



## Public Announcement Logic

From, Asta Halkjær

*Published in:*  
Archive of Formal Proofs

*Publication date:*  
2021

*Document Version*  
Publisher's PDF, also known as Version of record

[Link back to DTU Orbit](#)

*Citation (APA):*  
From, A. H. (2021). Public Announcement Logic. *Archive of Formal Proofs*, 2021.

---

### General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

# Public Announcement Logic

Asta Halkjær From

December 14, 2021

## Abstract

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

## Contents

<b>1</b>	<b>Syntax</b>	<b>2</b>
<b>2</b>	<b>Semantics</b>	<b>2</b>
<b>3</b>	<b>Soundness of Reduction</b>	<b>3</b>
<b>4</b>	<b>Proof System</b>	<b>4</b>
<b>5</b>	<b>Soundness</b>	<b>5</b>
<b>6</b>	<b>Completeness</b>	<b>6</b>

theory PAL imports Epistemic-Logic.Epistemic-Logic begin

## 1 Syntax

**datatype**  $\langle 'i \text{ pfm} \rangle$   
 $= FF \ (\perp_!)$   
 $| \text{Pro}' \text{id} \ (\text{Pro}_!)$   
 $| \text{Dis} \ \langle 'i \text{ pfm} \rangle \ \langle 'i \text{ pfm} \rangle \ (\mathbf{infixr} \ \vee_! \ 30)$   
 $| \text{Con} \ \langle 'i \text{ pfm} \rangle \ \langle 'i \text{ pfm} \rangle \ (\mathbf{infixr} \ \wedge_! \ 35)$   
 $| \text{Imp} \ \langle 'i \text{ pfm} \rangle \ \langle 'i \text{ pfm} \rangle \ (\mathbf{infixr} \ \longrightarrow_! \ 25)$   
 $| \text{K}' \ 'i \ \langle 'i \text{ pfm} \rangle \ (\text{K}_!)$   
 $| \text{Ann} \ \langle 'i \text{ pfm} \rangle \ \langle 'i \text{ pfm} \rangle \ ([-]_! \ - \ [50, 50] \ 50)$

**abbreviation**  $\text{PIff} :: \langle 'i \text{ pfm} \Rightarrow 'i \text{ pfm} \Rightarrow 'i \text{ pfm} \rangle \ (\mathbf{infixr} \ \longleftrightarrow_! \ 25)$  **where**  
 $\langle p \longleftrightarrow_! q \equiv (p \longrightarrow_! q) \wedge_! (q \longrightarrow_! p) \rangle$

## 2 Semantics

**fun**

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

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

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

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

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

|  $\langle \text{lower } ([r]_! p) = \text{undefined} \rangle$

**primrec lift** ::  $\langle 'i \text{ fm} \Rightarrow 'i \text{ pfm} \rangle$  **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** (*induct p arbitrary: w*) *simp-all*

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

**by** (*induct p arbitrary: w*) *simp-all*

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

**by** (*induct p*) *simp-all*

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

**by** (*induct p*) *simp-all*

### 3 Soundness of Reduction

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

$\langle \text{reduce}' \ r \ \perp_! = (r \longrightarrow_! \ \perp_!) \rangle$   
|  $\langle \text{reduce}' \ r \ (\text{Pro}_! \ x) = (r \longrightarrow_! \ \text{Pro}_! \ x) \rangle$   
|  $\langle \text{reduce}' \ r \ (p \vee_! \ q) = (\text{reduce}' \ r \ p \vee_! \ \text{reduce}' \ r \ q) \rangle$   
|  $\langle \text{reduce}' \ r \ (p \wedge_! \ q) = (\text{reduce}' \ r \ p \wedge_! \ \text{reduce}' \ r \ q) \rangle$   
|  $\langle \text{reduce}' \ r \ (p \longrightarrow_! \ q) = (\text{reduce}' \ r \ p \longrightarrow_! \ \text{reduce}' \ r \ q) \rangle$   
|  $\langle \text{reduce}' \ r \ (K_! \ i \ p) = (r \longrightarrow_! \ K_! \ i \ (\text{reduce}' \ r \ p)) \rangle$   
|  $\langle \text{reduce}' \ r \ ([p]_! \ q) = \text{undefined} \rangle$

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

$\langle \text{reduce } \perp_! = \perp_! \rangle$   
|  $\langle \text{reduce } (\text{Pro}_! \ x) = \text{Pro}_! \ x \rangle$   
|  $\langle \text{reduce } (p \vee_! \ q) = (\text{reduce } p \vee_! \ \text{reduce } q) \rangle$   
|  $\langle \text{reduce } (p \wedge_! \ q) = (\text{reduce } p \wedge_! \ \text{reduce } q) \rangle$   
|  $\langle \text{reduce } (p \longrightarrow_! \ q) = (\text{reduce } p \longrightarrow_! \ \text{reduce } q) \rangle$   
|  $\langle \text{reduce } (K_! \ i \ p) = K_! \ i \ (\text{reduce } p) \rangle$   
|  $\langle \text{reduce } ([r]_! \ p) = \text{reduce}' \ (\text{reduce } r) \ (\text{reduce } p) \rangle$

**lemma static-reduce':**  $\langle \text{static } p \Longrightarrow \text{static } r \Longrightarrow \text{static } (\text{reduce}' \ r \ p) \rangle$

**by** (*induct p*) *simp-all*

**lemma static-reduce:**  $\langle \text{static } (\text{reduce } p) \rangle$

**by** (*induct p*) (*simp-all add: static-reduce'*)

**lemma** *reduce'-semantics*:  
**assumes**  $\langle \text{static } q \rangle$   
**shows**  $\langle (M, w \models [p]_! (q)) = (M, w \models \text{reduce}' p q) \rangle$   
**using** *assms* **by** (*induct q arbitrary: w*) *auto*

**lemma** *reduce-semantics*:  $\langle (M, w \models p) = (M, w \models \text{reduce } p) \rangle$   
**proof** (*induct p arbitrary: M w*)  
**case** (*Ann p q*)  
**then show** *?case*  
**using** *reduce'-semantics static-reduce* **by** *fastforce*  
**qed** *simp-all*

## 4 Proof System

**primrec** *peval* ::  $\langle (id \Rightarrow bool) \Rightarrow ('i \text{ pfm} \Rightarrow bool) \Rightarrow 'i \text{ pfm} \Rightarrow bool \rangle$  **where**  
 $\langle \text{peval } - \text{ } \perp_! = \text{False} \rangle$   
 $\langle \text{peval } g \text{ } (Pro_! x) = g x \rangle$   
 $\langle \text{peval } g \text{ } h (p \vee_! q) = (\text{peval } g \text{ } h p \vee \text{peval } g \text{ } h q) \rangle$   
 $\langle \text{peval } g \text{ } h (p \wedge_! q) = (\text{peval } g \text{ } h p \wedge \text{peval } g \text{ } h q) \rangle$   
 $\langle \text{peval } g \text{ } h (p \longrightarrow_! q) = (\text{peval } g \text{ } h p \longrightarrow \text{peval } g \text{ } h q) \rangle$   
 $\langle \text{peval } - \text{ } h (K_! i p) = h (K_! i p) \rangle$   
 $\langle \text{peval } - \text{ } h ([r]_! p) = h ([r]_! p) \rangle$

**abbreviation**  $\langle \text{ptautology } p \equiv \forall g h. \text{peval } g \text{ } h p \rangle$

**inductive** *PAK* ::  $\langle ('i \text{ pfm} \Rightarrow bool) \Rightarrow 'i \text{ pfm} \Rightarrow bool \rangle$  ( $- \vdash_! - [50, 50] 50$ )  
**for**  $A :: \langle 'i \text{ pfm} \Rightarrow bool \rangle$  **where**  
 $\langle \text{PA1: } \langle \text{ptautology } p \Longrightarrow A \vdash_! p \rangle \rangle$   
 $\langle \text{PA2: } \langle A \vdash_! (K_! i p \wedge_! K_! i (p \longrightarrow_! q)) \longrightarrow_! K_! i q \rangle \rangle$   
 $\langle \text{PAx: } \langle A p \Longrightarrow A \vdash_! p \rangle \rangle$   
 $\langle \text{PR1: } \langle A \vdash_! p \Longrightarrow A \vdash_! (p \longrightarrow_! q) \Longrightarrow A \vdash_! q \rangle \rangle$   
 $\langle \text{PR2: } \langle A \vdash_! p \Longrightarrow A \vdash_! K_! i p \rangle \rangle$   
 $\langle \text{PFF: } \langle A \vdash_! ([r]_! \perp_! \longleftarrow_! (r \longrightarrow_! \perp_!)) \rangle \rangle$   
 $\langle \text{PPro: } \langle A \vdash_! ([r]_! Pro_! x \longleftarrow_! (r \longrightarrow_! Pro_! x)) \rangle \rangle$   
 $\langle \text{PDis: } \langle A \vdash_! ([r]_! (p \vee_! q) \longleftarrow_! [r]_! p \vee_! [r]_! q) \rangle \rangle$   
 $\langle \text{PCon: } \langle A \vdash_! ([r]_! (p \wedge_! q) \longleftarrow_! [r]_! p \wedge_! [r]_! q) \rangle \rangle$   
 $\langle \text{PImp: } \langle A \vdash_! (([r]_! (p \longrightarrow_! q)) \longleftarrow_! ([r]_! p \longrightarrow_! [r]_! q)) \rangle \rangle$   
 $\langle \text{PK: } \langle A \vdash_! (([r]_! K_! i p) \longleftarrow_! (r \longrightarrow_! K_! i ([r]_! p))) \rangle \rangle$   
 $\langle \text{PAnn: } \langle A \vdash_! p \Longrightarrow A \vdash_! [r]_! p \rangle \rangle$

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

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

**lemma** *peval-eval*:  
**assumes**  $\langle \text{static } p \rangle$

**shows**  $\langle \text{eval } h \ g \ (\text{lower } p) = \text{peval } h \ (g \ o \ \text{lower}) \ p \rangle$   
**using** *assms* **by**  $(\text{induct } p) \ \text{simp-all}$

**lemma** *ptautology-tautology*:

**assumes**  $\langle \text{static } p \rangle$   
**shows**  $\langle \text{ptautology } p \implies \text{tautology } (\text{lower } p) \rangle$   
**using** *assms* *peval-eval* **by** *blast*

**theorem** *AK-PAK*:  $\langle A \ o \ \text{lift} \vdash p \implies A \vdash_{\dagger} \text{lift } p \rangle$

**by**  $(\text{induct } p \ \text{rule: } \text{AK.induct}) \ (\text{auto } \text{intro: } \text{PAK.intros}(1-5) \ \text{simp: } \text{tautology-ptautology})$

**theorem** *static-completeness*:

**assumes**  $\langle \text{static } p \rangle \ \langle \forall (M :: ('i :: \text{countable}, 'i \ \text{fm set}) \ \text{kripke}) \ w. \ M, \ w \models_{\dagger} p \rangle$   
**shows**  $\langle A \vdash_{\dagger} p \rangle$

**proof** –

**have**  $\langle \forall (M :: ('i :: \text{countable}, 'i \ \text{fm set}) \ \text{kripke}) \ w. \ M, \ w \models \text{lower } p \rangle$

**using** *assms* **by**  $(\text{simp } \text{add: } \text{lower-semantic})$

**then have**  $\langle A \ o \ \text{lift} \vdash \text{lower } p \rangle$

**by**  $(\text{simp } \text{add: } \text{completeness})$

**then have**  $\langle A \vdash_{\dagger} \text{lift } (\text{lower } p) \rangle$

**using** *AK-PAK* **by** *fast*

**then show** *?thesis*

**using** *assms*(1) *lift-lower* **by** *metis*

**qed**

## 5 Soundness

**lemma** *peval-semantic*:  $\langle \text{peval } (\text{val } w) \ (\lambda q. \ \text{Kripke } W \ \text{val } r, \ w \models_{\dagger} q) \ p = (\text{Kripke } W \ \text{val } r, \ w \models_{\dagger} p) \rangle$

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

**lemma** *ptautology*:

**assumes**  $\langle \text{ptautology } p \rangle$

**shows**  $\langle M, \ w \models_{\dagger} p \rangle$

**proof** –

**from** *assms* **have**  $\langle \text{peval } (g \ w) \ (\lambda q. \ \text{Kripke } W \ g \ r, \ w \models_{\dagger} q) \ p \rangle$  **for**  $W \ g \ r$

**by** *simp*

**then have**  $\langle \text{Kripke } W \ g \ r, \ w \models_{\dagger} p \rangle$  **for**  $W \ g \ r$

**using** *peval-semantic* **by** *fast*

**then show**  $\langle M, \ w \models_{\dagger} p \rangle$

**by**  $(\text{metis } \text{kripke.collapse})$

**qed**

**theorem** *soundness*:

**fixes**  $M :: \langle ('i, 'w) \ \text{kripke} \rangle$

**assumes**

$\langle \bigwedge (M :: ('i, 'w) \ \text{kripke}) \ w \ p. \ A \ p \implies P \ M \implies M, \ w \models_{\dagger} p \rangle$

$\langle \bigwedge (M :: ('i, 'w) \ \text{kripke}) \ p. \ P \ M \implies P \ (\text{restrict } M \ p) \rangle$

**shows**  $\langle A \vdash_{\dagger} p \implies P \ M \implies M, \ w \models_{\dagger} p \rangle$

**proof** (*induct p arbitrary: M w rule: PAK.induct*)  
**case** (*PAnn p r*)  
**moreover have**  $\langle P \text{ (restrict } M \text{ } r) \rangle$   
**using** *PAnn(3) assms(2)* **by** *simp*  
**ultimately show** *?case*  
**by** *simp*  
**qed** (*simp-all add: assms ptautology*)

**corollary**  $\langle \lambda-. \text{ False} \rangle \vdash_! p \implies M, w \models_! p$   
**using** *soundness[where P= $\langle \lambda-. \text{ True} \rangle$ ]* **by** *metis*

## 6 Completeness

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

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

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

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

**lemma** *Iff-sym*:  $\langle A \vdash_! (p \longleftrightarrow_! q) \rangle = \langle A \vdash_! (q \longleftrightarrow_! p) \rangle$

**proof** –

**have**  $\langle A \vdash_! ((p \longleftrightarrow_! q) \longleftrightarrow_! (q \longleftrightarrow_! p)) \rangle$

**by** (*simp add: PA1*)

**then show** *?thesis*

**using** *PR1 ConE* **by** *blast*

**qed**

**lemma** *Iff-Iff*:

**assumes**  $\langle A \vdash_! (p \longleftrightarrow_! p') \rangle \langle A \vdash_! (p \longleftrightarrow_! q) \rangle$

**shows**  $\langle A \vdash_! (p' \longleftrightarrow_! q) \rangle$

**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**  $\langle A \vdash_! ((p \longleftrightarrow_! p') \longrightarrow_! (p \longleftrightarrow_! q) \longrightarrow_! (p' \longleftrightarrow_! q)) \rangle$  .

**then show** *?thesis*

**using** *assms PR1* **by** *blast*

**qed**

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

**proof** –

**have**  $\langle A \vdash_! (K_! i p \wedge_! K_! i (p \longrightarrow_! q) \longrightarrow_! K_! i q) \rangle$

**using** *PA2* **by** *fast*

**moreover have**  $\langle A \vdash_! ((P \wedge_! Q \longrightarrow_! R) \longrightarrow_! (Q \longrightarrow_! P \longrightarrow_! R)) \rangle$  **for** *P Q R*

**by** (*simp add: PA1*)

**ultimately show** *?thesis*

**using** *PR1* **by** *fast*

**qed**

**lemma** *K'-map*:

**assumes**  $\langle A \vdash_! (p \longrightarrow_! q) \rangle$

**shows**  $\langle A \vdash_! (K_! i p \longrightarrow_! K_! i q) \rangle$

**proof** –

**note**  $\langle A \vdash_! (p \longrightarrow_! q) \rangle$

**then have**  $\langle A \vdash_! K_! i (p \longrightarrow_! q) \rangle$

**using** *PR2* **by** *fast*

**moreover have**  $\langle A \vdash_! (K_! i (p \longrightarrow_! q) \longrightarrow_! K_! i p \longrightarrow_! K_! i q) \rangle$

**using** *K'-A2'* **by** *fast*

**ultimately show** *?thesis*

**using** *PR1* **by** *fast*

**qed**

**lemma** *ConI*:

**assumes**  $\langle A \vdash_! p \rangle \langle A \vdash_! q \rangle$

**shows**  $\langle A \vdash_! (p \wedge_! q) \rangle$

**proof** –

**have**  $\langle A \vdash_! (p \longrightarrow_! q \longrightarrow_! p \wedge_! q) \rangle$

**by** (*simp add: PA1*)

**then show** *?thesis*

**using** *assms PR1* **by** *blast*



qed

lemma *Iff-wk*:

assumes  $\langle A \vdash! (p \longleftrightarrow! q) \rangle$

shows  $\langle A \vdash! ((r \longrightarrow! p) \longleftrightarrow! (r \longrightarrow! q)) \rangle$

proof –

have  $\langle A \vdash! ((p \longleftrightarrow! q) \longrightarrow! ((r \longrightarrow! p) \longleftrightarrow! (r \longrightarrow! q))) \rangle$

by (*simp add: PA1*)

then show *?thesis*

using *assms PR1* by *blast*

qed

lemma *Iff-reduce'*:

assumes  $\langle \text{static } p \rangle$

shows  $\langle A \vdash! ([r]! p \longleftrightarrow! \text{reduce}' r p) \rangle$

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  $\langle A \vdash! ([r]! p \vee! [r]! q \longleftrightarrow! \text{reduce}' r (p \vee! q)) \rangle$

using *Iff-Dis* by *force*

moreover have  $\langle A \vdash! (([r]! p \vee! [r]! q) \longleftrightarrow! ([r]! (p \vee! q))) \rangle$

using *PDis Iff-sym* by *fastforce*

ultimately show *?case*

using *PA1 PR1 Iff-Iff* by *blast*

next

case (*Con p q*)

then have  $\langle A \vdash! ([r]! p \wedge! [r]! q \longleftrightarrow! \text{reduce}' r (p \wedge! q)) \rangle$

using *Iff-Con* by *force*

moreover have  $\langle A \vdash! (([r]! p \wedge! [r]! q) \longleftrightarrow! ([r]! (p \wedge! q))) \rangle$

using *PCon Iff-sym* by *fastforce*

ultimately show *?case*

using *PA1 PR1 Iff-Iff* by *blast*

next

case (*Imp p q*)

then have  $\langle A \vdash! (([r]! p \longrightarrow! [r]! q) \longleftrightarrow! \text{reduce}' r (p \longrightarrow! q)) \rangle$

using *Iff-Imp* by *force*

moreover have  $\langle A \vdash! (([r]! p \longrightarrow! [r]! q) \longleftrightarrow! ([r]! (p \longrightarrow! q))) \rangle$

using *PImp Iff-sym* by *fastforce*

ultimately show *?case*

using *PA1 PR1 Iff-Iff* by *blast*

next

**case**  $\langle K' i p \rangle$   
**then have**  $\langle A \vdash ([r]_! p \longleftrightarrow_! \text{reduce}' r p) \rangle$   
   **by** *simp*  
**then have**  $\langle A \vdash (K_! i ([r]_! p) \longleftrightarrow_! K_! i (\text{reduce}' r p)) \rangle$   
   **using** *K'-map ConE ConI* **by** *metis*  
**moreover have**  $\langle A \vdash ([r]_! K_! i p \longleftrightarrow_! r \longrightarrow_! K_! i ([r]_! p)) \rangle$   
   **using** *PK* .  
**ultimately have**  $\langle A \vdash ([r]_! K_! i p \longleftrightarrow_! r \longrightarrow_! K_! i (\text{reduce}' r p)) \rangle$   
   **by** (*meson Iff-Iff Iff-sym Iff-wk*)  
**then show** *?case*  
   **by** *simp*  
**next**  
   **case**  $\langle \text{Ann } r p \rangle$   
   **then show** *?case*  
     **by** *simp*  
**qed**

**lemma** *Iff-Ann1*:

**assumes**  $r$ :  $\langle A \vdash (r \longleftrightarrow_! r') \rangle$  **and**  $\langle \text{static } p \rangle$   
**shows**  $\langle A \vdash ([r]_! p \longleftrightarrow_! [r']_! p) \rangle$   
**using** *assms(2-)*  
**proof** (*induct p*)  
   **case** *FF*  
   **have**  $\langle A \vdash ((r \longleftrightarrow_! r') \longrightarrow_! ((r \longrightarrow_! \perp_!) \longleftrightarrow_! (r' \longrightarrow_! \perp_!))) \rangle$   
     **by** (*auto intro: PA1*)  
   **then have**  $\langle A \vdash ((r \longrightarrow_! \perp_!) \longleftrightarrow_! (r' \longrightarrow_! \perp_!)) \rangle$   
     **using** *r PR1* **by** *blast*  
   **then show** *?case*  
     **by** (*meson PFF Iff-Iff Iff-sym*)  
**next**  
   **case**  $\langle \text{Pro}' x \rangle$   
   **have**  $\langle A \vdash ((r \longleftrightarrow_! r') \longrightarrow_! ((r \longrightarrow_! \text{Pro}_! x) \longleftrightarrow_! (r' \longrightarrow_! \text{Pro}_! x))) \rangle$   
     **by** (*auto intro: PA1*)  
   **then have**  $\langle A \vdash ((r \longrightarrow_! \text{Pro}_! x) \longleftrightarrow_! (r' \longrightarrow_! \text{Pro}_! x)) \rangle$   
     **using** *r PR1* **by** *blast*  
   **then show** *?case*  
     **by** (*meson PPro Iff-Iff Iff-sym*)  
**next**  
   **case**  $\langle \text{Dis } p q \rangle$   
   **then have**  $\langle A \vdash ([r]_! p \vee_! [r]_! q \longleftrightarrow_! [r']_! p \vee_! [r']_! q) \rangle$   
     **by** (*simp add: Iff-Dis*)  
   **then show** *?case*  
     **by** (*meson PDis Iff-Iff Iff-sym*)  
**next**  
   **case**  $\langle \text{Con } p q \rangle$   
   **then have**  $\langle A \vdash ([r]_! p \wedge_! [r]_! q \longleftrightarrow_! [r']_! p \wedge_! [r']_! q) \rangle$   
     **by** (*simp add: Iff-Con*)  
   **then show** *?case*  
     **by** (*meson PCon Iff-Iff Iff-sym*)

```

next
  case (Imp p q)
  then have  $\langle A \vdash_! ([r]_! p \longrightarrow_! [r]_! q) \longleftrightarrow_! ([r']_! p \longrightarrow_! [r']_! q) \rangle$ 
    by (simp add: Iff-Imp)
  then show ?case
    by (meson PImp Iff-Iff Iff-sym)
next
  case (K' i p)
  then have  $\langle A \vdash_! ([r]_! p \longleftrightarrow_! [r']_! p) \rangle$ 
    by simp
  then have  $\langle A \vdash_! (K_! i ([r]_! p) \longleftrightarrow_! K_! i ([r']_! p)) \rangle$ 
    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  $\langle A \vdash_! (p \longleftrightarrow_! p') \rangle$ 
  shows  $\langle A \vdash_! ([r]_! p \longleftrightarrow_! [r]_! p') \rangle$ 
  using assms PAnn ConE ConI PImp PR1 by metis

```

```

lemma Iff-reduce:  $\langle A \vdash_! (p \longleftrightarrow_! \text{reduce } p) \rangle$ 

```

```

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)
  have
     $\langle A \vdash_! (K_! i p \longrightarrow_! K_! i (\text{reduce } p)) \rangle$ 
     $\langle A \vdash_! (K_! i (\text{reduce } p) \longrightarrow_! K_! i p) \rangle$ 
    using K' K'-map ConE by fast+
  then have  $\langle A \vdash_! (K_! i p \longleftrightarrow_! K_! i (\text{reduce } p)) \rangle$ 
    using ConI by blast
  then show ?case
    by simp
next

```

**case** (*Ann r p*)  
**have**  $\langle A \vdash_! ([\text{reduce } r]_!, \text{reduce } p \longleftrightarrow_! \text{reduce}' (\text{reduce } r) (\text{reduce } p)) \rangle$   
**using** *Iff-reduce' static-reduce* **by** *blast*  
**moreover have**  $\langle A \vdash_! ([r]_!, \text{reduce } p \longleftrightarrow_! [\text{reduce } r]_!, \text{reduce } p) \rangle$   
**using** *Ann(1) Iff-Ann1 static-reduce* **by** *blast*  
**ultimately have**  $\langle A \vdash_! ([r]_!, \text{reduce } p \longleftrightarrow_! \text{reduce}' (\text{reduce } r) (\text{reduce } p)) \rangle$   
**using** *Iff-Iff Iff-sym* **by** *blast*  
**moreover have**  $\langle A \vdash_! ([r]_!, \text{reduce } p \longleftrightarrow_! [r]_!, p) \rangle$   
**by** (*meson Ann(2) static-reduce Iff-Ann2 Iff-sym*)  
**ultimately have**  $\langle A \vdash_! ([r]_!, p \longleftrightarrow_! \text{reduce}' (\text{reduce } r) (\text{reduce } p)) \rangle$   
**using** *Iff-Iff* **by** *blast*  
**then show** *?case*  
**by** *simp*  
**qed** (*simp-all add: PA1*)

**theorem** *completeness*:

**assumes**  $\langle \forall (M :: ('i :: \text{countable}, 'i \text{ fm set}) \text{kripke}) w. M, w \models_! p \rangle$   
**shows**  $\langle A \vdash_! p \rangle$

**proof** –

**have**  $\langle \forall (M :: ('i :: \text{countable}, 'i \text{ fm set}) \text{kripke}) w. M, w \models_! \text{reduce } p \rangle$   
**using** *assms reduce-semantics* **by** *fast*  
**moreover have**  $\langle \text{static } (\text{reduce } p) \rangle$   
**using** *static-reduce* **by** *fast*  
**ultimately have**  $\langle A \vdash_! \text{reduce } p \rangle$   
**using** *static-completeness* **by** *blast*  
**moreover have**  $\langle A \vdash_! (p \longleftrightarrow_! \text{reduce } p) \rangle$   
**using** *Iff-reduce* **by** *blast*  
**ultimately show** *?thesis*  
**using** *ConE(2) PR1* **by** *blast*

**qed**

**end**

## References

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