviation *PL* $\langle L_! \rangle$ **where**
 $\langle L_! i p \equiv (\neg_! (K_! i (\neg_! p))) \rangle$

primrec *anns* $:: \langle 'i pfm \Rightarrow 'i pfm \text{ set} \rangle$ **where**
 $\langle anns \perp_! = \{\} \rangle$
 $| \langle anns (Pro_! -) = \{\} \rangle$
 $| \langle anns (p \vee_! q) = (anns p \cup anns q) \rangle$
 $| \langle anns (p \wedge_! q) = (anns p \cup anns q) \rangle$
 $| \langle anns (p \longrightarrow_! q) = (anns p \cup anns q) \rangle$
 $| \langle anns (K_! i p) = anns p \rangle$
 $| \langle anns ([r]_! p) = \{r\} \cup anns r \cup anns p \rangle$

2 Semantics

fun

psemantics $:: \langle ('i, 'w) \text{ kripke} \Rightarrow 'w \Rightarrow 'i pfm \Rightarrow bool \rangle \langle \langle -, - \models_! \rangle \rightarrow [50, 50, 50] 50) \text{ and}$
restrict $:: \langle ('i, 'w) \text{ kripke} \Rightarrow 'i pfm \Rightarrow ('i, 'w) \text{ kripke} \rangle \langle \langle [-]_! \rangle [50, 50] 50) \text{ where}$
 $\langle M, w \models_! \perp_! \longleftrightarrow False \rangle$
 $| \langle M, w \models_! Pro_! x \longleftrightarrow \pi M w x \rangle$
 $| \langle M, w \models_! p \vee_! q \longleftrightarrow M, w \models_! p \vee M, w \models_! q \rangle$
 $| \langle M, w \models_! p \wedge_! q \longleftrightarrow M, w \models_! p \wedge M, w \models_! q \rangle$
 $| \langle M, w \models_! p \longrightarrow_! q \longleftrightarrow M, w \models_! p \longrightarrow M, w \models_! q \rangle$
 $| \langle M, w \models_! K_! i p \longleftrightarrow (\forall v \in \mathcal{W} M \cap \mathcal{K} M i w. M, v \models_! p) \rangle$
 $| \langle M, w \models_! [r]_! p \longleftrightarrow M, w \models_! r \longrightarrow M[r!], w \models_! p \rangle$
 $| \langle M[r!] = M \langle \mathcal{W} := \{w. w \in \mathcal{W} M \wedge M, w \models_! r\} \rangle \rangle$

abbreviation $\text{validPStar} :: \langle \langle 'i, 'w \rangle \text{kripke} \Rightarrow \text{bool} \rangle \Rightarrow 'i \text{ pfm set} \Rightarrow 'i \text{ pfm} \Rightarrow \text{bool} \rangle$

$\langle \langle -; - \models_{!} \star \rightarrow [50, 50, 50] 50 \rangle \text{ where} \langle P; G \models_{!} \star p \equiv \forall M. P M \longrightarrow (\forall w \in \mathcal{W} M. (\forall q \in G. M, w \models_{!} q) \longrightarrow M, w \models_{!} p) \rangle \rangle$

primrec $\text{static} :: \langle 'i \text{ pfm} \Rightarrow \text{bool} \rangle \text{ where}$

$\langle \text{static } \perp_{!} = \text{True} \rangle$
 $| \langle \text{static } (\text{Pro}_{!} -) = \text{True} \rangle$
 $| \langle \text{static } (p \vee_{!} q) = (\text{static } p \wedge \text{static } q) \rangle$
 $| \langle \text{static } (p \wedge_{!} q) = (\text{static } p \wedge \text{static } q) \rangle$
 $| \langle \text{static } (p \longrightarrow_{!} q) = (\text{static } p \wedge \text{static } q) \rangle$
 $| \langle \text{static } (K_{!} i p) = \text{static } p \rangle$
 $| \langle \text{static } ([r]_{!} p) = \text{False} \rangle$

primrec $\text{lower} :: \langle 'i \text{ pfm} \Rightarrow 'i \text{ fm} \rangle \text{ where}$

$\langle \text{lower } \perp_{!} = \perp \rangle$
 $| \langle \text{lower } (\text{Pro}_{!} x) = \text{Pro } x \rangle$
 $| \langle \text{lower } (p \vee_{!} q) = (\text{lower } p \vee \text{lower } q) \rangle$
 $| \langle \text{lower } (p \wedge_{!} q) = (\text{lower } p \wedge \text{lower } q) \rangle$
 $| \langle \text{lower } (p \longrightarrow_{!} q) = (\text{lower } p \longrightarrow \text{lower } q) \rangle$
 $| \langle \text{lower } (K_{!} i p) = K i (\text{lower } p) \rangle$
 $| \langle \text{lower } ([r]_{!} p) = \text{undefined} \rangle$

primrec $\text{lift} :: \langle 'i \text{ fm} \Rightarrow 'i \text{ pfm} \rangle \text{ where}$

$\langle \text{lift } \perp = \perp_{!} \rangle$
 $| \langle \text{lift } (\text{Pro } x) = \text{Pro}_{!} x \rangle$
 $| \langle \text{lift } (p \vee q) = (\text{lift } p \vee_{!} \text{lift } q) \rangle$
 $| \langle \text{lift } (p \wedge q) = (\text{lift } p \wedge_{!} \text{lift } q) \rangle$
 $| \langle \text{lift } (p \longrightarrow q) = (\text{lift } p \longrightarrow_{!} \text{lift } q) \rangle$
 $| \langle \text{lift } (K i p) = K_{!} i (\text{lift } p) \rangle$

lemma *lower-semantics:*

assumes $\langle \text{static } p \rangle$
shows $\langle (M, w \models \text{lower } p) \longleftrightarrow (M, w \models_{!} p) \rangle$
using *assms* **by** $(\text{induct } p \text{ arbitrary: } w) \text{ simp-all}$

lemma *lift-semantics:* $\langle (M, w \models p) \longleftrightarrow (M, w \models_{!} \text{lift } p) \rangle$

by $(\text{induct } p \text{ arbitrary: } w) \text{ simp-all}$

lemma *lower-lift:* $\langle \text{lower } (\text{lift } p) = p \rangle$

by $(\text{induct } p) \text{ simp-all}$

lemma *lift-lower:* $\langle \text{static } p \Longrightarrow \text{lift } (\text{lower } p) = p \rangle$

by $(\text{induct } p) \text{ simp-all}$

3 Soundness of Reduction

primrec $\text{reduce}' :: \langle 'i \text{ pfm} \Rightarrow 'i \text{ pfm} \Rightarrow 'i \text{ pfm} \rangle \text{ where}$

```

  <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* :: $\langle (id \Rightarrow bool) \Rightarrow ('i\ pfm \Rightarrow bool) \Rightarrow 'i\ pfm \Rightarrow bool \rangle$ **where**
 $\langle peval\ -\ -\ \perp! = False \rangle$
 $\langle peval\ g\ -\ (Pro! x) = g\ x \rangle$
 $\langle peval\ g\ h\ (p\ \vee! q) = (peval\ g\ h\ p\ \vee\ peval\ g\ h\ q) \rangle$
 $\langle peval\ g\ h\ (p\ \wedge! q) = (peval\ g\ h\ p\ \wedge\ peval\ g\ h\ q) \rangle$
 $\langle peval\ g\ h\ (p\ \longrightarrow! q) = (peval\ g\ h\ p\ \longrightarrow\ peval\ g\ h\ q) \rangle$
 $\langle peval\ -\ h\ (K! i\ p) = h\ (K! i\ p) \rangle$
 $\langle peval\ -\ h\ ([r]! p) = h\ ([r]! p) \rangle$

abbreviation $\langle ptautology\ p \equiv \forall g\ h. peval\ g\ h\ p \rangle$

inductive *PAK* :: $\langle ('i\ pfm \Rightarrow bool) \Rightarrow ('i\ pfm \Rightarrow bool) \Rightarrow 'i\ pfm \Rightarrow bool \rangle$
 $\langle \langle -, -\vdash! \rightarrow [50, 50, 50] 50 \rangle$
for *A B* :: $\langle 'i\ pfm \Rightarrow bool \rangle$ **where**
 $PA1: \langle ptautology\ p \Longrightarrow A; B\vdash! p \rangle$
 $PA2: \langle A; B\vdash! K! i\ p\ \wedge! K! i\ (p\ \longrightarrow! q) \longrightarrow! K! i\ q \rangle$
 $PAX: \langle A\ p \Longrightarrow A; B\vdash! p \rangle$
 $PR1: \langle A; B\vdash! p \Longrightarrow A; B\vdash! p\ \longrightarrow! q \Longrightarrow A; B\vdash! q \rangle$
 $PR2: \langle A; B\vdash! p \Longrightarrow A; B\vdash! K! i\ p \rangle$
 $PAnn: \langle A; B\vdash! p \Longrightarrow B\ r \Longrightarrow A; B\vdash! [r]! p \rangle$
 $PPF: \langle A; B\vdash! [r]! \perp! \longleftrightarrow! (r\ \longrightarrow! \perp!) \rangle$
 $PPro: \langle A; B\vdash! [r]! Pro! x \longleftrightarrow! (r\ \longrightarrow! Pro! x) \rangle$
 $PDis: \langle A; B\vdash! [r]! (p\ \vee! q) \longleftrightarrow! [r]! p\ \vee! [r]! q \rangle$
 $PCon: \langle A; B\vdash! [r]! (p\ \wedge! q) \longleftrightarrow! [r]! p\ \wedge! [r]! q \rangle$
 $PImp: \langle A; B\vdash! [r]! (p\ \longrightarrow! q) \longleftrightarrow! ([r]! p\ \longrightarrow! [r]! q) \rangle$
 $PK: \langle A; B\vdash! [r]! K! i\ p \longleftrightarrow! (r\ \longrightarrow! K! i\ ([r]! p)) \rangle$

abbreviation *PAK-assms* $\langle \langle -, -; -\vdash! \rightarrow [50, 50, 50, 50] 50 \rangle$ **where**
 $\langle A; B; G\vdash! p \equiv \exists qs. set\ qs \subseteq G \wedge (A; B\vdash! qs \rightsquigarrow! p) \rangle$

lemma *eval-peval*: $\langle eval\ h\ (g\ o\ lift)\ p = peval\ h\ g\ (lift\ p) \rangle$
by (*induct p*) *simp-all*

lemma *tautology-ptautology*: $\langle tautology\ p \Longrightarrow ptautology\ (lift\ p) \rangle$
using *eval-peval* **by** *blast*

theorem *AK-PAK*: $\langle A\ o\ lift\vdash p \Longrightarrow A; B\vdash! lift\ p \rangle$
by (*induct p rule: AK.induct*) (*auto intro: PAK.intros(1-5) simp: tautology-ptautology*)

abbreviation *validP*
 $:: \langle ((i, 'i\ fm\ set)\ kripke \Rightarrow bool) \Rightarrow 'i\ pfm\ set \Rightarrow 'i\ pfm \Rightarrow bool \rangle$
 $\langle \langle -, -\models! \rightarrow [50, 50, 50] 50 \rangle$
where $\langle P; G\models! p \equiv P; G\models! \star p \rangle$

lemma *set-map-inv*:
assumes $\langle set\ xs \subseteq f\ 'X \rangle$
shows $\langle \exists ys. set\ ys \subseteq X \wedge map\ f\ ys = xs \rangle$

```

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.prem 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 $\langle A; B \vdash! p \rangle$
 using *static-completeness'* *assms* .

corollary

assumes $\langle \text{static } p \rangle \langle (\lambda-. \text{ True}); \{\} \models! p \rangle$
 shows $\langle A; B \vdash! p \rangle$
 using *assms static-completeness*[**where** $P = \langle \lambda-. \text{ True} \rangle$ **and** $p = p$] *completeness_A*
 by *blast*

6 Soundness

lemma *peval-semantics*:

$\langle \text{peval } (\text{val } w) (\lambda q. (\mathcal{W} = W, \mathcal{K} = r, \pi = \text{val}), w \models! q) p = ((\mathcal{W} = W, \mathcal{K} = r, \pi = \text{val}), w \models! p) \rangle$
 by (*induct p*) *simp-all*

lemma *ptautology*:

assumes $\langle \text{ptautology } p \rangle$
 shows $\langle M, w \models! p \rangle$

proof –

from *assms* **have** $\langle \text{peval } (g \ w) (\lambda q. (\mathcal{W} = W, \mathcal{K} = r, \pi = g), w \models! q) p \rangle$ **for** $W \ g \ r$
 by *simp*
 then **have** $\langle (\mathcal{W} = W, \mathcal{K} = r, \pi = g), w \models! p \rangle$ **for** $W \ g \ r$
 using *peval-semantics* **by** *fast*
 then **show** $\langle M, w \models! p \rangle$
 by (*metis kripke.cases*)

qed

theorem *soundness_P*:

assumes
 $\langle \bigwedge M \ p \ w. A \ p \implies P \ M \implies w \in \mathcal{W} \ M \implies M, w \models! p \rangle$
 $\langle \bigwedge M \ r. P \ M \implies B \ r \implies P \ (M[r!]) \rangle$
 shows $\langle A; B \vdash! p \implies P \ M \implies w \in \mathcal{W} \ M \implies M, w \models! p \rangle$

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 $\langle (\lambda-. \text{ False}); B \vdash! p \implies w \in \mathcal{W} \ M \implies M, w \models! p \rangle$

using *soundness_P*[**where** $P = \langle \lambda-. \text{ True} \rangle$] **by** *metis*

7 Strong Soundness

lemma *ptautology-imply-superset*:

assumes $\langle \text{set } ps \subseteq \text{set } qs \rangle$

shows $\langle \text{ptautology } (ps \rightsquigarrow! r \longrightarrow! qs \rightsquigarrow! r) \rangle$

```

proof (rule ccontr)
  assume  $\langle \neg \text{?thesis} \rangle$ 
  then obtain  $g\ h$  where  $\langle \neg \text{peval } g\ h\ (ps \rightsquigarrow! r \longrightarrow! qs \rightsquigarrow! r) \rangle$ 
    by blast
  then have  $\langle \text{peval } g\ h\ (ps \rightsquigarrow! r) \rangle \langle \neg \text{peval } g\ h\ (qs \rightsquigarrow! r) \rangle$ 
    by simp-all
  then consider (np)  $\langle \exists p \in \text{set } ps. \neg \text{peval } g\ h\ p \rangle \mid (r) \langle \forall p \in \text{set } ps. \text{peval } g\ h\ p \rangle$ 
     $\langle \text{peval } g\ h\ r \rangle$ 
    by (induct ps) auto
  then show False
proof cases
  case np
    then have  $\langle \exists p \in \text{set } qs. \neg \text{peval } g\ h\ p \rangle$ 
      using  $\langle \text{set } ps \subseteq \text{set } qs \rangle$  by blast
    then have  $\langle \text{peval } g\ h\ (qs \rightsquigarrow! r) \rangle$ 
      by (induct qs) simp-all
    then show ?thesis
      using  $\langle \neg \text{peval } g\ h\ (qs \rightsquigarrow! r) \rangle$  by blast
  next
    case r
    then have  $\langle \text{peval } g\ h\ (qs \rightsquigarrow! r) \rangle$ 
      by (induct qs) simp-all
    then show ?thesis
      using  $\langle \neg \text{peval } g\ h\ (qs \rightsquigarrow! r) \rangle$  by blast
  qed
qed

lemma PK-imply-weaken:
  assumes  $\langle A; B \vdash! ps \rightsquigarrow! q \rangle \langle \text{set } ps \subseteq \text{set } ps' \rangle$ 
  shows  $\langle A; B \vdash! ps' \rightsquigarrow! q \rangle$ 
proof -
  have  $\langle \text{ptautology } (ps \rightsquigarrow! q \longrightarrow! ps' \rightsquigarrow! q) \rangle$ 
    using  $\langle \text{set } ps \subseteq \text{set } ps' \rangle$  ptautology-imply-superset by blast
  then have  $\langle A; B \vdash! ps \rightsquigarrow! q \longrightarrow! ps' \rightsquigarrow! q \rangle$ 
    using PA1 by blast
  then show ?thesis
    using  $\langle A; B \vdash! ps \rightsquigarrow! q \rangle$  PR1 by blast
qed

lemma implyP-append:  $\langle (ps @ ps' \rightsquigarrow! q) = (ps \rightsquigarrow! ps' \rightsquigarrow! q) \rangle$ 
  by (induct ps) simp-all

lemma PK-ImpI:
  assumes  $\langle A; B \vdash! p \# G \rightsquigarrow! q \rangle$ 
  shows  $\langle A; B \vdash! G \rightsquigarrow! (p \longrightarrow! q) \rangle$ 
proof -
  have  $\langle \text{set } (p \# G) \subseteq \text{set } (G @ [p]) \rangle$ 
    by simp
  then have  $\langle A; B \vdash! G @ [p] \rightsquigarrow! q \rangle$ 

```

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

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

8 Completeness

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

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

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

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

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

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

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

lemma *K'-A2'*: $\langle A; B \vdash_! K_! i (p \longrightarrow_! q) \longrightarrow_! K_! i p \longrightarrow_! K_! i q \rangle$
proof –

have $\langle A; B \vdash! K! i p \wedge! K! i (p \longrightarrow! q) \longrightarrow! K! i q \rangle$
 using *PA2* by *fast*
 moreover have $\langle A; B \vdash! (P \wedge! Q \longrightarrow! R) \longrightarrow! (Q \longrightarrow! P \longrightarrow! R) \rangle$ for $P Q$
R
 by (*simp add: PA1*)
 ultimately show *?thesis*
 using *PR1* by *fast*
 qed

lemma *K'-map*:
 assumes $\langle A; B \vdash! p \longrightarrow! q \rangle$
 shows $\langle A; B \vdash! K! i p \longrightarrow! K! i q \rangle$
 proof –
 note $\langle A; B \vdash! p \longrightarrow! q \rangle$
 then have $\langle A; B \vdash! K! i (p \longrightarrow! q) \rangle$
 using *PR2* by *fast*
 moreover have $\langle A; B \vdash! K! i (p \longrightarrow! q) \longrightarrow! K! i p \longrightarrow! K! i q \rangle$
 using *K'-A2'* by *fast*
 ultimately show *?thesis*
 using *PR1* by *fast*
 qed

lemma *ConI*:
 assumes $\langle A; B \vdash! p \rangle \langle A; B \vdash! q \rangle$
 shows $\langle A; B \vdash! p \wedge! q \rangle$
 proof –
 have $\langle A; B \vdash! p \longrightarrow! q \longrightarrow! p \wedge! q \rangle$
 by (*simp add: PA1*)
 then show *?thesis*
 using *assms PR1* by *blast*
 qed

lemma *Iff-wk*:
 assumes $\langle A; B \vdash! p \longleftrightarrow! q \rangle$
 shows $\langle A; B \vdash! (r \longrightarrow! p) \longleftrightarrow! (r \longrightarrow! q) \rangle$
 proof –
 have $\langle A; B \vdash! (p \longleftrightarrow! q) \longrightarrow! ((r \longrightarrow! p) \longleftrightarrow! (r \longrightarrow! q)) \rangle$
 by (*simp add: PA1*)
 then show *?thesis*
 using *assms PR1* by *blast*
 qed

lemma *Iff-reduce'*:
 assumes $\langle \text{static } p \rangle$
 shows $\langle A; B \vdash! [r]! p \longleftrightarrow! \text{reduce}' r p \rangle$
 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  $\langle A; B \vdash [r]! p \vee [r]! q \longleftrightarrow! \text{reduce}' r (p \vee [r]! q) \rangle$ 
    using Iff-Dis by fastforce
  moreover have  $\langle A; B \vdash ([r]! p \vee [r]! q) \longleftrightarrow! ([r]! (p \vee [r]! q)) \rangle$ 
    using PDis Iff-sym by fastforce
  ultimately show ?case
    using PA1 PR1 Iff-Iff by blast
next
  case (Con p q)
  then have  $\langle A; B \vdash [r]! p \wedge [r]! q \longleftrightarrow! \text{reduce}' r (p \wedge [r]! q) \rangle$ 
    using Iff-Con by fastforce
  moreover have  $\langle A; B \vdash ([r]! p \wedge [r]! q) \longleftrightarrow! ([r]! (p \wedge [r]! q)) \rangle$ 
    using PCon Iff-sym by fastforce
  ultimately show ?case
    using PA1 PR1 Iff-Iff by blast
next
  case (Imp p q)
  then have  $\langle A; B \vdash ([r]! p \longrightarrow [r]! q) \longleftrightarrow! \text{reduce}' r (p \longrightarrow [r]! q) \rangle$ 
    using Iff-Imp by fastforce
  moreover have  $\langle A; B \vdash ([r]! p \longrightarrow [r]! q) \longleftrightarrow! ([r]! (p \longrightarrow [r]! q)) \rangle$ 
    using PImp Iff-sym by fastforce
  ultimately show ?case
    using PA1 PR1 Iff-Iff by blast
next
  case (K' i p)
  then have  $\langle A; B \vdash [r]! p \longleftrightarrow! \text{reduce}' r p \rangle$ 
    by simp
  then have  $\langle A; B \vdash K! i ([r]! p) \longleftrightarrow! K! i (\text{reduce}' r p) \rangle$ 
    using K'-map ConE ConI by metis
  moreover have  $\langle A; B \vdash [r]! K! i p \longleftrightarrow! r \longrightarrow! K! i ([r]! p) \rangle$ 
    using PK .
  ultimately have  $\langle A; B \vdash [r]! K! i p \longleftrightarrow! r \longrightarrow! K! i (\text{reduce}' r p) \rangle$ 
    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 $r: \langle A; B \vdash r \longleftrightarrow! r' \rangle$ and $\langle \text{static } p \rangle$

```

shows  $\langle A; B \vdash_! [r]_! p \longleftrightarrow_! [r']_! p \rangle$ 
using assms( $\mathcal{Q}-$ )
proof (induct p)
  case FF
  have  $\langle A; B \vdash_! (r \longleftrightarrow_! r') \longrightarrow_! ((r \longrightarrow_! \perp_!) \longleftrightarrow_! (r' \longrightarrow_! \perp_!)) \rangle$ 
    by (auto intro: PA1)
  then have  $\langle A; B \vdash_! (r \longrightarrow_! \perp_!) \longleftrightarrow_! (r' \longrightarrow_! \perp_!) \rangle$ 
    using r PR1 by blast
  then show ?case
    by (meson PFF Iff-Iff Iff-sym)
next
  case (Pro' x)
  have  $\langle A; B \vdash_! (r \longleftrightarrow_! r') \longrightarrow_! ((r \longrightarrow_! \text{Pro}_! x) \longleftrightarrow_! (r' \longrightarrow_! \text{Pro}_! x)) \rangle$ 
    by (auto intro: PA1)
  then have  $\langle A; B \vdash_! (r \longrightarrow_! \text{Pro}_! x) \longleftrightarrow_! (r' \longrightarrow_! \text{Pro}_! x) \rangle$ 
    using r PR1 by blast
  then show ?case
    by (meson PPro Iff-Iff Iff-sym)
next
  case (Dis p q)
  then have  $\langle A; B \vdash_! [r]_! p \vee_! [r]_! q \longleftrightarrow_! [r']_! p \vee_! [r']_! q \rangle$ 
    by (simp add: Iff-Dis)
  then show ?case
    by (meson PDis Iff-Iff Iff-sym)
next
  case (Con p q)
  then have  $\langle A; B \vdash_! [r]_! p \wedge_! [r]_! q \longleftrightarrow_! [r']_! p \wedge_! [r']_! q \rangle$ 
    by (simp add: Iff-Con)
  then show ?case
    by (meson PCon Iff-Iff Iff-sym)
next
  case (Imp p q)
  then have  $\langle A; B \vdash_! ([r]_! p \longrightarrow_! [r]_! q) \longleftrightarrow_! ([r']_! p \longrightarrow_! [r']_! q) \rangle$ 
    by (simp add: Iff-Imp)
  then show ?case
    by (meson PImp Iff-Iff Iff-sym)
next
  case (K' i p)
  then have  $\langle A; B \vdash_! [r]_! p \longleftrightarrow_! [r']_! p \rangle$ 
    by simp
  then have  $\langle A; B \vdash_! K_! i ([r]_! p) \longleftrightarrow_! K_! i ([r']_! p) \rangle$ 
    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  $\langle A; B \vdash_! p \longleftrightarrow_! p' \rangle$  and  $\langle B \ r \rangle$ 
  shows  $\langle A; B \vdash_! [r]_! p \longleftrightarrow_! [r]_! p' \rangle$ 
  using assms PAnn ConE ConI PImp PR1 by metis

lemma Iff-reduce:
  assumes  $\langle \forall r \in \text{anns } p. B \ r \rangle$ 
  shows  $\langle A; B \vdash_! p \longleftrightarrow_! \text{reduce } p \rangle$ 
  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
     $\langle A; B \vdash_! K_! i \ p \longrightarrow_! K_! i \ (\text{reduce } p) \rangle$ 
     $\langle A; B \vdash_! K_! i \ (\text{reduce } p) \longrightarrow_! K_! i \ p \rangle$ 
    using K'-map ConE by fastforce+
  then have  $\langle A; B \vdash_! K_! i \ p \longleftrightarrow_! K_! i \ (\text{reduce } p) \rangle$ 
    using ConI by blast
  then show ?case
    by simp
next
  case (Ann r p)
  then have  $\langle B \ r \rangle$ 
    by simp
  have  $\langle A; B \vdash_! [\text{reduce } r]_! \text{reduce } p \longleftrightarrow_! \text{reduce}' (\text{reduce } r) (\text{reduce } p) \rangle$ 
    using Iff-reduce' static-reduce by blast
  moreover have  $\langle A; B \vdash_! [r]_! \text{reduce } p \longleftrightarrow_! [\text{reduce } r]_! \text{reduce } p \rangle$ 
    using Ann Iff-Ann1 static-reduce by fastforce
  ultimately have  $\langle A; B \vdash_! [r]_! \text{reduce } p \longleftrightarrow_! \text{reduce}' (\text{reduce } r) (\text{reduce } p) \rangle$ 
    using Iff-Iff Iff-sym by blast
  moreover have  $\langle \forall r \in \text{anns } p. B \ r \rangle$ 
    using Ann.prems by simp
  then have  $\langle A; B \vdash_! p \longleftrightarrow_! \text{reduce } p \rangle$ 
    using Ann.hyps(2) by blast
  then have  $\langle A; B \vdash_! [r]_! \text{reduce } p \longleftrightarrow_! [r]_! p \rangle$ 
    using  $\langle B \ r \rangle$  Iff-Ann2 Iff-sym by blast
  ultimately have  $\langle A; B \vdash_! [r]_! p \longleftrightarrow_! \text{reduce}' (\text{reduce } r) (\text{reduce } p) \rangle$ 

```

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

lemma *anns-implyP* [*simp*]:
 $\langle \text{anns } (ps \rightsquigarrow_! q) = \text{anns } q \cup (\bigcup p \in \text{set } ps. \text{anns } p) \rangle$
 by (*induct ps*) *auto*

lemma *strong-completeness_{P'}*:
 assumes $\langle P; G \models_! p \rangle$
 and $\langle \forall r \in \text{anns } p. B \ r \rangle \langle \forall q \in G. \forall r \in \text{anns } q. B \ r \rangle$
 and $\langle P; \text{lower } ' \text{reduce } ' G \models_\star \text{lower } (\text{reduce } p) \implies$
 $A \text{ o lift}; \text{lower } ' \text{reduce } ' G \vdash \text{lower } (\text{reduce } p) \rangle$
 shows $\langle A; B; G \vdash_! p \rangle$
proof –
 have $\langle P; \text{reduce } ' G \models_\star \text{reduce } p \rangle$
 using *assms(1) reduce-semantics* by *fast*
 moreover have $\langle \text{static } (\text{reduce } p) \rangle \langle \forall q \in \text{reduce } ' G. \text{static } q \rangle$
 using *static-reduce* by *fast+*
 ultimately have $\langle A; B; \text{reduce } ' G \vdash_! \text{reduce } p \rangle$
 using *assms(4) strong-static-completeness'* [where $G = \langle \text{reduce } ' G \rangle$ and $p = \langle \text{reduce } p \rangle$]
 by *presburger*
 then have $\langle \exists qs. \text{set } qs \subseteq G \wedge (A; B \vdash_! \text{map reduce } qs \rightsquigarrow_! \text{reduce } p) \rangle$
 using *set-map-inv* by *fast*
 then obtain *qs* where *qs*: $\langle \text{set } qs \subseteq G \rangle$ and $\langle A; B \vdash_! \text{map reduce } qs \rightsquigarrow_! \text{reduce } p \rangle$
 by *blast*
 then have $\langle A; B \vdash_! \text{reduce } (qs \rightsquigarrow_! p) \rangle$
 using *reduce-implyP* by *metis*
 moreover have $\langle \forall r \in \text{anns } (qs \rightsquigarrow_! p). B \ r \rangle$
 using *assms(2-3) qs* by *auto*
 then have $\langle A; B \vdash_! qs \rightsquigarrow_! p \longleftrightarrow_! \text{reduce } (qs \rightsquigarrow_! p) \rangle$
 using *Iff-reduce* by *blast*
 ultimately have $\langle A; B \vdash_! qs \rightsquigarrow_! p \rangle$
 using *ConE(2) PR1* by *blast*
 then show *?thesis*
 using *qs* by *blast*
 qed

theorem *strong-completeness_P*:
 assumes $\langle P; G \models_! p \rangle$
 and $\langle \forall r \in \text{anns } p. B \ r \rangle \langle \forall q \in G. \forall r \in \text{anns } q. B \ r \rangle$
 and $\langle \bigwedge G \ p. P; G \models_\star p \implies A \text{ o lift}; G \vdash p \rangle$
 shows $\langle A; B; G \vdash_! p \rangle$
 using *strong-completeness_{P'} assms* .

theorem *main_P*:

assumes $\langle \bigwedge M w p. A p \implies P M \implies w \in \mathcal{W} M \implies M, w \models! p \rangle$
and $\langle \bigwedge M r. P M \implies B r \implies P (M[r!]) \rangle$
and $\langle \forall r \in \text{anns } p. B r \rangle \langle \forall q \in G. \forall r \in \text{anns } q. B r \rangle$
and $\langle \bigwedge G p. P; G \models_\star p \implies A \text{ o lift}; G \vdash p \rangle$
shows $\langle P; G \models! p \longleftrightarrow A; B; G \vdash! p \rangle$
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 $\langle P; G \models! p \rangle$
and $\langle \bigwedge G p. P; G \models_\star p \implies A \text{ o lift}; G \vdash p \rangle$
shows $\langle A; (\lambda-. \text{True}); G \vdash! p \rangle$
using *strong-completeness_P*[**where** $B = \langle \lambda-. \text{True} \rangle$] *assms by blast*

corollary *completeness_{P'}*:

assumes $\langle P; \{\} \models! p \rangle$
and $\langle \forall r \in \text{anns } p. B r \rangle$
and $\langle \bigwedge p. P; \{\} \models \text{lower } p \implies A \text{ o lift} \vdash \text{lower } p \rangle$
shows $\langle A; B \vdash! p \rangle$
using *assms strong-completeness_{P'}*[**where** $P = P$ and $G = \{\}$] *by simp*

corollary *completeness_P*:

assumes $\langle P; \{\} \models! p \rangle$
and $\langle \forall r \in \text{anns } p. B r \rangle$
and $\langle \bigwedge p. P; \{\} \models p \implies A \text{ o lift} \vdash p \rangle$
shows $\langle A; B \vdash! p \rangle$
using *completeness_{P'}* *assms* .

corollary *completeness_{PA}*:

assumes $\langle (\lambda-. \text{True}); \{\} \models! p \rangle$
shows $\langle A; (\lambda-. \text{True}) \vdash! p \rangle$
using *assms completeness_P*[of $\langle \lambda-. \text{True} \rangle p \langle \lambda-. \text{True} \rangle$] *completeness_A* **by blast**

9 System PAL + K

abbreviation *SystemPK* $(\langle \vdash \vdash!_K \rightarrow [50, 50] \ 50 \rangle \text{ where}$

$\langle G \vdash!_K p \equiv (\lambda-. \text{False}); (\lambda-. \text{True}); G \vdash! p \rangle$

lemma *strong-soundness_{PK}*: $\langle G \vdash!_K p \implies (\lambda-. \text{True}); G \models!_\star p \rangle$

using *strong-soundness_P*[of $\langle \lambda-. \text{False} \rangle \langle \lambda-. \text{True} \rangle$] **by fast**

abbreviation *validPK* $(\langle \vdash \models!_K \rightarrow [50, 50] \ 50 \rangle \text{ where}$

$\langle G \models!_K p \equiv (\lambda-. \text{True}); G \models! p \rangle$

lemma *strong-completeness_{PK}*:

assumes $\langle G \models!_K p \rangle$

shows $\langle G \vdash!_K p \rangle$

using *strong-completeness_{PB}* *assms strong-completeness_K* **unfolding comp-apply**

.

theorem main_{PK} : $\langle G \models_K p \longleftrightarrow G \vdash_K p \rangle$
using $\text{strong-soundness}_{PK}[\text{of } G \ p]$ $\text{strong-completeness}_{PK}[\text{of } G \ p]$ **by** *fast*

corollary $\langle G \models_K p \implies (\lambda-. \text{True}); G \models_{!}\star p \rangle$
using $\text{strong-soundness}_{PK}[\text{of } G \ p]$ $\text{strong-completeness}_{PK}[\text{of } G \ p]$ **by** *fast*

10 System PAL + T

Also known as System PAL + M

inductive AxPT :: $\langle 'i \text{ pfm} \Rightarrow \text{bool} \rangle$ **where**
 $\langle \text{AxPT} \ (K! \ i \ p \longrightarrow! \ p) \rangle$

abbreviation SystemPT $(\langle - \vdash_T - \rangle [50, 50] \ 50)$ **where**
 $\langle G \vdash_T p \equiv \text{AxPT}; (\lambda-. \text{True}); G \vdash! \ p \rangle$

lemma soundness-AxPT : $\langle \text{AxPT} \ p \implies \text{reflexive } M \implies w \in \mathcal{W} \ M \implies M, w \models! \ p \rangle$
unfolding reflexive-def **by** $(\text{induct } p \text{ rule: } \text{AxPT.induct}) \text{ simp}$

lemma $\text{reflexive-restrict}$: $\langle \text{reflexive } M \implies \text{reflexive } (M[r!]) \rangle$
unfolding reflexive-def **by** *simp*

lemma $\text{strong-soundness}_{PT}$: $\langle G \vdash_T p \implies \text{reflexive}; G \models_{!}\star p \rangle$
using $\text{strong-soundness}_P[\text{of } \text{AxPT} \ \text{reflexive} \ \langle \lambda-. \text{True} \rangle \ G \ p]$
 $\text{soundness-AxPT} \ \text{reflexive-restrict}$ **by** *fast*

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

abbreviation validPT $(\langle - \models_T - \rangle [50, 50] \ 50)$ **where**
 $\langle G \models_T p \equiv \text{reflexive}; G \models_{!}\star p \rangle$

lemma $\text{strong-completeness}_{PT}$:
assumes $\langle G \models_T p \rangle$
shows $\langle G \vdash_T p \rangle$
using $\text{strong-completeness}_{PB} \ \text{assms} \ \text{strong-completeness}_T$ **unfolding** AxT-AxPT
.

theorem main_{PT} : $\langle G \models_T p \longleftrightarrow G \vdash_T p \rangle$
using $\text{strong-soundness}_{PT}[\text{of } G \ p]$ $\text{strong-completeness}_{PT}[\text{of } G \ p]$ **by** *fast*

corollary $\langle G \models_T p \implies \text{reflexive}; G \models_{!}\star p \rangle$
using $\text{strong-soundness}_{PT}[\text{of } G \ p]$ $\text{strong-completeness}_{PT}[\text{of } G \ p]$ **by** *fast*

11 System PAL + KB

inductive $AxPB :: \langle 'i \text{ pfm} \Rightarrow \text{bool} \rangle$ **where**

$\langle AxPB (p \longrightarrow! K! i (L! i p)) \rangle$

abbreviation $SystemPKB (\langle - \vdash!_{KB} - \rangle [50, 50] 50)$ **where**

$\langle G \vdash!_{KB} p \equiv AxPB; (\lambda -. \text{True}); G \vdash! p \rangle$

lemma *soundness-AxPB*: $\langle AxPB p \Longrightarrow \text{symmetric } M \Longrightarrow w \in \mathcal{W} \ M \Longrightarrow M, w \models! p \rangle$

unfolding *symmetric-def* **by** (*induct p rule: AxPB.induct*) *auto*

lemma *symmetric-restrict*: $\langle \text{symmetric } M \Longrightarrow \text{symmetric } (M[r!]) \rangle$

unfolding *symmetric-def* **by** *simp*

lemma *strong-soundness_{PKB}*: $\langle G \vdash!_{KB} p \Longrightarrow \text{symmetric}; G \models!_{\star} p \rangle$

using *strong-soundness_P*[*of AxPB symmetric* $\langle \lambda -. \text{True} \rangle G p$]

soundness-AxPB symmetric-restrict **by** *fast*

lemma *AxB-AxPB*: $\langle AxB = AxPB \circ \text{lift} \rangle$

proof

fix $p :: \langle 'i \text{ fm} \rangle$

show $\langle AxB p = (AxPB \circ \text{lift}) p \rangle$

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 $validPKB (\langle - \models!_{KB} - \rangle [50, 50] 50)$ **where**

$\langle G \models!_{KB} p \equiv \text{symmetric}; G \models! p \rangle$

lemma *strong-completeness_{PKB}*:

assumes $\langle G \models!_{KB} p \rangle$

shows $\langle G \vdash!_{KB} p \rangle$

using *strong-completeness_{PB} assms strong-completeness_{KB}* **unfolding** *AxB-AxPB*

.

theorem *main_{PKB}*: $\langle G \models!_{KB} p \longleftrightarrow G \vdash!_{KB} p \rangle$

using *strong-soundness_{PKB}*[*of G p*] *strong-completeness_{PKB}*[*of G p*] **by** *fast*

corollary $\langle G \models!_{KB} p \Longrightarrow \text{symmetric}; G \models!_{\star} p \rangle$

using *strong-soundness_{PKB}*[*of G p*] *strong-completeness_{PKB}*[*of G p*] **by** *fast*

12 System PAL + K4

inductive $AxP4 :: \langle 'i \text{ pfm} \Rightarrow \text{bool} \rangle$ **where**

$\langle AxP4 (K! i p \longrightarrow! K! i (K! i p)) \rangle$

abbreviation $SystemPK4 (\langle - \vdash!_{K4} - \rangle [50, 50] 50)$ **where**

$\langle G \vdash!_{K4} p \equiv AxP4; (\lambda -. \text{True}); G \vdash! p \rangle$

lemma *pos-introspection*:

assumes $\langle \text{transitive } M \rangle \langle w \in \mathcal{W} \ M \rangle$
shows $\langle M, w \models_! (K! \ i \ p \longrightarrow_! K! \ i \ (K! \ i \ p)) \rangle$
proof –
 { **assume** $\langle M, w \models_! K! \ i \ p \rangle$
 then have $\langle \forall v \in \mathcal{W} \ M \cap \mathcal{K} \ M \ i \ w. \ M, v \models_! p \rangle$
 by *simp*
 then have $\langle \forall v \in \mathcal{W} \ M \cap \mathcal{K} \ M \ i \ w. \ \forall u \in \mathcal{W} \ M \cap \mathcal{K} \ M \ i \ v. \ M, u \models_! p \rangle$
 using $\langle \text{transitive } M \rangle \langle w \in \mathcal{W} \ M \rangle$ **unfolding** *transitive-def* **by** *blast*
 then have $\langle \forall v \in \mathcal{W} \ M \cap \mathcal{K} \ M \ i \ w. \ M, v \models_! K! \ i \ p \rangle$
 by *simp*
 then have $\langle M, w \models_! K! \ i \ (K! \ i \ p) \rangle$
 by *simp* }
then show *?thesis*
 by *fastforce*
qed

lemma *soundness-AxP4*: $\langle \text{AxP4 } p \implies \text{transitive } M \implies w \in \mathcal{W} \ M \implies M, w \models_! p \rangle$

by (*induct p rule: AxP4.induct*) (*metis pos-introspection*)

lemma *transitive-restrict*: $\langle \text{transitive } M \implies \text{transitive } (M[r!]) \rangle$

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}*: $\langle G \vdash_{!K4} p \implies \text{transitive}; G \models_{! \star} p \rangle$

using *strong-soundness_P* [*of AxP4 transitive* $\langle \lambda-. \text{True} \rangle G \ p$]
soundness-AxP4 transitive-restrict **by** *fast*

lemma *Ax4-AxP4*: $\langle \text{Ax4} = \text{AxP4} \circ \text{lift} \rangle$

proof

fix $p :: \langle 'i \ fm \rangle$

show $\langle \text{Ax4 } p = (\text{AxP4} \circ \text{lift}) \ p \rangle$

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* ($\langle - \models_{!K4} - \rangle [50, 50] \ 50$) **where**

$\langle G \models_{!K4} p \equiv \text{transitive}; G \models_! p \rangle$

lemma *strong-completeness_{PK4}*:

assumes $\langle G \models_{!K4} p \rangle$

shows $\langle G \vdash_{!K4} p \rangle$

using *strong-completeness_{PB}* *assms strong-completeness_{K4}* **unfolding** *Ax4-AxP4*

.

theorem *main_{PK4}*: $\langle G \models_{!K4} p \longleftrightarrow G \vdash_{!K4} p \rangle$

using *strong-soundness_{PK4}* [*of G p*] *strong-completeness_{PK4}* [*of G p*] **by** *fast*

corollary $\langle G \models_{!K4} p \implies \text{transitive}; G \models_{! \star} p \rangle$
using *strong-soundness_{PK4}[of G p] strong-completeness_{PK4}[of G p]* **by** *fast*

13 System PAL + K5

inductive *AxP5* :: $\langle 'i \text{ pfm} \Rightarrow \text{bool} \rangle$ **where**
 $\langle \text{AxP5 } (L \vdash i \text{ p} \longrightarrow K \vdash i (L \vdash i \text{ p})) \rangle$

abbreviation *SystemPK5* ($\langle \vdash_{!K5} \rightarrow [50, 50] \ 50 \rangle$) **where**
 $\langle G \vdash_{!K5} p \equiv \text{AxP5}; (\lambda \cdot \text{True}); G \vdash \text{p} \rangle$

lemma *soundness-AxP5*: $\langle \text{AxP5 } p \implies \text{Euclidean } M \implies w \in \mathcal{W} \ M \implies M, w \models_{!} p \rangle$
by (*induct p rule: AxP5.induct*) (*unfold Euclidean-def psemantics.simps, blast*)

lemma *Euclidean-restrict*: $\langle \text{Euclidean } M \implies \text{Euclidean } (M[r!]) \rangle$
unfolding *Euclidean-def* **by** *auto*

lemma *strong-soundness_{PK5}*: $\langle G \vdash_{!K5} p \implies \text{Euclidean}; G \models_{! \star} p \rangle$
using *strong-soundness_P[of AxP5 Euclidean $\langle \lambda \cdot \text{True} \rangle G p]$*
soundness-AxP5 Euclidean-restrict **by** *fast*

lemma *Ax5-AxP5*: $\langle \text{Ax5} = \text{AxP5} \circ \text{lift} \rangle$
proof
fix *p* :: $\langle 'i \text{ fm} \rangle$
show $\langle \text{Ax5 } p = (\text{AxP5} \circ \text{lift}) \text{ p} \rangle$
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* ($\langle \vdash_{!K5} \rightarrow [50, 50] \ 50 \rangle$) **where**
 $\langle G \models_{!K5} p \equiv \text{Euclidean}; G \models_{!} p \rangle$

lemma *strong-completeness_{PK5}*:
assumes $\langle G \models_{!K5} p \rangle$
shows $\langle G \vdash_{!K5} p \rangle$
using *strong-completeness_{PB} assms strong-completeness_{K5}* **unfolding** *Ax5-AxP5*
.

theorem *main_{PK5}*: $\langle G \models_{!K5} p \longleftrightarrow G \vdash_{!K5} p \rangle$
using *strong-soundness_{PK5}[of G p] strong-completeness_{PK5}[of G p]* **by** *fast*

corollary $\langle G \models_{!K5} p \implies \text{Euclidean}; G \models_{! \star} p \rangle$
using *strong-soundness_{PK5}[of G p] strong-completeness_{PK5}[of G p]* **by** *fast*

14 System PAL + S4

abbreviation $SystemPS_4$ ($\langle \cdot \vdash_{!S_4} \cdot \rangle [50, 50]$ 50) **where**

$\langle G \vdash_{!S_4} p \equiv AxPT \oplus AxP_4; (\lambda \cdot. True); G \vdash_{!} p \rangle$

lemma $soundness-AxPT_4$: $\langle (AxPT \oplus AxP_4) p \implies refltrans M \implies w \in \mathcal{W} M \implies M, w \models_{!} p \rangle$

using $soundness-AxPT$ $soundness-AxP_4$ **by** $fast$

lemma $refltrans-restrict$: $\langle refltrans M \implies refltrans (M[r!]) \rangle$

using $reflexive-restrict$ $transitive-restrict$ **by** $blast$

lemma $strong-soundness_{PS_4}$: $\langle G \vdash_{!S_4} p \implies refltrans; G \models_{!}^* p \rangle$

using $strong-soundness_P$ [of $\langle AxPT \oplus AxP_4 \rangle refltrans \langle \lambda \cdot. True \rangle G p$]
 $soundness-AxPT_4$ $refltrans-restrict$ **by** $fast$

lemma AxT_4-AxPT_4 : $\langle (AxT \oplus Ax_4) = (AxPT \oplus AxP_4) \text{ o lift} \rangle$

using $AxT-AxPT$ Ax_4-AxP_4 **unfolding** $comp-apply$ **by** $metis$

abbreviation $validPS_4$ ($\langle \cdot \models_{!S_4} \cdot \rangle [50, 50]$ 50) **where**

$\langle G \models_{!S_4} p \equiv refltrans; G \models_{!} p \rangle$

theorem $strong-completeness_{PS_4}$:

assumes $\langle G \models_{!S_4} p \rangle$

shows $\langle G \vdash_{!S_4} p \rangle$

using $strong-completeness_{PB}$ $assms$ $strong-completeness_{S_4}$ **unfolding** AxT_4-AxPT_4

.

theorem $main_{PS_4}$: $\langle G \models_{!S_4} p \longleftrightarrow G \vdash_{!S_4} p \rangle$

using $strong-soundness_{PS_4}$ [of $G p$] $strong-completeness_{PS_4}$ [of $G p$] **by** $fast$

corollary $\langle G \models_{!S_4} p \implies refltrans; G \models_{!}^* p \rangle$

using $strong-soundness_{PS_4}$ [of $G p$] $strong-completeness_{PS_4}$ [of $G p$] **by** $fast$

15 System PAL + S5

abbreviation $SystemPS_5$ ($\langle \cdot \vdash_{!S_5} \cdot \rangle [50, 50]$ 50) **where**

$\langle G \vdash_{!S_5} p \equiv AxPT \oplus AxPB \oplus AxP_4; (\lambda \cdot. True); G \vdash_{!} p \rangle$

abbreviation $AxPTB_4$:: $\langle 'i \text{ pfm} \Rightarrow bool \rangle$ **where**

$\langle AxPTB_4 \equiv AxPT \oplus AxPB \oplus AxP_4 \rangle$

lemma $soundness-AxPTB_4$: $\langle AxPTB_4 p \implies equivalence M \implies w \in \mathcal{W} M \implies M, w \models_{!} p \rangle$

using $soundness-AxPT$ $soundness-AxPB$ $soundness-AxP_4$ **by** $fast$

lemma $equivalence-restrict$: $\langle equivalence M \implies equivalence (M[r!]) \rangle$

using $reflexive-restrict$ $symmetric-restrict$ $transitive-restrict$ **by** $blast$

lemma *strong-soundness_{PS5}*: $\langle G \vdash_{!S5} p \implies \text{equivalence}; G \Vdash_{! \star} p \rangle$
using *strong-soundness_P*[of *AxPTB4* equivalence $\langle \lambda -. \text{True} \rangle G p$]
soundness-AxPTB4 equivalence-restrict **by** *fast*

lemma *AxTB4-AxPTB4*: $\langle \text{AxTB4} = \text{AxPTB4} \text{ o lift} \rangle$
using *AxT-AxPT* *AxB-AxPB* *Ax4-AxP4* **unfolding** *comp-apply* **by** *metis*

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

theorem *strong-completeness_{PS5}*:
assumes $\langle G \Vdash_{!S5} p \rangle$
shows $\langle G \vdash_{!S5} p \rangle$
using *strong-completeness_{PB}* *assms* *strong-completeness_{S5}* **unfolding** *AxTB4-AxPTB4*
.

theorem *main_{PS5}*: $\langle G \Vdash_{!S5} p \longleftrightarrow G \vdash_{!S5} p \rangle$
using *strong-soundness_{PS5}*[of *G p*] *strong-completeness_{PS5}*[of *G p*] **by** *fast*

corollary $\langle G \Vdash_{!S5} p \implies \text{equivalence}; G \Vdash_{! \star} p \rangle$
using *strong-soundness_{PS5}*[of *G p*] *strong-completeness_{PS5}*[of *G p*] **by** *fast*

16 System PAL + S5'

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

abbreviation *AxPT5* :: $\langle 'i \text{ pfm} \Rightarrow \text{bool} \rangle$ **where**
 $\langle \text{AxPT5} \equiv \text{AxPT} \oplus \text{AxP5} \rangle$

lemma *soundness-AxPT5*: $\langle \text{AxPT5 } p \implies \text{equivalence } M \implies w \in \mathcal{W} \ M \implies M, w \Vdash_{!} p \rangle$
using *soundness-AxPT* *soundness-AxPT* *soundness-AxP5* *symm-trans-Euclid* **by** *fast*

lemma *strong-soundness_{PS5'}*: $\langle G \vdash_{!S5'} p \implies \text{equivalence}; G \Vdash_{! \star} p \rangle$
using *strong-soundness_P*[of *AxPT5* equivalence $\langle \lambda -. \text{True} \rangle G p$]
soundness-AxPT5 equivalence-restrict **by** *fast*

lemma *AxT5-AxPT5*: $\langle \text{AxT5} = \text{AxPT5} \text{ o lift} \rangle$
using *AxT-AxPT* *Ax5-AxP5* **unfolding** *comp-apply* **by** *metis*

theorem *strong-completeness_{PS5'}*:
assumes $\langle G \Vdash_{!S5'} p \rangle$
shows $\langle G \vdash_{!S5'} p \rangle$
using *strong-completeness_{PB}* *assms* *strong-completeness_{S5'}* **unfolding** *AxT5-AxPT5*
.

theorem *main_{PS5'}*: $\langle G \Vdash_{!S5'} p \longleftrightarrow G \vdash_{!S5'} p \rangle$

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.