

# Epistemic Logic

Asta Halkjr From

September 30, 2020

## Abstract

This work is a formalization of epistemic logic with countably many agents. It includes proofs of soundness and completeness for the axiom system K. The completeness proof is based on the textbook “Reasoning About Knowledge” by Fagin, Halpern, Moses and Vardi (MIT Press 1995) [1].

## Contents

<b>1</b>	<b>Syntax</b>	<b>2</b>
<b>2</b>	<b>Semantics</b>	<b>2</b>
<b>3</b>	<b>Utility</b>	<b>2</b>
<b>4</b>	<b>S5 Axioms</b>	<b>3</b>
<b>5</b>	<b>Axiom System K</b>	<b>4</b>
<b>6</b>	<b>Soundness</b>	<b>4</b>
<b>7</b>	<b>Derived rules</b>	<b>5</b>
<b>8</b>	<b>Consistency</b>	<b>9</b>
8.1	Closure under subsets . . . . .	9
8.2	Finite character . . . . .	12
8.3	Maximal extension . . . . .	17
8.4	K consistency . . . . .	20
<b>9</b>	<b>Model existence</b>	<b>24</b>
9.1	Completeness . . . . .	34
<b>10</b>	<b>Main Result</b>	<b>35</b>
<b>11</b>	<b>Acknowledgements</b>	<b>36</b>

**theory** *Epistemic-Logic* **imports** *HOL-Library.Countable* **begin**

## 1 Syntax

**type-synonym**  $id = string$

**datatype**  $\langle 'i\ fm \rangle$   
 $= FF (\perp)$   
 $| Pro\ id$   
 $| Dis\ \langle 'i\ fm \rangle\ \langle 'i\ fm \rangle\ (\mathbf{infixr}\ \vee\ 30)$   
 $| Con\ \langle 'i\ fm \rangle\ \langle 'i\ fm \rangle\ (\mathbf{infixr}\ \wedge\ 35)$   
 $| Imp\ \langle 'i\ fm \rangle\ \langle 'i\ fm \rangle\ (\mathbf{infixr}\ \longrightarrow\ 25)$   
 $| K\ 'i\ \langle 'i\ fm \rangle$

**abbreviation**  $TT (\top)$  **where**

$\langle TT \equiv \perp \longrightarrow \perp \rangle$

**abbreviation**  $Neg (\neg - [40] 40)$  **where**

$\langle Neg\ p \equiv p \longrightarrow \perp \rangle$

## 2 Semantics

**datatype**  $\langle ('i, 's)\ kripke = Kripke (\pi: \langle 's \Rightarrow id \Rightarrow bool \rangle) (\mathcal{K}: \langle 'i \Rightarrow 's \Rightarrow 's\ set \rangle)$

**primrec**  $semantics :: \langle ('i, 's)\ kripke \Rightarrow 's \Rightarrow 'i\ fm \Rightarrow bool \rangle$

$(-, - \models - [50,50] 50)$  **where**

$\langle (-, - \models \perp) = False \rangle$

$| \langle (M, s \models Pro\ i) = \pi\ M\ s\ i \rangle$   
 $| \langle (M, s \models (p \vee q)) = ((M, s \models p) \vee (M, s \models q)) \rangle$   
 $| \langle (M, s \models (p \wedge q)) = ((M, s \models p) \wedge (M, s \models q)) \rangle$   
 $| \langle (M, s \models (p \longrightarrow q)) = ((M, s \models p) \longrightarrow (M, s \models q)) \rangle$   
 $| \langle (M, s \models K\ i\ p) = (\forall t \in \mathcal{K}\ M\ i\ s. M, t \models p) \rangle$

## 3 Utility

**abbreviation**  $reflexive :: \langle ('i, 's)\ kripke \Rightarrow bool \rangle$  **where**

$\langle reflexive\ M \equiv \forall i\ s. s \in \mathcal{K}\ M\ i\ s \rangle$

**abbreviation**  $symmetric :: \langle ('i, 's)\ kripke \Rightarrow bool \rangle$  **where**

$\langle symmetric\ M \equiv \forall i\ s\ t. t \in \mathcal{K}\ M\ i\ s \longleftrightarrow s \in \mathcal{K}\ M\ i\ t \rangle$

**abbreviation**  $transitive :: \langle ('i, 's)\ kripke \Rightarrow bool \rangle$  **where**

$\langle transitive\ M \equiv \forall i\ s\ t\ u. t \in \mathcal{K}\ M\ i\ s \wedge u \in \mathcal{K}\ M\ i\ t \longrightarrow u \in \mathcal{K}\ M\ i\ s \rangle$

**lemma**  $Imp\ intro: \langle (M, s \models p \Longrightarrow M, s \models q) \Longrightarrow M, s \models Imp\ p\ q \rangle$

**by** *simp*

## 4 S5 Axioms

**theorem** *distribution*:  $\langle M, s \models (K i p \wedge K i (p \longrightarrow q) \longrightarrow K i q) \rangle$

**proof** (*rule Imp-intro*)

**assume**  $\langle M, s \models (K i p \wedge K i (p \longrightarrow q)) \rangle$

**then have**  $\langle M, s \models K i p \rangle \langle M, s \models K i (p \longrightarrow q) \rangle$

**by** *simp-all*

**then have**  $\langle \forall t \in \mathcal{K} M i s. M, t \models p \rangle \langle \forall t \in \mathcal{K} M i s. M, t \models (p \longrightarrow q) \rangle$

**by** *simp-all*

**then have**  $\langle \forall t \in \mathcal{K} M i s. M, t \models q \rangle$

**by** *simp*

**then show**  $\langle M, s \models K i q \rangle$

**by** *simp*

**qed**

**theorem** *generalization*:

**assumes** *valid*:  $\langle \forall (M :: ('i, 's) kripke) s. M, s \models p \rangle$

**shows**  $\langle (M :: ('i, 's) kripke), s \models K i p \rangle$

**proof** –

**have**  $\langle \forall t \in \mathcal{K} M i s. M, t \models p \rangle$

**using** *valid by blast*

**then show**  $\langle M, s \models K i p \rangle$

**by** *simp*

**qed**

**theorem** *truth*:

**assumes**  $\langle \text{reflexive } M \rangle$

**shows**  $\langle M, s \models (K i p \longrightarrow p) \rangle$

**proof** (*rule Imp-intro*)

**assume**  $\langle M, s \models K i p \rangle$

**then have**  $\langle \forall t \in \mathcal{K} M i s. M, t \models p \rangle$

**by** *simp*

**moreover have**  $\langle s \in \mathcal{K} M i s \rangle$

**using**  $\langle \text{reflexive } M \rangle$  **by** *blast*

**ultimately show**  $\langle M, s \models p \rangle$

**by** *blast*

**qed**

**theorem** *pos-introspection*:

**assumes**  $\langle \text{transitive } M \rangle$

**shows**  $\langle M, s \models (K i p \longrightarrow K i (K i p)) \rangle$

**proof** (*rule Imp-intro*)

**assume**  $\langle M, s \models K i p \rangle$

**then have**  $\langle \forall t \in \mathcal{K} M i s. M, t \models p \rangle$

**by** *simp*

**then have**  $\langle \forall t \in \mathcal{K} M i s. \forall u \in \mathcal{K} M i t. M, u \models p \rangle$

**using**  $\langle \text{transitive } M \rangle$  **by** *blast*

**then have**  $\langle \forall t \in \mathcal{K} M i s. M, t \models K i p \rangle$

**by** *simp*

**then show**  $\langle M, s \models K i (K i p) \rangle$   
 by *simp*  
**qed**

**theorem** *neg-introspection*:  
**assumes**  $\langle \text{symmetric } M \rangle \langle \text{transitive } M \rangle$   
**shows**  $\langle M, s \models (\neg K i p \longrightarrow K i (\neg K i p)) \rangle$   
**proof** (*rule Imp-intro*)  
**assume**  $\langle M, s \models \neg (K i p) \rangle$   
**then obtain**  $u$  **where**  $\langle u \in \mathcal{K} M i s \rangle \langle \neg (M, u \models p) \rangle$   
 by *auto*  
**moreover have**  $\langle \forall t \in \mathcal{K} M i s. u \in \mathcal{K} M i t \rangle$   
 using  $\langle u \in \mathcal{K} M i s \rangle \langle \text{symmetric } M \rangle \langle \text{transitive } M \rangle$  **by** *blast*  
**ultimately have**  $\langle \forall t \in \mathcal{K} M i s. M, t \models \neg K i p \rangle$   
 by *auto*  
**then show**  $\langle M, s \models K i (\neg K i p) \rangle$   
 by *simp*  
**qed**

## 5 Axiom System K

**primrec** *eval* ::  $\langle (id \Rightarrow bool) \Rightarrow ('i\ fm \Rightarrow bool) \Rightarrow 'i\ fm \Rightarrow bool \rangle$  **where**  
 $\langle \text{eval } - \ - \perp = \text{False} \rangle$   
 $\langle \text{eval } g \ - \ (\text{Pro } i) = g \ i \rangle$   
 $\langle \text{eval } g \ h \ (p \vee q) = (\text{eval } g \ h \ p \vee \text{eval } g \ h \ q) \rangle$   
 $\langle \text{eval } g \ h \ (p \wedge q) = (\text{eval } g \ h \ p \wedge \text{eval } g \ h \ q) \rangle$   
 $\langle \text{eval } g \ h \ (p \longrightarrow q) = (\text{eval } g \ h \ p \longrightarrow \text{eval } g \ h \ q) \rangle$   
 $\langle \text{eval } - \ h \ (K i p) = h \ (K i p) \rangle$

**abbreviation**  $\langle \text{tautology } p \equiv \forall g \ h. \text{eval } g \ h \ p \rangle$

**inductive** *SystemK* ::  $\langle 'i\ fm \Rightarrow bool \rangle (\vdash - [50] 50)$  **where**  
 $A1: \langle \text{tautology } p \Longrightarrow \vdash p \rangle$   
 $A2: \langle \vdash (K i p \wedge K i (p \longrightarrow q)) \longrightarrow \vdash K i q \rangle$   
 $R1: \langle \vdash p \Longrightarrow \vdash (p \longrightarrow q) \Longrightarrow \vdash q \rangle$   
 $R2: \langle \vdash p \Longrightarrow \vdash K i p \rangle$

## 6 Soundness

**lemma** *eval-antics*:  $\langle \text{eval } (pi \ s) \ (\lambda q. \text{Kripke } pi \ r, s \models q) \ p = (\text{Kripke } pi \ r, s \models p) \rangle$   
 by (*induct p*) *simp-all*

**theorem** *tautology*:  $\langle \text{tautology } p \Longrightarrow M, s \models p \rangle$   
**proof** –  
**assume**  $\langle \text{tautology } p \rangle$   
**then have**  $\langle \text{eval } (g \ s) \ (\lambda q. \text{Kripke } g \ r, s \models q) \ p \rangle$  **for**  $g \ r$   
 by *simp*

**then have**  $\langle \text{Kripke } g \ r, s \models p \rangle$  **for**  $g \ r$   
**using** *eval-antics* **by** *metis*  
**then show**  $\langle M, s \models p \rangle$   
**by** (*metis kripke.collapse*)  
**qed**

**theorem** *soundness*:  $\langle \vdash p \implies M, s \models p \rangle$   
**by** (*induct p arbitrary: s rule: SystemK.induct*) (*simp-all add: tautology*)

## 7 Derived rules

**lemma** *K-FFI*:  $\langle \vdash (p \longrightarrow (\neg p) \longrightarrow \perp) \rangle$   
**by** (*simp add: A1*)

**primrec** *conjoin* ::  $\langle 'i \text{ fm list} \Rightarrow 'i \text{ fm} \Rightarrow 'i \text{ fm} \rangle$  **where**  
 $\langle \text{conjoin } [] \ q = q \rangle$   
 $\mid \langle \text{conjoin } (p \ # \ ps) \ q = (p \wedge \text{conjoin } ps \ q) \rangle$

**primrec** *imply* ::  $\langle 'i \text{ fm list} \Rightarrow 'i \text{ fm} \Rightarrow 'i \text{ fm} \rangle$  **where**  
 $\langle \text{imply } [] \ q = q \rangle$   
 $\mid \langle \text{imply } (p \ # \ ps) \ q = (p \longrightarrow \text{imply } ps \ q) \rangle$

**lemma** *K-imply-head*:  $\langle \vdash \text{imply } (p \ # \ ps) \ p \rangle$

**proof** –  
**have**  $\langle \text{tautology } (\text{imply } (p \ # \ ps) \ p) \rangle$   
**by** (*induct ps*) *simp-all*  
**then show** *?thesis*  
**using** *A1* **by** *blast*  
**qed**

**lemma** *K-imply-Cons*:

**assumes**  $\langle \vdash \text{imply } ps \ q \rangle$   
**shows**  $\langle \vdash \text{imply } (p \ # \ ps) \ q \rangle$

**proof** –  
**have**  $\langle \text{tautology } (\text{imply } ps \ q \longrightarrow \text{imply } (p \ # \ ps) \ q) \rangle$   
**by** *simp*  
**then have**  $\langle \vdash (\text{imply } ps \ q \longrightarrow \text{imply } (p \ # \ ps) \ q) \rangle$   
**using** *A1* **by** *blast*  
**then show** *?thesis*  
**using** *assms R1* **by** *blast*  
**qed**

**lemma** *K-imply-member*:  $\langle p \in \text{set } ps \implies \vdash \text{imply } ps \ p \rangle$

**proof** (*induct ps*)  
**case** *Nil*  
**then show** *?case*  
**by** *simp*  
**next**  
**case** (*Cons a ps*)

```

then show ?case
proof (cases ⟨a = p⟩)
  case True
    then show ?thesis
    using Cons K-imply-head by blast
  next
    case False
    then show ?thesis
    using Cons K-imply-Cons by simp
qed
qed

```

```

lemma K-right-mp:
  assumes ⟨⊢ imply ps p⟩ ⟨⊢ imply ps (p → q)⟩
  shows ⟨⊢ imply ps q⟩
proof –
  have ⟨tautology (imply ps p → imply ps (p → q) → imply ps q)⟩
    by (induct ps) simp-all
  then have ⟨⊢ (imply ps p → imply ps (p → q) → imply ps q)⟩
    using A1 by blast
  then show ?thesis
    using assms R1 by blast
qed

```

```

lemma tautology-imply-superset:
  assumes ⟨set ps ⊆ set qs⟩
  shows ⟨tautology (imply ps r → imply qs r)⟩
proof (rule ccontr)
  assume ⟨¬ tautology (imply ps r → imply qs r)⟩
  then obtain g h where ⟨¬ eval g h (imply ps r → imply qs r)⟩
    by blast
  then have ⟨eval g h (imply ps r)⟩ ⟨¬ eval g h (imply qs r)⟩
    by simp-all
  then consider (np) ⟨∃ p ∈ set ps. ¬ eval g h p⟩ | (r) ⟨∀ p ∈ set ps. eval g h p⟩
    ⟨eval g h r⟩
    by (induct ps) auto
  then show False
proof cases
  case np
    then have ⟨∃ p ∈ set qs. ¬ eval g h p⟩
      using ⟨set ps ⊆ set qs⟩ by blast
    then have ⟨eval g h (imply qs r)⟩
      by (induct qs) simp-all
    then show ?thesis
      using ⟨¬ eval g h (imply qs r)⟩ by blast
  next
    case r
    then have ⟨eval g h (imply qs r)⟩
      by (induct qs) simp-all

```

```

    then show ?thesis
      using  $\langle \neg \text{eval } g \ h \ (\text{imply } qs \ r) \rangle$  by blast
  qed
qed

```

```

lemma tautology-imply:  $\langle \text{tautology } q \implies \text{tautology } (\text{imply } ps \ q) \rangle$ 
  by (induct ps) simp-all

```

```

theorem K-imply-weaken:
  assumes  $\langle \vdash \text{imply } ps \ q \rangle$   $\langle \text{set } ps \subseteq \text{set } ps' \rangle$ 
  shows  $\langle \vdash \text{imply } ps' \ q \rangle$ 
proof -
  have  $\langle \text{tautology } (\text{imply } ps \ q \longrightarrow \text{imply } ps' \ q) \rangle$ 
    using  $\langle \text{set } ps \subseteq \text{set } ps' \rangle$ 
  proof (induct ps arbitrary: ps')
    case Nil
    then show ?case
      by (induct ps') simp-all
  next
    case (Cons a G)
    then show ?case
      using tautology-imply-superset by blast
  qed
  then have  $\langle \vdash (\text{imply } ps \ q \longrightarrow \text{imply } ps' \ q) \rangle$ 
    using A1 by blast
  then show ?thesis
    using  $\langle \vdash \text{imply } ps \ q \rangle$  R1 by blast
qed

```

```

lemma imply-append:  $\langle \text{imply } (ps \ @ \ ps') \ q = \text{imply } ps \ (\text{imply } ps' \ q) \rangle$ 
  by (induct ps) simp-all

```

```

lemma K-ImpI:
  assumes  $\langle \vdash \text{imply } (p \ \# \ G) \ q \rangle$ 
  shows  $\langle \vdash \text{imply } G \ (p \longrightarrow q) \rangle$ 
proof -
  have  $\langle \text{set } (p \ \# \ G) \subseteq \text{set } (G \ @ \ [p]) \rangle$ 
    by simp
  then have  $\langle \vdash \text{imply } (G \ @ \ [p]) \ q \rangle$ 
    using assms K-imply-weaken by blast
  then have  $\langle \vdash \text{imply } G \ (\text{imply } [p] \ q) \rangle$ 
    using imply-append by metis
  then show ?thesis
    by simp
qed

```

```

lemma cut:  $\langle \vdash \text{imply } G \ p \implies \vdash \text{imply } (p \ \# \ G) \ q \implies \vdash \text{imply } G \ q \rangle$ 
  using K-ImpI K-right-mp by blast

```

**lemma** *K-Boole*:  $\langle \vdash \text{imply } ((\neg p) \# G) \perp \implies \vdash \text{imply } G p \rangle$

**proof** –

**assume**  $\langle \vdash \text{imply } ((\neg p) \# G) \perp \rangle$

**then have**  $\langle \vdash \text{imply } G (\neg \neg p) \rangle$

**using** *K-ImpI* **by** *blast*

**moreover have**  $\langle \text{tautology } (\text{imply } G (\neg \neg p) \longrightarrow \text{imply } G p) \rangle$

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

**then have**  $\langle \vdash (\text{imply } G (\neg \neg p) \longrightarrow \text{imply } G p) \rangle$

**using** *A1* **by** *blast*

**ultimately show** *?thesis*

**using** *R1* **by** *blast*

**qed**

**lemma** *K-DisE*:

**assumes**  $\langle \vdash \text{imply } (A \# G) C \rangle \langle \vdash \text{imply } (B \# G) C \rangle \langle \vdash \text{imply } G (A \vee B) \rangle$

**shows**  $\langle \vdash \text{imply } G C \rangle$

**proof** –

**have**  $\langle \text{tautology } (\text{imply } (A \# G) C \longrightarrow \text{imply } (B \# G) C \longrightarrow \text{imply } G (A \vee B)) \longrightarrow \text{imply } G C \rangle$

**by**  $(\text{induct } G) \text{ auto}$

**then have**  $\langle \vdash (\text{imply } (A \# G) C \longrightarrow \text{imply } (B \# G) C \longrightarrow \text{imply } G (A \vee B)) \longrightarrow \text{imply } G C \rangle$

**using** *A1* **by** *blast*

**then show** *?thesis*

**using** *assms R1* **by** *blast*

**qed**

**lemma** *K-conjoin-imply*:

**assumes**  $\langle \vdash (\neg \text{conjoin } G (\neg p)) \rangle$

**shows**  $\langle \vdash \text{imply } G p \rangle$

**proof** –

**have**  $\langle \text{tautology } (\neg \text{conjoin } G (\neg p) \longrightarrow \text{imply } G p) \rangle$

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

**then have**  $\langle \vdash (\neg \text{conjoin } G (\neg p) \longrightarrow \text{imply } G p) \rangle$

**using** *A1* **by** *blast*

**then show** *?thesis*

**using** *assms R1* **by** *blast*

**qed**

**lemma** *K-distrib-K-imp*:

**assumes**  $\langle \vdash K i (\text{imply } G q) \rangle$

**shows**  $\langle \vdash \text{imply } (\text{map } (K i) G) (K i q) \rangle$

**proof** –

**have**  $\langle \vdash (K i (\text{imply } G q) \longrightarrow \text{imply } (\text{map } (K i) G) (K i q)) \rangle$

**proof**  $(\text{induct } G)$

**case** *Nil*

**then show** *?case*

**by**  $(\text{simp add: } A1)$

**next**



**case** (*Cons a G*)  
**have**  $\vdash (K\ i\ a \wedge K\ i\ (imply\ (a\ \# \ G)\ q) \longrightarrow K\ i\ (imply\ G\ q))$   
**by** (*simp add: A2*)  
**moreover have**  
 $\vdash ((K\ i\ a \wedge K\ i\ (imply\ (a\ \# \ G)\ q) \longrightarrow K\ i\ (imply\ G\ q)) \longrightarrow$   
 $(K\ i\ (imply\ G\ q) \longrightarrow imply\ (map\ (K\ i)\ G)\ (K\ i\ q)) \longrightarrow$   
 $(K\ i\ a \wedge K\ i\ (imply\ (a\ \# \ G)\ q) \longrightarrow imply\ (map\ (K\ i)\ G)\ (K\ i\ q)))$   
**by** (*simp add: A1*)  
**ultimately have**  $\vdash (K\ i\ a \wedge K\ i\ (imply\ (a\ \# \ G)\ q) \longrightarrow imply\ (map\ (K\ i)$   
 $G)\ (K\ i\ q))$   
**using** *Cons R1* **by** *blast*  
**moreover have**  
 $\vdash ((K\ i\ a \wedge K\ i\ (imply\ (a\ \# \ G)\ q) \longrightarrow imply\ (map\ (K\ i)\ G)\ (K\ i\ q)) \longrightarrow$   
 $(K\ i\ (imply\ (a\ \# \ G)\ q) \longrightarrow K\ i\ a \longrightarrow imply\ (map\ (K\ i)\ G)\ (K\ i\ q)))$   
**by** (*simp add: A1*)  
**ultimately have**  $\vdash (K\ i\ (imply\ (a\ \# \ G)\ q) \longrightarrow K\ i\ a \longrightarrow imply\ (map\ (K$   
 $i)\ G)\ (K\ i\ q))$   
**using** *R1* **by** *blast*  
**then show** *?case*  
**by** *simp*  
**qed**  
**then show** *?thesis*  
**using** *assms R1* **by** *blast*  
**qed**

## 8 Consistency

**definition** *consistency* ::  $\langle 'i\ fm\ set\ set \Rightarrow bool \rangle$  **where**

$\langle consistency\ C \equiv \forall S \in C.$   
 $(\forall p. \neg (Pro\ p \in S \wedge (\neg Pro\ p) \in S)) \wedge$   
 $\perp \notin S \wedge$   
 $(\forall Z. (\neg (\neg Z)) \in S \longrightarrow S \cup \{Z\} \in C) \wedge$   
 $(\forall A\ B. (A \wedge B) \in S \longrightarrow S \cup \{A, B\} \in C) \wedge$   
 $(\forall A\ B. (\neg (A \vee B)) \in S \longrightarrow S \cup \{\neg A, \neg B\} \in C) \wedge$   
 $(\forall A\ B. (A \vee B) \in S \longrightarrow S \cup \{A\} \in C \vee S \cup \{B\} \in C) \wedge$   
 $(\forall A\ B. (\neg (A \wedge B)) \in S \longrightarrow S \cup \{\neg A\} \in C \vee S \cup \{\neg B\} \in C) \wedge$   
 $(\forall A\ B. (A \longrightarrow B) \in S \longrightarrow S \cup \{\neg A\} \in C \vee S \cup \{B\} \in C) \wedge$   
 $(\forall A\ B. (\neg (A \longrightarrow B)) \in S \longrightarrow S \cup \{A, \neg B\} \in C) \wedge$   
 $(\forall A. tautology\ A \longrightarrow S \cup \{A\} \in C) \wedge$   
 $(\forall A\ i. \neg (K\ i\ A \in S \wedge (\neg K\ i\ A) \in S)) \rangle$

### 8.1 Closure under subsets

**definition** *close* ::  $\langle 'i\ fm\ set\ set \Rightarrow 'i\ fm\ set\ set \rangle$  **where**

$\langle close\ C \equiv \{S. \exists S' \in C. S \subseteq S'\} \rangle$

**definition** *subset-closed* ::  $\langle 'a\ set\ set \Rightarrow bool \rangle$  **where**

$\langle subset-closed\ C \equiv (\forall S' \in C. \forall S. S \subseteq S' \longrightarrow S \in C) \rangle$

```

lemma subset-in-close:
  assumes  $\langle S' \subseteq S \rangle \langle S \cup x \in C \rangle$ 
  shows  $\langle S' \cup x \in \text{close } C \rangle$ 
proof –
  have  $\langle S \cup x \in \text{close } C \rangle$ 
    unfolding close-def using  $\langle S \cup x \in C \rangle$  by blast
  then show ?thesis
    unfolding close-def using  $\langle S' \subseteq S \rangle$  by blast
qed

theorem close-consistency:
  fixes  $C :: \langle 'i \text{ fm set set} \rangle$ 
  assumes  $\langle \text{consistency } C \rangle$ 
  shows  $\langle \text{consistency } (\text{close } C) \rangle$ 
  unfolding consistency-def
proof (intro ballI allI impI conjI)
  fix  $S' :: \langle 'i \text{ fm set} \rangle$ 
  assume  $\langle S' \in \text{close } C \rangle$ 
  then obtain  $S$  where  $\langle S \in C \rangle \langle S' \subseteq S \rangle$ 
    unfolding close-def by blast

  { fix  $p$ 
    have  $\langle \neg (Pro\ p \in S \wedge (\neg Pro\ p) \in S) \rangle$ 
      using  $\langle S \in C \rangle \langle \text{consistency } C \rangle$  unfolding consistency-def by simp
    then show  $\langle \neg (Pro\ p \in S' \wedge (\neg Pro\ p) \in S') \rangle$ 
      using  $\langle S' \subseteq S \rangle$  by blast }

  { have  $\langle \perp \notin S \rangle$ 
    using  $\langle S \in C \rangle \langle \text{consistency } C \rangle$  unfolding consistency-def by blast
    then show  $\langle \perp \notin S' \rangle$ 
      using  $\langle S' \subseteq S \rangle$  by blast }

  { fix  $Z$ 
    assume  $\langle (\neg (\neg Z)) \in S' \rangle$ 
    then have  $\langle (\neg (\neg Z)) \in S \rangle$ 
      using  $\langle S' \subseteq S \rangle$  by blast
    then have  $\langle S \cup \{Z\} \in C \rangle$ 
      using  $\langle S \in C \rangle \langle \text{consistency } C \rangle$  unfolding consistency-def by simp
    then show  $\langle S' \cup \{Z\} \in \text{close } C \rangle$ 
      using  $\langle S' \subseteq S \rangle$  subset-in-close by blast }

  { fix  $A\ B$ 
    assume  $\langle (A \wedge B) \in S' \rangle$ 
    then have  $\langle (A \wedge B) \in S \rangle$ 
      using  $\langle S' \subseteq S \rangle$  by blast
    then have  $\langle S \cup \{A, B\} \in C \rangle$ 
      using  $\langle S \in C \rangle \langle \text{consistency } C \rangle$  unfolding consistency-def by simp
    then show  $\langle S' \cup \{A, B\} \in \text{close } C \rangle$ 
      using  $\langle S' \subseteq S \rangle$  subset-in-close by blast }

```

```

{ fix A B
  assume  $\langle \neg (A \vee B) \rangle \in S'$ 
  then have  $\langle \neg (A \vee B) \rangle \in S$ 
    using  $\langle S' \subseteq S \rangle$  by blast
  then have  $\langle S \cup \{\neg A, \neg B\} \in C \rangle$ 
    using  $\langle S \in C \rangle$  (consistency C) unfolding consistency-def by simp
  then show  $\langle S' \cup \{\neg A, \neg B\} \in \text{close } C \rangle$ 
    using  $\langle S' \subseteq S \rangle$  subset-in-close by blast }

```

```

{ fix A B
  assume  $\langle \neg (A \longrightarrow B) \rangle \in S'$ 
  then have  $\langle \neg (A \longrightarrow B) \rangle \in S$ 
    using  $\langle S' \subseteq S \rangle$  by blast
  then have  $\langle S \cup \{A, \neg B\} \in C \rangle$ 
    using  $\langle S \in C \rangle$  (consistency C) unfolding consistency-def by metis
  then show  $\langle S' \cup \{A, \neg B\} \in \text{close } C \rangle$ 
    using  $\langle S' \subseteq S \rangle$  subset-in-close by blast }

```

```

{ fix A B
  assume  $\langle A \vee B \rangle \in S'$ 
  then have  $\langle A \vee B \rangle \in S$ 
    using  $\langle S' \subseteq S \rangle$  by blast
  then have  $\langle S \cup \{A\} \in C \vee S \cup \{B\} \in C \rangle$ 
    using  $\langle S \in C \rangle$  (consistency C) unfolding consistency-def by simp
  then show  $\langle S' \cup \{A\} \in \text{close } C \vee S' \cup \{B\} \in \text{close } C \rangle$ 
    using  $\langle S' \subseteq S \rangle$  subset-in-close by blast }

```

```

{ fix A B
  assume  $\langle \neg (A \wedge B) \rangle \in S'$ 
  then have  $\langle \neg (A \wedge B) \rangle \in S$ 
    using  $\langle S' \subseteq S \rangle$  by blast
  then have  $\langle S \cup \{\neg A\} \in C \vee S \cup \{\neg B\} \in C \rangle$ 
    using  $\langle S \in C \rangle$  (consistency C) unfolding consistency-def by simp
  then show  $\langle S' \cup \{\neg A\} \in \text{close } C \vee S' \cup \{\neg B\} \in \text{close } C \rangle$ 
    using  $\langle S' \subseteq S \rangle$  subset-in-close by blast }

```

```

{ fix A B
  assume  $\langle A \longrightarrow B \rangle \in S'$ 
  then have  $\langle A \longrightarrow B \rangle \in S$ 
    using  $\langle S' \subseteq S \rangle$  by blast
  then have  $\langle S \cup \{\neg A\} \in C \vee S \cup \{B\} \in C \rangle$ 
    using  $\langle S \in C \rangle$  (consistency C) unfolding consistency-def by simp
  then show  $\langle S' \cup \{\neg A\} \in \text{close } C \vee S' \cup \{B\} \in \text{close } C \rangle$ 
    using  $\langle S' \subseteq S \rangle$  subset-in-close by blast }

```

```

{ fix A ::  $\langle 'i \text{ fm} \rangle$ 
  assume (tautology A)
  then have  $\langle S \cup \{A\} \in C \rangle$ 

```

**using**  $\langle S \in C \rangle$  *consistency C* **unfolding** *consistency-def* **by** *simp*  
**then show**  $\langle S' \cup \{A\} \in \text{close } C \rangle$   
**using**  $\langle S' \subseteq S \rangle$  *subset-in-close* **by** *blast* }

**{ fix**  $A$   $i$   
**have**  $\langle \neg (K i A \in S \wedge (\neg K i A) \in S) \rangle$   
**using**  $\langle S \in C \rangle$  *consistency C* **unfolding** *consistency-def* **by** *simp*  
**then show**  $\langle \neg (K i A \in S' \wedge (\neg K i A) \in S') \rangle$   
**using**  $\langle S' \subseteq S \rangle$  **by** *blast* }

qed

**theorem** *close-closed*:  $\langle \text{subset-closed } (\text{close } C) \rangle$   
**unfolding** *close-def subset-closed-def* **by** *blast*

**theorem** *close-subset*:  $\langle C \subseteq \text{close } C \rangle$   
**unfolding** *close-def* **by** *blast*

## 8.2 Finite character

**definition** *finite-char* ::  $\langle 'a \text{ set set } \Rightarrow \text{bool} \rangle$  **where**  
 $\langle \text{finite-char } C \equiv (\forall S. S \in C = (\forall S'. \text{finite } S' \longrightarrow S' \subseteq S \longrightarrow S' \in C)) \rangle$

**definition** *mk-finite-char* ::  $\langle 'a \text{ set set } \Rightarrow 'a \text{ set set} \rangle$  **where**  
 $\langle \text{mk-finite-char } C \equiv \{S. \forall S'. \text{finite } S' \longrightarrow S' \subseteq S \longrightarrow S' \in C\} \rangle$

**theorem** *finite-char*:  $\langle \text{finite-char } (\text{mk-finite-char } C) \rangle$   
**unfolding** *finite-char-def mk-finite-char-def* **by** *blast*

**theorem** *finite-char-closed*:  $\langle \text{finite-char } C \Longrightarrow \text{subset-closed } C \rangle$   
**unfolding** *finite-char-def subset-closed-def*

**proof** (*intro ballI allI impI*)

**fix**  $S$   $S'$

**assume** \*:  $\langle \forall S. (S \in C) = (\forall S'. \text{finite } S' \longrightarrow S' \subseteq S \longrightarrow S' \in C) \rangle$

**and**  $\langle S' \in C \rangle$   $\langle S \subseteq S' \rangle$

**then have**  $\langle \forall S'. \text{finite } S' \longrightarrow S' \subseteq S \longrightarrow S' \in C \rangle$

**by** *blast*

**then show**  $\langle S \in C \rangle$

**using** \* **by** *blast*

qed

**theorem** *finite-char-subset*:  $\langle \text{subset-closed } C \Longrightarrow C \subseteq \text{mk-finite-char } C \rangle$   
**unfolding** *mk-finite-char-def subset-closed-def* **by** *blast*

**theorem** *finite-consistency*:

**fixes**  $C$  ::  $\langle 'i \text{ fm set set} \rangle$

**assumes**  $\langle \text{consistency } C \rangle$   $\langle \text{subset-closed } C \rangle$

**shows**  $\langle \text{consistency } (\text{mk-finite-char } C) \rangle$

**unfolding** *consistency-def*

```

proof (intro ballI allI impI conjI)
  fix S
  assume ⟨S ∈ mk-finite-char C⟩
  then have finc: ⟨∀ S' ⊆ S. finite S' ⟶ S' ∈ C⟩
    unfolding mk-finite-char-def by blast

  have ⟨∀ S' ∈ C. ∀ S ⊆ S'. S ∈ C⟩
    using ⟨subset-closed C⟩ unfolding subset-closed-def by blast
  then have sc: ⟨∀ S' x. S' ∪ x ∈ C ⟶ (∀ S ⊆ S' ∪ x. S ∈ C)⟩
    by blast

  { fix i
    show ⟨¬ (Pro i ∈ S ∧ (¬ Pro i) ∈ S)⟩
    proof
      assume ⟨Pro i ∈ S ∧ (¬ Pro i) ∈ S⟩
      then have ⟨{Pro i, (¬ Pro i)} ∈ C⟩
        using finc by simp
      then show False
        using ⟨consistency C⟩ unfolding consistency-def by fast
    } qed }

  show ⟨⊥ ∉ S⟩
  proof
    assume ⟨⊥ ∈ S⟩
    then have ⟨{⊥} ∈ C⟩
      using finc by simp
    then show False
      using ⟨consistency C⟩ unfolding consistency-def by fast
  } qed

  { fix Z
    assume *: ⟨(¬ ¬ Z) ∈ S⟩
    show ⟨S ∪ {Z} ∈ mk-finite-char C⟩
      unfolding mk-finite-char-def
    proof (intro allI impI CollectI)
      fix S'
      let ?S' = ⟨S' - {Z} ∪ {¬ ¬ Z}⟩

      assume ⟨S' ⊆ S ∪ {Z}⟩ ⟨finite S'⟩
      then have ⟨?S' ⊆ S⟩
        using * by blast
      moreover have ⟨finite ?S'⟩
        using ⟨finite S'⟩ by blast
      ultimately have ⟨?S' ∈ C⟩
        using finc by blast
      then have ⟨?S' ∪ {Z} ∈ C⟩
        using ⟨consistency C⟩ unfolding consistency-def by simp
      then show ⟨S' ∈ C⟩
        using sc by blast
    }
  }

```

```

qed }

{ fix A B
  assume *:  $\langle (A \wedge B) \in S \rangle$ 
  show  $\langle S \cup \{A, B\} \in \text{mk-finite-char } C \rangle$ 
    unfolding mk-finite-char-def
  proof (intro allI impI CollectI)
    fix S'
    let ?S' =  $\langle S' - \{A, B\} \cup \{A \wedge B\} \rangle$ 

    assume  $\langle S' \subseteq S \cup \{A, B\} \rangle \langle \text{finite } S' \rangle$ 
    then have  $\langle ?S' \subseteq S \rangle$ 
      using * by blast
    moreover have  $\langle \text{finite } ?S' \rangle$ 
      using  $\langle \text{finite } S' \rangle$  by blast
    ultimately have  $\langle ?S' \in C \rangle$ 
      using fine by blast
    then have  $\langle ?S' \cup \{A, B\} \in C \rangle$ 
      using  $\langle \text{consistency } C \rangle$  unfolding consistency-def by simp
    then show  $\langle S' \in C \rangle$ 
      using sc by blast
  qed }

{ fix A B
  assume *:  $\langle (\neg (A \vee B)) \in S \rangle$ 
  show  $\langle S \cup \{\neg A, \neg B\} \in \text{mk-finite-char } C \rangle$ 
    unfolding mk-finite-char-def
  proof (intro allI impI CollectI)
    fix S'
    let ?S' =  $\langle S' - \{\neg A, \neg B\} \cup \{\neg (A \vee B)\} \rangle$ 

    assume  $\langle S' \subseteq S \cup \{\neg A, \neg B\} \rangle \langle \text{finite } S' \rangle$ 
    then have  $\langle ?S' \subseteq S \rangle$ 
      using * by blast
    moreover have  $\langle \text{finite } ?S' \rangle$ 
      using  $\langle \text{finite } S' \rangle$  by blast
    ultimately have  $\langle ?S' \in C \rangle$ 
      using fine by blast
    then have  $\langle ?S' \cup \{\neg A, \neg B\} \in C \rangle$ 
      using  $\langle \text{consistency } C \rangle$  unfolding consistency-def by simp
    then show  $\langle S' \in C \rangle$ 
      using sc by blast
  qed }

{ fix A B
  assume *:  $\langle (\neg (A \longrightarrow B)) \in S \rangle$ 
  show  $\langle S \cup \{A, \neg B\} \in \text{mk-finite-char } C \rangle$ 
    unfolding mk-finite-char-def
  proof (intro allI impI CollectI)

```

```

fix S'
let ?S' = ⟨S' - {A, ¬ B} ∪ {¬ (A → B)}⟩

assume ⟨S' ⊆ S ∪ {A, ¬ B}⟩ ⟨finite S'⟩
then have ⟨?S' ⊆ S⟩
  using * by blast
moreover have ⟨finite ?S'⟩
  using ⟨finite S'⟩ by blast
ultimately have ⟨?S' ∈ C⟩
  using fine by blast
then have ⟨?S' ∪ {A, ¬ B} ∈ C⟩
  using ⟨consistency C⟩ unfolding consistency-def by simp
then show ⟨S' ∈ C⟩
  using sc by blast
qed }

{ fix A B
  assume *: ⟨(A ∨ B) ∈ S⟩
  show ⟨S ∪ {A} ∈ mk-finite-char C ∨ S ∪ {B} ∈ mk-finite-char C⟩
  proof (rule ccontr)
    assume ¬ ?thesis
    then obtain Sa Sb
      where ⟨Sa ⊆ S ∪ {A}⟩ ⟨finite Sa⟩ ⟨Sa ∉ C⟩
        and ⟨Sb ⊆ S ∪ {B}⟩ ⟨finite Sb⟩ ⟨Sb ∉ C⟩
        unfolding mk-finite-char-def by blast

    let ?S' = ⟨(Sa - {A}) ∪ (Sb - {B}) ∪ {A ∨ B}⟩

    have ⟨?S' ⊆ S⟩
      using ⟨Sa ⊆ S ∪ {A}⟩ ⟨Sb ⊆ S ∪ {B}⟩ * by blast
    moreover have ⟨finite ?S'⟩
      using ⟨finite Sa⟩ ⟨finite Sb⟩ by blast
    ultimately have ⟨?S' ∈ C⟩
      using fine by blast
    then have ⟨?S' ∪ {A} ∈ C ∨ ?S' ∪ {B} ∈ C⟩
      using ⟨consistency C⟩ unfolding consistency-def by simp
    then have ⟨Sa ∈ C ∨ Sb ∈ C⟩
      using sc by blast
    then show False
      using ⟨Sa ∉ C⟩ ⟨Sb ∉ C⟩ by blast
  qed }

{ fix A B
  assume *: ⟨(¬ (A ∧ B)) ∈ S⟩
  show ⟨S ∪ {¬ A} ∈ mk-finite-char C ∨ S ∪ {¬ B} ∈ mk-finite-char C⟩
  proof (rule ccontr)
    assume ¬ ?thesis
    then obtain Sa Sb
      where ⟨Sa ⊆ S ∪ {¬ A}⟩ ⟨finite Sa⟩ ⟨Sa ∉ C⟩

```

```

    and  $\langle Sb \subseteq S \cup \{\neg B\} \rangle \langle \text{finite } Sb \rangle \langle Sb \notin C \rangle$ 
    unfolding mk-finite-char-def by blast

let  $?S' = \langle (Sa - \{\neg A\}) \cup (Sb - \{\neg B\}) \cup \{\neg(A \wedge B)\} \rangle$ 

have  $\langle ?S' \subseteq S \rangle$ 
  using  $\langle Sa \subseteq S \cup \{\neg A\} \rangle \langle Sb \subseteq S \cup \{\neg B\} \rangle$  * by blast
moreover have  $\langle \text{finite } ?S' \rangle$ 
  using  $\langle \text{finite } Sa \rangle \langle \text{finite } Sb \rangle$  by blast
ultimately have  $\langle ?S' \in C \rangle$ 
  using finv by blast
then have  $\langle ?S' \cup \{\neg A\} \in C \vee ?S' \cup \{\neg B\} \in C \rangle$ 
  using  $\langle \text{consistency } C \rangle$  unfolding consistency-def by simp
then have  $\langle Sa \in C \vee Sb \in C \rangle$ 
  using sc by blast
then show False
  using  $\langle Sa \notin C \rangle \langle Sb \notin C \rangle$  by blast
qed }

{ fix A B
  assume *:  $\langle (A \longrightarrow B) \in S \rangle$ 
  show  $\langle S \cup \{\neg A\} \in \text{mk-finite-char } C \vee S \cup \{B\} \in \text{mk-finite-char } C \rangle$ 
  proof (rule ccontr)
    assume  $\langle \neg ?thesis \rangle$ 
    then obtain Sa Sb
      where  $\langle Sa \subseteq S \cup \{\neg A\} \rangle \langle \text{finite } Sa \rangle \langle Sa \notin C \rangle$ 
      and  $\langle Sb \subseteq S \cup \{B\} \rangle \langle \text{finite } Sb \rangle \langle Sb \notin C \rangle$ 
      unfolding mk-finite-char-def by blast

let  $?S' = \langle (Sa - \{\neg A\}) \cup (Sb - \{B\}) \cup \{A \longrightarrow B\} \rangle$ 

have  $\langle ?S' \subseteq S \rangle$ 
  using  $\langle Sa \subseteq S \cup \{\neg A\} \rangle \langle Sb \subseteq S \cup \{B\} \rangle$  * by blast
moreover have  $\langle \text{finite } ?S' \rangle$ 
  using  $\langle \text{finite } Sa \rangle \langle \text{finite } Sb \rangle$  by blast
ultimately have  $\langle ?S' \in C \rangle$ 
  using finv by blast
then have  $\langle ?S' \cup \{\neg A\} \in C \vee ?S' \cup \{B\} \in C \rangle$ 
  using  $\langle \text{consistency } C \rangle$  unfolding consistency-def by simp
then have  $\langle Sa \in C \vee Sb \in C \rangle$ 
  using sc by blast
then show False
  using  $\langle Sa \notin C \rangle \langle Sb \notin C \rangle$  by blast
qed }

{ fix A ::  $\langle 'i \text{ fm} \rangle$ 
  assume *:  $\langle \text{tautology } A \rangle$ 
  show  $\langle S \cup \{A\} \in \text{mk-finite-char } C \rangle$ 
  unfolding mk-finite-char-def

```



```

proof (intro allI impI CollectI)
  fix S'
  let ?S' = ⟨S' - {A}⟩

  assume ⟨S' ⊆ S ∪ {A}⟩ (finite S')
  then have ⟨?S' ⊆ S⟩
    using * by blast
  moreover have (finite ?S')
    using (finite S') by blast
  ultimately have ⟨?S' ∈ C⟩
    using fine by blast
  then have ⟨S' ∪ {A} ∈ C⟩
    using * (consistency C) unfolding consistency-def by metis
  then show ⟨S' ∈ C⟩
    using sc by blast
qed }

```

```

{ fix A i
  assume ⟨S ∈ mk-finite-char C⟩
  show ⟨¬ (K i A ∈ S ∧ (¬ K i A) ∈ S)⟩
  proof
    assume ⟨K i A ∈ S ∧ (¬ K i A) ∈ S⟩
    then have ⟨{K i A, (¬ K i A)} ∈ C⟩
      using fine by simp
    then show False
      using (consistency C) unfolding consistency-def by auto
  qed }

```

qed

### 8.3 Maximal extension

```

instantiation fm :: (countable) countable begin
instance by countable-datatype
end

```

```

definition is-chain :: ⟨(nat ⇒ 'a set) ⇒ bool⟩ where
  (is-chain f ≡ ∀ n. f n ⊆ f (Suc n))

```

```

lemma is-chainD: ⟨is-chain f ⇒ x ∈ f m ⇒ x ∈ f (m + n)⟩
  by (induct n) (auto simp: is-chain-def)

```

```

lemma is-chainD':
  assumes ⟨is-chain f⟩ ⟨x ∈ f m⟩ ⟨m ≤ k⟩
  shows ⟨x ∈ f k⟩
proof -
  obtain n where ⟨k = m + n⟩
    using ⟨m ≤ k⟩ le-iff-add
    by metis

```

**then show**  $\langle x \in f k \rangle$   
**using**  $\langle \text{is-chain } f \rangle \langle x \in f m \rangle \text{is-chainD}$   
**by** *metis*  
**qed**

**lemma** *chain-index*:  
**assumes**  $\langle \text{is-chain } f \rangle \langle \text{finite } F \rangle$   
**shows**  $\langle F \subseteq (\bigcup n. f n) \implies \exists n. F \subseteq f n \rangle$   
**using**  $\langle \text{finite } F \rangle$   
**proof** (*induct F rule: finite-induct*)  
**case** *empty*  
**then show** *?case*  
**by** *blast*  
**next**  
**case** (*insert x F*)  
**then have**  $\langle \exists n. F \subseteq f n \rangle \langle \exists m. x \in f m \rangle \langle F \subseteq (\bigcup x. f x) \rangle$   
**using**  $\langle \text{is-chain } f \rangle$  **by** *simp-all*  
**then obtain** *n* **and** *m* **where**  $\langle F \subseteq f n \rangle \langle x \in f m \rangle$   
**by** *blast*  
**have**  $\langle m \leq \max n m \rangle \langle n \leq \max n m \rangle$   
**by** *simp-all*  
**have**  $\langle x \in f (\max n m) \rangle$   
**using**  $\text{is-chainD}' \langle \text{is-chain } f \rangle \langle x \in f m \rangle \langle m \leq \max n m \rangle$   
**by** *fast*  
**moreover have**  $\langle F \subseteq f (\max n m) \rangle$   
**using**  $\text{is-chainD}' \langle \text{is-chain } f \rangle \langle F \subseteq f n \rangle \langle n \leq \max n m \rangle$   
**by** *fast*  
**ultimately show** *?case*  
**by** *blast*  
**qed**

**lemma** *chain-union-closed'*:  
**assumes**  $\langle \text{is-chain } f \rangle \langle \forall n. f n \in C \rangle \langle \forall S' \in C. \forall S \subseteq S'. S \in C \rangle \langle \text{finite } S' \rangle \langle S' \subseteq (\bigcup n. f n) \rangle$   
**shows**  $\langle S' \in C \rangle$   
**proof** –  
**obtain** *n* **where**  $\langle S' \subseteq f n \rangle$   
**using** *chain-index*  $\langle \text{is-chain } f \rangle \langle \text{finite } S' \rangle \langle S' \subseteq (\bigcup n. f n) \rangle$   
**by** *blast*  
**moreover have**  $\langle f n \in C \rangle$   
**using**  $\langle \forall n. f n \in C \rangle$   
**by** *blast*  
**ultimately show**  $\langle S' \in C \rangle$   
**using**  $\langle \forall S' \in C. \forall S \subseteq S'. S \in C \rangle$   
**by** *blast*  
**qed**

**lemma** *chain-union-closed*:  
**assumes**  $\langle \text{finite-char } C \rangle \langle \text{is-chain } f \rangle \langle \forall n. f n \in C \rangle$

**shows**  $\langle (\bigcup n. f n) \in C \rangle$   
**proof** –  
**have**  $\langle \text{subset-closed } C \rangle$   
**using**  $\text{finite-char-closed } \langle \text{finite-char } C \rangle$   
**by** *blast*  
**then have**  $\langle \forall S' \in C. \forall S \subseteq S'. S \in C \rangle$   
**unfolding**  $\text{subset-closed-def}$   
**by** *blast*  
**then have**  $\langle \forall S'. \text{finite } S' \longrightarrow S' \subseteq (\bigcup n. f n) \longrightarrow S' \in C \rangle$   
**using**  $\text{chain-union-closed' } \text{assms}$   
**by** *blast*  
**moreover have**  $\langle ((\bigcup n. f n) \in C) = (\forall S'. \text{finite } S' \longrightarrow S' \subseteq (\bigcup n. f n) \longrightarrow S' \in C) \rangle$   
**using**  $\langle \text{finite-char } C \rangle$  **unfolding**  $\text{finite-char-def}$   
**by** *blast*  
**ultimately show** *?thesis*  
**by** *blast*  
**qed**

**primrec**  $\text{extend} :: 'i \text{ fm set} \Rightarrow 'i \text{ fm set set} \Rightarrow (\text{nat} \Rightarrow 'i \text{ fm}) \Rightarrow \text{nat} \Rightarrow 'i \text{ fm set}$   
**where**  
 $\langle \text{extend } S \ C \ f \ 0 = S \rangle |$   
 $\langle \text{extend } S \ C \ f \ (\text{Suc } n) =$   
 $\quad \langle \text{if } \text{extend } S \ C \ f \ n \cup \{f \ n\} \in C$   
 $\quad \text{then } \text{extend } S \ C \ f \ n \cup \{f \ n\}$   
 $\quad \text{else } \text{extend } S \ C \ f \ n \rangle$

**definition**  $\text{Extend} :: 'i \text{ fm set} \Rightarrow 'i \text{ fm set set} \Rightarrow (\text{nat} \Rightarrow 'i \text{ fm}) \Rightarrow 'i \text{ fm set}$   
**where**  
 $\langle \text{Extend } S \ C \ f \equiv \bigcup n. \text{extend } S \ C \ f \ n \rangle$

**lemma**  $\text{is-chain-extend}$ :  $\langle \text{is-chain } (\text{extend } S \ C \ f) \rangle$   
**by**  $(\text{simp } \text{add: } \text{is-chain-def})$  *blast*

**lemma**  $\text{extend-in-C}$ :  $\langle \text{consistency } C \Longrightarrow S \in C \Longrightarrow \text{extend } S \ C \ f \ n \in C \rangle$   
**by**  $(\text{induct } n)$  *simp-all*

**theorem**  $\text{Extend-in-C}$ :  $\langle \text{consistency } C \Longrightarrow \text{finite-char } C \Longrightarrow S \in C \Longrightarrow \text{Extend } S \ C \ f \in C \rangle$   
**using**  $\text{chain-union-closed } \text{is-chain-extend } \text{extend-in-C}$  **unfolding**  $\text{Extend-def}$   
**by** *blast*

**theorem**  $\text{Extend-subset}$ :  $\langle S \subseteq \text{Extend } S \ C \ f \rangle$   
**unfolding**  $\text{Extend-def}$  **using**  $\text{Union-upper } \text{extend.simps}(1)$   $\text{range-eqI}$   
**by** *metis*

**definition**  $\text{maximal} :: 'a \text{ set} \Rightarrow 'a \text{ set set} \Rightarrow \text{bool}$  **where**  
 $\langle \text{maximal } S \ C \equiv \forall S' \in C. S \subseteq S' \longrightarrow S = S' \rangle$

**theorem** *Extend-maximal*:  
**assumes**  $\langle \forall y :: 'i \text{ fm. } \exists n. y = f n \rangle$  *finite-char C*  
**shows**  $\langle \text{maximal } (\text{Extend } S \ C \ f) \ C \rangle$   
**unfolding** *maximal-def Extend-def*  
**proof** (*intro ballI impI*)  
**fix**  $S'$   
**assume**  $\langle S' \in C \rangle \langle \bigcup x. \text{extend } S \ C \ f \ x \subseteq S' \rangle$   
**moreover have**  $\langle S' \subseteq \bigcup x. \text{extend } S \ C \ f \ x \rangle$   
**proof** (*rule ccontr*)  
**assume**  $\langle \neg S' \subseteq \bigcup x. \text{extend } S \ C \ f \ x \rangle$   
**then obtain**  $z$  **where**  $\langle z \in S' \rangle \langle z \notin \bigcup x. \text{extend } S \ C \ f \ x \rangle$   
**by** *blast*  
**then obtain**  $n$  **where**  $\langle z = f n \rangle$   
**using**  $\langle \forall y. \exists n. y = f n \rangle$   
**by** *blast*  
  
**have**  $\langle \text{extend } S \ C \ f \ n \cup \{f n\} \subseteq S' \rangle$   
**using**  $\langle \bigcup x. \text{extend } S \ C \ f \ x \subseteq S' \rangle \langle z = f n \rangle \langle z \in S' \rangle$   
**by** *blast*  
  
**have**  $\langle \text{subset-closed } C \rangle$   
**using**  $\langle \text{finite-char } C \rangle$  *finite-char-closed*  
**by** *blast*  
**then have**  $\langle \forall S' \in C. \forall S \subseteq S'. S \in C \rangle$   
**unfolding** *subset-closed-def*  
**by** *simp*  
**then have**  $\langle \forall S \subseteq S'. S \in C \rangle$   
**using**  $\langle S' \in C \rangle$   
**by** *blast*  
**then have**  $\langle \text{extend } S \ C \ f \ n \cup \{f n\} \in C \rangle$   
**using**  $\langle \text{extend } S \ C \ f \ n \cup \{f n\} \subseteq S' \rangle$   
**by** *blast*  
**then have**  $\langle z \in \text{extend } S \ C \ f \ (\text{Suc } n) \rangle$   
**using**  $\langle z \notin \bigcup x. \text{extend } S \ C \ f \ x \rangle \langle z = f n \rangle$   
**by** *simp*  
**then show** *False*  
**using**  $\langle z \notin \bigcup x. \text{extend } S \ C \ f \ x \rangle$   
**by** *blast*  
**qed**  
**ultimately show**  $\langle \bigcup x. \text{extend } S \ C \ f \ x = S' \rangle$   
**by** *simp*  
**qed**

## 8.4 K consistency

**theorem** *K-consistency*:  $\langle \text{consistency } \{ \text{set } G \mid G. \neg \vdash \text{ imply } G \ \perp \} \rangle$   
**unfolding** *consistency-def*  
**proof** (*intro conjI ballI allI impI notI*)  
**fix**  $S :: \langle 'i \text{ fm set} \rangle$

assume  $\langle S \in \{set\ G \mid G. \neg \vdash imply\ G\ \perp\} \rangle$  (is  $\langle S \in ?C \rangle$ )  
then obtain  $G$  where  $*$ :  $\langle S = set\ G \rangle$  and  $\langle \neg \vdash imply\ G\ \perp \rangle$   
by *blast*

```
{ fix i
  assume  $\langle Pro\ i \in S \wedge (\neg\ Pro\ i) \in S \rangle$ 
  then have  $\langle \vdash imply\ G\ (Pro\ i) \rangle \langle \vdash imply\ G\ (\neg\ Pro\ i) \rangle$ 
    using K-imp-ly-member * by blast+
  then have  $\langle \vdash imply\ G\ \perp \rangle$ 
    using K-FFI K-right-mp by blast
  then show False
    using  $\langle \neg \vdash imply\ G\ \perp \rangle$  by blast }
```

```
{ assume  $\langle \perp \in S \rangle$ 
  then have  $\langle \vdash imply\ G\ \perp \rangle$ 
    using K-imp-ly-member * by blast
  then show False
    using  $\langle \neg \vdash imply\ G\ \perp \rangle$  by blast }
```

```
{ fix Z
  assume  $\langle (\neg \neg Z) \in S \rangle$ 
  then have  $\langle \vdash imply\ G\ (\neg \neg Z) \rangle$ 
    using K-imp-ly-member * by simp
  then have  $\langle \neg \vdash imply\ (Z \# G)\ \perp \rangle$ 
    using K-Impl K-right-mp  $\langle \neg \vdash imply\ G\ \perp \rangle$  by blast
  moreover have  $\langle S \cup \{Z\} = set\ (Z \# G) \rangle$ 
    using * by simp
  ultimately show  $\langle S \cup \{Z\} \in ?C \rangle$ 
    by blast }
```

```
{ fix A B
  assume  $\langle (A \wedge B) \in S \rangle$ 
  then have  $\langle \vdash imply\ G\ (A \wedge B) \rangle$ 
    using K-imp-ly-member * by simp
  moreover have  $\langle \vdash ((A \wedge B) \longrightarrow A) \rangle \langle \vdash ((A \wedge B) \longrightarrow B) \rangle$ 
    using A1 by force+
  ultimately have  $\langle \vdash imply\ G\ A \rangle \langle \vdash imply\ G\ B \rangle$ 
    using K-right-mp K-imp-ly-head R1 by (metis imp-ly.simps(2))+
  then have  $\langle \neg \vdash imply\ (A \# B \# G)\ \perp \rangle$ 
    using K-imp-ly-Cons  $\langle \neg \vdash imply\ G\ \perp \rangle$  cut by metis
  moreover have  $\langle S \cup \{A, B\} = set\ (A \# B \# G) \rangle$ 
    using * by simp
  ultimately show  $\langle S \cup \{A, B\} \in ?C \rangle$ 
    by blast }
```

```
{ fix A B
  assume  $\langle (\neg (A \vee B)) \in S \rangle$ 
  then have  $\langle \vdash imply\ ((\neg B) \# G)\ (\neg (A \vee B)) \rangle \langle \vdash imply\ G\ (\neg (A \vee B)) \rangle$ 
    using * K-imp-ly-Cons K-imp-ly-member by blast+
```

```

moreover have  $\langle \vdash \text{ imply } ((\neg B) \# G) (\neg (A \vee B) \longrightarrow \neg A) \rangle \langle \vdash \text{ imply } G (\neg (A \vee B) \longrightarrow \neg B) \rangle$ 
  by (simp-all add: A1 tautology-imply)
ultimately have  $\langle \vdash \text{ imply } ((\neg B) \# G) (\neg A) \rangle \langle \vdash \text{ imply } G (\neg B) \rangle$ 
  using K-right-mp by blast+

then have  $\langle \neg \vdash \text{ imply } ((\neg A) \# (\neg B) \# G) \perp \rangle$ 
  using  $\langle \neg \vdash \text{ imply } G \perp \rangle$  cut by blast
moreover have  $\langle S \cup \{\neg A, \neg B\} = \text{set } ((\neg A) \# (\neg B) \# G) \rangle$ 
  using * by simp
ultimately show  $\langle S \cup \{\neg A, \neg B\} \in ?C \rangle$ 
  by blast }

{ fix A B
  assume  $\langle (\neg (A \longrightarrow B)) \in S \rangle$ 
  then have  $\langle \vdash \text{ imply } ((\neg B) \# G) (\neg (A \longrightarrow B)) \rangle \langle \vdash \text{ imply } G (\neg (A \longrightarrow B)) \rangle$ 
    using * K-imply-member K-imply-Cons by blast+
  moreover have
     $\langle \vdash \text{ imply } ((\neg B) \# G) (\neg (A \longrightarrow B) \longrightarrow A) \rangle$ 
     $\langle \vdash \text{ imply } G (\neg (A \longrightarrow B) \longrightarrow \neg B) \rangle$ 
    by (simp-all add: A1 tautology-imply)
  ultimately have  $\langle \vdash \text{ imply } ((\neg B) \# G) A \rangle \langle \vdash \text{ imply } G (\neg B) \rangle$ 
    using K-right-mp by blast+

  then have  $\langle \neg \vdash \text{ imply } (A \# (\neg B) \# G) \perp \rangle$ 
    using  $\langle \neg \vdash \text{ imply } G \perp \rangle$  cut by blast
  moreover have  $\langle S \cup \{A, \neg B\} = \text{set } (A \# (\neg B) \# G) \rangle$ 
    using * by simp
  ultimately show  $\langle S \cup \{A, \neg B\} \in ?C \rangle$ 
    by blast }

{ fix A B
  assume  $\langle (A \vee B) \in S \rangle$ 
  then have  $\langle \vdash \text{ imply } G (A \vee B) \rangle$ 
    using * K-imply-member by simp

  { assume  $\langle (\forall G'. \text{set } G' = S \cup \{A\} \longrightarrow \vdash \text{ imply } G' \perp) \rangle$ 
    and  $\langle (\forall G'. \text{set } G' = S \cup \{B\} \longrightarrow \vdash \text{ imply } G' \perp) \rangle$ 
    then have  $\langle \vdash \text{ imply } (A \# G) \perp \rangle \langle \vdash \text{ imply } (B \# G) \perp \rangle$ 
      using * by (metis Un-insert-right list.simps(15) sup-bot.right-neutral)+
    then have  $\langle \vdash \text{ imply } G \perp \rangle$ 
      using  $\langle \vdash \text{ imply } G (A \vee B) \rangle$  K-DisE by blast
    then have False
      using  $\langle \neg \vdash \text{ imply } G \perp \rangle$  by blast }
  then show  $\langle S \cup \{A\} \in ?C \vee S \cup \{B\} \in ?C \rangle$ 
    by blast }

{ fix A B
  assume  $\langle (\neg (A \wedge B)) \in S \rangle$ 

```

```

then have  $\langle \vdash \text{imply } G (\neg (A \wedge B)) \rangle$ 
  using * K-imply-member by blast
moreover have  $\langle \vdash \text{imply } G (\neg (A \wedge B) \longrightarrow (\neg A) \vee (\neg B)) \rangle$ 
  by (simp add: A1 tautology-imply)
ultimately have  $\langle \vdash \text{imply } G ((\neg A) \vee (\neg B)) \rangle$ 
  using K-right-mp by blast

{ assume  $\langle (\forall G'. \text{ set } G' = S \cup \{\neg A\} \longrightarrow \vdash \text{imply } G' \perp) \rangle$ 
  and  $\langle (\forall G'. \text{ set } G' = S \cup \{\neg B\} \longrightarrow \vdash \text{imply } G' \perp) \rangle$ 
  then have  $\langle \vdash \text{imply } ((\neg A) \# G) \perp \rangle \langle \vdash \text{imply } ((\neg B) \# G) \perp \rangle$ 
    using * by (metis Un-insert-right list.simps(15) sup-bot.right-neutral) +
  then have  $\langle \vdash \text{imply } G \perp \rangle$ 
    using K-DisE  $\langle \vdash \text{imply } G ((\neg A) \vee (\neg B)) \rangle$  by blast
  then have False
    using  $\langle \neg \vdash \text{imply } G \perp \rangle$  by blast }
then show  $\langle S \cup \{\neg A\} \in ?C \vee S \cup \{\neg B\} \in ?C \rangle$ 
  by blast }

{ fix A B
  assume  $\langle (A \longrightarrow B) \in S \rangle$ 
  then have  $\langle \vdash \text{imply } G (A \longrightarrow B) \rangle$ 
    using * K-imply-member by blast
  moreover have  $\langle \vdash \text{imply } G ((A \longrightarrow B) \longrightarrow (\neg A) \vee B) \rangle$ 
    by (simp add: A1 tautology-imply)
  ultimately have  $\langle \vdash \text{imply } G ((\neg A) \vee B) \rangle$ 
    using K-right-mp by blast

  { assume  $\langle (\forall G'. \text{ set } G' = S \cup \{\neg A\} \longrightarrow \vdash \text{imply } G' \perp) \rangle$ 
    and  $\langle (\forall G'. \text{ set } G' = S \cup \{B\} \longrightarrow \vdash \text{imply } G' \perp) \rangle$ 
    then have  $\langle \vdash \text{imply } ((\neg A) \# G) \perp \rangle \langle \vdash \text{imply } (B \# G) \perp \rangle$ 
      using * by (metis Un-insert-right list.simps(15) sup-bot.right-neutral) +
    then have  $\langle \vdash \text{imply } G \perp \rangle$ 
      using K-DisE  $\langle \vdash \text{imply } G ((\neg A) \vee B) \rangle$  by blast
    then have False
      using  $\langle \neg \vdash \text{imply } G \perp \rangle$  by blast }
  then show  $\langle S \cup \{\neg A\} \in ?C \vee S \cup \{B\} \in ?C \rangle$ 
    by blast }

{ fix A ::  $\langle 'i \text{ fm} \rangle$ 
  assume tautology A
  then have  $\langle \vdash \text{imply } G A \rangle$ 
    using tautology-imply A1 by blast
  then have  $\langle \vdash \text{imply } (A \# G) \perp \implies \vdash \text{imply } G \perp \rangle$ 
    using cut by blast
  moreover have  $\langle S \cup \{A\} = \text{set } (A \# G) \rangle$ 
    using * by simp
  ultimately show  $\langle S \cup \{A\} \in ?C \rangle$ 
    using  $\langle \neg \vdash \text{imply } G \perp \rangle$  by blast }

```

```

{ fix A i
  assume ⟨K i A ∈ S ∧ (¬ K i A) ∈ S⟩
  then have ⟨⊢ imply G (K i A)⟩ ⟨⊢ imply G (¬ K i A)⟩
    using K-imply-member * by blast+
  then have ⟨⊢ imply G ⊥⟩
    using K-FFI K-right-mp by blast
  then show False
    using ⟨¬ ⊢ imply G ⊥⟩ by blast }

```

qed

**theorem** *K-finite-consistency*: ⟨consistency (mk-finite-char (close {set G | G. ¬ ⊢ imply G ⊥}))⟩  
 using K-consistency finite-consistency close-closed close-consistency by blast

**theorem** *K-concrete-finite-consistency*:  
 defines ⟨C ≡ mk-finite-char (close {set G | G. ¬ ⊢ imply G ⊥})⟩  
 assumes ⟨set G ∈ C⟩  
 shows ⟨¬ ⊢ imply G ⊥⟩  
 using assms unfolding C-def mk-finite-char-def close-def  
 using K-imply-weaken by fastforce

## 9 Model existence

**lemma** *at-least-one-in-maximal*:

```

assumes ⟨consistency C⟩ ⟨V ∈ C⟩ ⟨maximal V C⟩
shows ⟨p ∈ V ∨ (¬ p) ∈ V⟩

```

**proof** –

```

have ⟨V ∪ {p} ∈ C ∨ V ∪ {¬ p} ∈ C⟩

```

**proof** (rule ccontr)

```

assume ⟨¬ (V ∪ {p} ∈ C ∨ V ∪ {¬ p} ∈ C)⟩

```

```

then have ⟨V ∪ {p ∨ ¬ p} ∉ C⟩

```

```

  using assms unfolding consistency-def maximal-def

```

```

  by (metis (no-types, hide-lams) insert-iff subset-iff sup.absorb-iff1 sup.cobounded1)

```

```

then show False

```

```

  using assms unfolding consistency-def maximal-def

```

```

  by simp

```

qed

```

then show ?thesis

```

```

  using ⟨maximal V C⟩ unfolding maximal-def

```

```

  by fast

```

qed

**lemma** *at-most-one-in-maximal*:

```

assumes ⟨consistency C⟩ ⟨V ∈ C⟩ ⟨maximal V C⟩
shows ⟨¬ (p ∈ V ∧ (¬ p) ∈ V)⟩

```

**proof** (induct p)

```

case FF

```

```

then show ?case

```



```

    using assms unfolding consistency-def by blast
next
case (Pro x)
then show ?case
  using assms unfolding consistency-def by simp
next
case (Dis p q)
show ?case
proof
  assume  $\langle p \vee q \rangle \in V \wedge \langle \neg (p \vee q) \rangle \in V$ 
  then have  $\langle V \cup \{p\} \rangle \in C \vee \langle V \cup \{q\} \rangle \in C \wedge \langle V \cup \{\neg p, \neg q\} \rangle \in C$ 
    using  $\langle \text{consistency } C \rangle \langle V \in C \rangle$  unfolding consistency-def by simp-all
  then have  $\langle p \in V \vee q \in V \rangle \langle \neg p \rangle \in V \wedge \langle \neg q \rangle \in V$ 
    using  $\langle \text{maximal } V C \rangle$  unfolding maximal-def by fast+
  then show False
    using Dis by blast
qed
next
case (Con p q)
show ?case
proof
  assume  $\langle p \wedge q \rangle \in V \wedge \langle \neg (p \wedge q) \rangle \in V$ 
  then have  $\langle V \cup \{p, q\} \rangle \in C \wedge \langle V \cup \{\neg p\} \rangle \in C \vee \langle V \cup \{\neg q\} \rangle \in C$ 
    using  $\langle \text{consistency } C \rangle \langle V \in C \rangle$  unfolding consistency-def by simp-all
  then have  $\langle p \in V \rangle \langle q \in V \rangle \langle \neg p \rangle \in V \vee \langle \neg q \rangle \in V$ 
    using  $\langle \text{maximal } V C \rangle$  unfolding maximal-def by fast+
  then show False
    using Con by blast
qed
next
case (Imp p q)
show ?case
proof
  assume  $\langle p \longrightarrow q \rangle \in V \wedge \langle \neg (p \longrightarrow q) \rangle \in V$ 
  then have  $\langle V \cup \{\neg p\} \rangle \in C \vee \langle V \cup \{q\} \rangle \in C \wedge \langle V \cup \{p, \neg q\} \rangle \in C$ 
    using  $\langle \text{consistency } C \rangle \langle V \in C \rangle$  unfolding consistency-def by simp-all
  then have  $\langle \neg p \rangle \in V \vee \langle q \in V \rangle \langle p \in V \rangle \langle \neg q \rangle \in V$ 
    using  $\langle \text{maximal } V C \rangle$  unfolding maximal-def by fast+
  then show False
    using Imp by blast
qed
next
case (K i p)
then show ?case
  using  $\langle \text{consistency } C \rangle \langle V \in C \rangle$  unfolding consistency-def
  by simp
qed

```

**theorem** *exactly-one-in-maximal*:

**assumes**  $\langle \text{consistency } C \rangle \langle V \in C \rangle \langle \text{maximal } V C \rangle$   
**shows**  $\langle (p \in V) \neq ((\neg p) \in V) \rangle$   
**using** *assms at-least-one-in-maximal at-most-one-in-maximal* **by** *blast*

**theorem** *conjuncts-in-maximal:*

**assumes**  $\langle \text{consistency } C \rangle \langle V \in C \rangle \langle \text{maximal } V C \rangle$   
**shows**  $\langle (p \wedge q) \in V \longleftrightarrow p \in V \wedge q \in V \rangle$

**proof**

**assume**  $\langle (p \wedge q) \in V \rangle$   
**have**  $\langle V \cup \{p, q\} \in C \rangle$   
**using**  $\langle (p \wedge q) \in V \rangle \langle \text{consistency } C \rangle \langle V \in C \rangle$  **unfolding** *consistency-def*  
**by** *simp*  
**then show**  $\langle p \in V \wedge q \in V \rangle$   
**using**  $\langle \text{maximal } V C \rangle$  **unfolding** *maximal-def*  
**by** *fast*

**next**

**assume**  $\langle p \in V \wedge q \in V \rangle$   
**show**  $\langle (p \wedge q) \in V \rangle$   
**proof** (*rule ccontr*)  
**assume**  $\langle (p \wedge q) \notin V \rangle$   
**then have**  $\langle (\neg (p \wedge q)) \in V \rangle$   
**using** *at-least-one-in-maximal assms* **by** *blast*  
**then have**  $\langle V \cup \{\neg p\} \in C \vee V \cup \{\neg q\} \in C \rangle$   
**using**  $\langle \text{consistency } C \rangle \langle V \in C \rangle$  **unfolding** *consistency-def*  
**by** *simp*  
**then have**  $\langle (\neg p) \in V \vee (\neg q) \in V \rangle$   
**using**  $\langle \text{maximal } V C \rangle$  **unfolding** *maximal-def*  
**by** *fast*  
**then show** *False*  
**using**  $\langle p \in V \wedge q \in V \rangle$  *assms at-most-one-in-maximal*  
**by** *metis*

**qed**

**qed**

**theorem** *consequent-in-maximal:*

**assumes**  $\langle \text{consistency } C \rangle \langle V \in C \rangle \langle \text{maximal } V C \rangle \langle p \in V \rangle \langle (p \longrightarrow q) \in V \rangle$   
**shows**  $\langle q \in V \rangle$

**proof** –

**consider**  $\langle np \rangle \langle V \cup \{\neg p\} \in C \rangle \mid \langle q \rangle \langle V \cup \{q\} \in C \rangle$   
**using**  $\langle (p \longrightarrow q) \in V \rangle \langle \text{consistency } C \rangle \langle V \in C \rangle$  **unfolding** *consistency-def*  
**by** *metis*

**then show** *?thesis*

**proof** *cases*

**case** *np*

**then have**  $\langle (\neg p) \in V \rangle$   
**using**  $\langle \text{maximal } V C \rangle$  **unfolding** *maximal-def*  
**by** *fast*

**then show** *?thesis*

**using** *assms at-most-one-in-maximal*

```

    by blast
  next
  case q
  then show ?thesis
    using ⟨maximal V C⟩ unfolding maximal-def
    by fast
  qed
qed

```

**lemma** *K-not-neg-in-consistency*:  $\langle \neg \vdash (\neg p) \implies \{p\} \in \{\text{set } G \mid G. \neg \vdash \text{imply } G \perp\} \rangle$

```

proof –
  assume ⟨ $\neg \vdash (\neg p)$ ⟩
  moreover have ⟨ $\text{set } [p] = \{p\}$ ⟩
    by simp
  moreover have ⟨ $\text{imply } [p] \perp = (\neg p)$ ⟩
    by simp
  ultimately show ?thesis
    by fastforce
qed

```

**lemma** *K-inconsistent-neg*:

```

defines ⟨ $C \equiv \text{mk-finite-char } (\text{close } \{\text{set } G \mid G. \neg \vdash \text{imply } G \perp\})$ ⟩
assumes ⟨ $\{p\} \notin C$ ⟩
shows ⟨ $\vdash (\neg p)$ ⟩
using assms unfolding C-def mk-finite-char-def close-def
using K-not-neg-in-consistency by fastforce

```

**lemma** *conjuncts-in-consistency*:

```

assumes ⟨consistency C⟩ ⟨subset-closed C⟩ ⟨ $S \cup \{p \wedge q\} \in C$ ⟩
shows ⟨ $S \cup \{p, q\} \in C$ ⟩
proof –
  let ?S = ⟨ $S \cup \{p \wedge q\}$ ⟩
  have ⟨ $?S \in C$ ⟩
    using assms by blast
  moreover have ⟨ $(p \wedge q) \in ?S$ ⟩
    by simp
  ultimately have ⟨ $?S \cup \{p, q\} \in C$ ⟩
    using assms unfolding consistency-def by simp
  moreover have ⟨ $S \cup \{p, q\} \subseteq ?S \cup \{p, q\}$ ⟩
    by blast
  ultimately show ⟨ $S \cup \{p, q\} \in C$ ⟩
    using ⟨subset-closed C⟩ unfolding subset-closed-def by simp
qed

```

**lemma** *conjoined-in-consistency*:

```

assumes ⟨consistency C⟩ ⟨subset-closed C⟩ ⟨ $S \cup \{\text{conjoin } ps \ q\} \in C$ ⟩
shows ⟨ $S \cup \text{set } ps \cup \{q\} \in C$ ⟩
using ⟨ $S \cup \{\text{conjoin } ps \ q\} \in C$ ⟩

```

```

proof (induct ps arbitrary: S)
  case Nil
  then show ?case
    using ⟨subset-closed C⟩ unfolding subset-closed-def by simp
next
  case (Cons p ps)
  then have ⟨S ∪ {p, conjoin ps q} ∈ C⟩
    using assms conjuncts-in-consistency by auto
  then have ⟨S ∪ {p} ∪ {conjoin ps q} ∈ C⟩
    by (metis insert-is-Un sup-assoc)
  then show ?case
    using Cons by fastforce
qed

lemma inconsistent-conjoin:
  defines ⟨C ≡ mk-finite-char (close {set G | G. ¬ ⊢ imply G ⊥})⟩
  assumes ⟨set G ∪ {p} ∉ C⟩
  shows ⊢ (¬ conjoin G p)
proof (rule ccontr)
  have ⟨consistency C⟩
    unfolding C-def using K-finite-consistency by auto
  have ⟨subset-closed C⟩
    unfolding C-def by (simp add: finite-char finite-char-closed)

  assume ⟨¬ ⊢ (¬ conjoin G p)⟩
  then have ⟨{conjoin G p} ∈ C⟩
    unfolding C-def using K-inconsistent-neg by blast
  then have ⟨set G ∪ {p} ∈ C⟩
    using conjoined-in-consistency ⟨consistency C⟩ ⟨subset-closed C⟩ by fastforce
  then show False
    using assms(2) by blast
qed

theorem K-in-maximal:
  defines ⟨C ≡ mk-finite-char (close {set G | G. ¬ ⊢ imply G ⊥})⟩
  assumes ⊢ p ⟨V ∈ C⟩ ⟨maximal V C⟩
  shows ⟨p ∈ V⟩
proof –
  have ⟨consistency C⟩
    unfolding C-def using K-finite-consistency by auto
  have ⟨subset-closed C⟩
    unfolding C-def by (simp add: finite-char finite-char-closed)

  have ⊢ (p ⟶ ¬ ¬ p)
    by (simp add: A1)
  then have ⊢ (¬ ¬ p)
    using ⊢ p R1 by blast

  show ?thesis

```

```

proof (rule ccontr)
  assume  $\langle p \notin V \rangle$ 
  then have  $\langle \neg p \in V \rangle$ 
    using at-least-one-in-maximal  $\langle \text{consistency } C \rangle \langle V \in C \rangle \langle \text{maximal } V C \rangle$  by
    blast
  then have  $\langle \text{set } [\neg p] \in C \rangle$ 
    using  $\langle V \in C \rangle \langle \text{subset-closed } C \rangle$  unfolding subset-closed-def by simp
  then have  $\langle \neg \vdash (\neg \neg p) \rangle$ 
    using C-def K-concrete-finite-consistency by fastforce
  then show False
    using  $\langle \vdash (\neg \neg p) \rangle$  by blast
qed

```

```

lemma exists-finite-inconsistent:
  fixes  $C :: \langle 'i \text{ fm set set} \rangle$ 
  assumes  $\langle \text{finite-char } C \rangle \langle V \cup \{\neg p\} \notin C \rangle$ 
  shows  $\langle \exists W. W \cup \{\neg p\} \subseteq V \cup \{\neg p\} \wedge (\neg p) \notin W \wedge \text{finite } W \wedge W \cup \{\neg p\} \notin C \rangle$ 
    using assms unfolding finite-char-def
    by (smt Un-insert-right Un-subset-iff finite-insert inf-sup-ord(4)
      mk-disjoint-insert sup-bot.comm-neutral sup-ge1)

```

```

theorem exists-maximal-superset:
  fixes  $C :: \langle ('i :: \text{countable}) \text{ fm set set} \rangle$ 
  assumes  $\langle \text{consistency } C \rangle \langle \text{finite-char } C \rangle \langle V \in C \rangle$ 
  obtains  $W$  where  $\langle V \subseteq W \rangle \langle W \in C \rangle \langle \text{maximal } W C \rangle$ 
proof –
  let  $?W = \langle \text{Extend } V C \text{ from-nat} \rangle$ 

  have  $\langle V \subseteq ?W \rangle$ 
    using Extend-subset by blast
  moreover have  $\langle ?W \in C \rangle$ 
    using assms Extend-in-C by blast
  moreover have  $\langle \text{maximal } ?W C \rangle$ 
    using assms Extend-maximal
    by (metis from-nat-to-nat)
  ultimately show ?thesis
    using that by blast
qed

```

```

type-synonym  $'i \text{ s-max} = \langle 'i \text{ fm set} \rangle$ 

```

```

abbreviation  $pi :: \langle 'i \text{ s-max} \Rightarrow id \Rightarrow bool \rangle$  where
   $\langle pi \ s \ i \equiv \text{Pro } i \in s \rangle$ 

```

```

abbreviation partition  $:: \langle 'i \text{ fm set} \Rightarrow 'i \Rightarrow 'i \text{ fm set} \rangle$  where
   $\langle \text{partition } V \ i \equiv \{p. K \ i \ p \in V\} \rangle$ 

```

**abbreviation**  $reach :: \langle 'i \text{ fm set set} \Rightarrow 'i \Rightarrow 'i \text{ s-max} \Rightarrow 'i \text{ s-max set} \rangle$  **where**  
 $\langle reach \ C \ i \ V \equiv \{ W. \text{ partition } V \ i \subseteq W \wedge W \in C \wedge \text{ maximal } W \ C \} \rangle$

**theorem** *model-existence*:

**fixes**  $p :: \langle ('i :: \text{countable}) \text{ fm} \rangle$

**defines**  $\langle C \equiv \text{mk-finite-char } (\text{close } \{ \text{set } G \mid G. \neg \vdash \text{ imply } G \perp \}) \rangle$

**assumes**  $\langle V \in C \rangle \langle \text{maximal } V \ C \rangle$

**shows**  $\langle (p \in V \longleftrightarrow \text{Kripke } pi \ (reach \ C), \ V \models p) \wedge$   
 $((\neg p) \in V \longleftrightarrow \text{Kripke } pi \ (reach \ C), \ V \models \neg p) \rangle$

**proof** –

**have**  $\langle \text{consistency } C \rangle$

**unfolding**  $C\text{-def}$  **using**  $K\text{-finite-consistency}$  **by**  $auto$

**have**  $\langle \text{finite-char } C \rangle$

**unfolding**  $C\text{-def}$  **by**  $(\text{simp add: finite-char})$

**have**  $\langle \text{subset-closed } C \rangle$

**unfolding**  $C\text{-def}$  **by**  $(\text{simp add: finite-char finite-char-closed})$

**show**  $?thesis$

**using**  $\langle V \in C \rangle \langle \text{maximal } V \ C \rangle$

**proof**  $(\text{induct } p \text{ arbitrary: } V)$

**case**  $FF$

**then show**  $?case$

**proof**  $(\text{intro conjI impI iffI})$

**assume**  $\langle \perp \in V \rangle$

**then have**  $False$

**using**  $\langle \text{consistency } C \rangle \langle V \in C \rangle$  **unfolding**  $\text{consistency-def}$  **by**  $blast$

**then show**  $\langle \text{Kripke } pi \ (reach \ C), \ V \models \perp \rangle ..$

**next**

**assume**  $\langle (\neg \perp) \in V \rangle$

**show**  $\langle \text{Kripke } pi \ (reach \ C), \ V \models \neg \perp \rangle$

**by**  $\text{simp}$

**next**

**assume**  $\langle \text{Kripke } pi \ (reach \ C), \ V \models \perp \rangle$

**then show**  $\langle \perp \in V \rangle$

**by**  $\text{simp}$

**next**

**assume**  $\langle \text{Kripke } pi \ (reach \ C), \ V \models \neg \perp \rangle$

**have**  $\langle \perp \notin V \rangle$

**using**  $\langle \text{consistency } C \rangle \langle V \in C \rangle$  **unfolding**  $\text{consistency-def}$

**by**  $blast$

**then show**  $\langle (\neg \perp) \in V \rangle$

**using**  $\text{at-least-one-in-maximal } FF \ \langle \text{consistency } C \rangle$

**by**  $blast$

**qed**

**next**

**case**  $(Pro \ i)$

**then show**  $?case$

**proof**  $(\text{intro conjI impI iffI})$

**assume**  $\langle Pro \ i \in V \rangle$

```

then have  $\langle pi\ V\ i \rangle$ 
  using  $\langle maximal\ V\ C \rangle$  by blast
then show  $\langle Kripke\ pi\ (reach\ C),\ V \models Pro\ i \rangle$ 
  by simp
next
assume  $\langle (\neg\ Pro\ i) \in V \rangle$ 
then have  $\langle Pro\ i \notin V \rangle$ 
  using  $\langle consistency\ C \rangle$   $\langle V \in C \rangle$  unfolding consistency-def by fast
then have  $\langle \neg\ (pi\ V\ i) \rangle$ 
  using  $\langle maximal\ V\ C \rangle$  by blast
then show  $\langle Kripke\ pi\ (reach\ C),\ V \models \neg\ Pro\ i \rangle$ 
  by simp
next
assume  $\langle Kripke\ pi\ (reach\ C),\ V \models Pro\ i \rangle$ 
then show  $\langle Pro\ i \in V \rangle$ 
  by simp
next
assume  $\langle Kripke\ pi\ (reach\ C),\ V \models \neg\ Pro\ i \rangle$ 
then have  $\langle Pro\ i \notin V \rangle$ 
  by simp
then show  $\langle (\neg\ Pro\ i) \in V \rangle$ 
  using at-least-one-in-maximal Pro  $\langle consistency\ C \rangle$ 
  by blast
qed
next
case  $\langle Dis\ p\ q \rangle$ 
have  $\langle (p \vee q) \in V \longrightarrow Kripke\ pi\ (reach\ C),\ V \models (p \vee q) \rangle$ 
proof
  assume  $\langle (p \vee q) \in V \rangle$ 
then have  $\langle V \cup \{p\} \in C \vee V \cup \{q\} \in C \rangle$ 
  using  $\langle consistency\ C \rangle$   $\langle V \in C \rangle$  unfolding consistency-def by simp
then consider  $\langle p \in V \rangle \mid \langle q \in V \rangle$ 
  using  $\langle maximal\ V\ C \rangle$  unfolding maximal-def by fast
then show  $\langle Kripke\ pi\ (reach\ C),\ V \models (p \vee q) \rangle$ 
  using Dis by cases simp-all
qed
moreover have  $\langle (\neg\ (p \vee q)) \in V \longrightarrow Kripke\ pi\ (reach\ C),\ V \models \neg\ (p \vee q) \rangle$ 
proof
  assume  $\langle (\neg\ (p \vee q)) \in V \rangle$ 
then have  $\langle V \cup \{\neg\ p,\ \neg\ q\} \in C \rangle$ 
  using  $\langle consistency\ C \rangle$   $\langle V \in C \rangle$  unfolding consistency-def by simp
then have  $\langle (\neg\ p) \in V \wedge (\neg\ q) \in V \rangle$ 
  using  $\langle maximal\ V\ C \rangle$  unfolding maximal-def by fast
then show  $\langle Kripke\ pi\ (reach\ C),\ V \models \neg\ (p \vee q) \rangle$ 
  using Dis by simp
qed
ultimately show ?case
  using at-least-one-in-maximal Dis  $\langle consistency\ C \rangle$ 
  by auto

```

```

next
case (Con p q)
have ⟨(p ∧ q) ∈ V ⟶ Kripke pi (reach C), V ⊨ (p ∧ q)⟩
proof
  assume ⟨(p ∧ q) ∈ V⟩
  then have ⟨V ∪ {p, q} ∈ C⟩
    using ⟨consistency C⟩ ⟨V ∈ C⟩ unfolding consistency-def by simp
  then have ⟨p ∈ V⟩ ⟨q ∈ V⟩
    using ⟨maximal V C⟩ unfolding maximal-def by fast+
  then show ⟨Kripke pi (reach C), V ⊨ (p ∧ q)⟩
    using Con by simp
qed
moreover have ⟨(¬ (p ∧ q)) ∈ V ⟶ Kripke pi (reach C), V ⊨ ¬ (p ∧ q)⟩
proof
  assume ⟨(¬ (p ∧ q)) ∈ V⟩
  then have ⟨V ∪ {¬ p} ∈ C ∨ V ∪ {¬ q} ∈ C⟩
    using ⟨consistency C⟩ ⟨V ∈ C⟩ unfolding consistency-def by simp
  then consider ⟨(¬ p) ∈ V⟩ | ⟨(¬ q) ∈ V⟩
    using ⟨maximal V C⟩ unfolding maximal-def by fast
  then show ⟨Kripke pi (reach C), V ⊨ ¬ (p ∧ q)⟩
    using Con by cases simp-all
qed
ultimately show ?case
  using at-least-one-in-maximal Con ⟨consistency C⟩
  by auto
next
case (Imp p q)
have ⟨(p ⟶ q) ∈ V ⟶ Kripke pi (reach C), V ⊨ (p ⟶ q)⟩
proof
  assume ⟨(p ⟶ q) ∈ V⟩
  then have ⟨V ∪ {¬ p} ∈ C ∨ V ∪ {q} ∈ C⟩
    using ⟨consistency C⟩ ⟨V ∈ C⟩ unfolding consistency-def by simp
  then consider ⟨(¬ p) ∈ V⟩ | ⟨q ∈ V⟩
    using ⟨maximal V C⟩ unfolding maximal-def by fast
  then show ⟨Kripke pi (reach C), V ⊨ (p ⟶ q)⟩
    using Imp by cases simp-all
qed
moreover have ⟨(¬ (p ⟶ q)) ∈ V ⟶ Kripke pi (reach C), V ⊨ ¬ (p ⟶
q)⟩
proof
  assume ⟨(¬ (p ⟶ q)) ∈ V⟩
  then have ⟨V ∪ {p, ¬ q} ∈ C⟩
    using ⟨consistency C⟩ ⟨V ∈ C⟩ unfolding consistency-def by simp
  then have ⟨p ∈ V ∧ (¬ q) ∈ V⟩
    using ⟨maximal V C⟩ unfolding maximal-def by fast
  then show ⟨Kripke pi (reach C), V ⊨ ¬ (p ⟶ q)⟩
    using Imp by simp
qed
ultimately show ?case

```



```

    using at-least-one-in-maximal Imp ⟨consistency C⟩
    by auto
next
case (K i p)
have ⟨K i p ∈ V ⟶ Kripke pi (reach C), V ⊨ K i p⟩
proof
  assume ⟨K i p ∈ V⟩
  then have ⟨∀ W ∈ reach C i V. p ∈ W ∧ W ∈ C ∧ maximal W C⟩
    using ⟨consistency C⟩ by blast
  then have ⟨∀ W ∈ reach C i V. Kripke pi (reach C), W ⊨ p⟩
    using K by simp
  then show ⟨Kripke pi (reach C), V ⊨ K i p⟩
    by simp
qed
moreover have ⟨(Kripke pi (reach C), V ⊨ K i p) ⟶ K i p ∈ V⟩
proof (intro allI impI)
  assume ⟨Kripke pi (reach C), V ⊨ K i p⟩

  have ⟨partition V i ∪ {¬ p} ∉ C⟩
  proof
    assume ⟨partition V i ∪ {¬ p} ∈ C⟩
    then obtain W where ⟨partition V i ∪ {¬ p} ⊆ W⟩ ⟨W ∈ C⟩ ⟨maximal
W C⟩
    using ⟨consistency C⟩ ⟨finite-char C⟩ exists-maximal-superset by blast
    then have ⟨Kripke pi (reach C), W ⊨ ¬ p⟩
      using K ⟨consistency C⟩ ⟨finite-char C⟩ by blast
    moreover have ⟨W ∈ reach C i V⟩
      using ⟨partition V i ∪ {¬ p} ⊆ W⟩ ⟨W ∈ C⟩ ⟨maximal W C⟩
      by simp
    ultimately have ⟨Kripke pi (reach C), V ⊨ ¬ K i p⟩
      by auto
    then show False
      using ⟨Kripke pi (reach C), V ⊨ K i p⟩
      by auto
  qed

  then obtain W where
    ⟨W ∪ {¬ p} ⊆ partition V i ∪ {¬ p}⟩ ⟨(¬ p) ∉ W⟩ ⟨finite W⟩ ⟨W ∪ {¬ p}
∉ C⟩
    using ⟨finite-char C⟩ exists-finite-inconsistent by force

  obtain L where ⟨set L = W⟩
    using ⟨finite W⟩ finite-list by blast

  then have ⟨⊢ (¬ conjoin L (¬ p))⟩
    using inconsistent-conjoin ⟨W ∪ {¬ p} ∉ C⟩ C-def by blast
  then have ⟨⊢ imply L p⟩
    using K-conjoin-imply by blast
  then have ⟨⊢ K i (imply L p)⟩

```

```

    using R2 by fast
  then have  $\langle \vdash \text{imply } (\text{map } (K \ i) \ L) \ (K \ i \ p) \rangle$ 
    using K-distrib-K-imp by fast
  then have  $\langle \text{imply } (\text{map } (K \ i) \ L) \ (K \ i \ p) \in V \rangle$ 
    using K-in-maximal K.premis(1, 2) unfolding C-def by blast
  then show  $\langle K \ i \ p \in V \rangle$ 
    using  $\langle \text{set } L = W \rangle \langle W \cup \{\neg p\} \subseteq \text{partition } V \ i \cup \{\neg p\} \rangle \langle \neg p \notin W \rangle$ 
  proof (induct L arbitrary: W)
    case Nil
    then show ?case
      by simp
  next
    case (Cons a L)
    then have  $\langle K \ i \ a \in V \rangle$ 
      by auto
    then have  $\langle \text{imply } (\text{map } (K \ i) \ L) \ (K \ i \ p) \in V \rangle$ 
      using Cons(2)  $\langle \text{consistency } C \rangle \langle V \in C \rangle \langle \text{maximal } V \ C \rangle \text{ consequent-in-maximal}$ 
  by auto
    then show ?case
      using Cons by auto
  qed
  qed
  moreover have  $\langle (Kripke \ pi \ (\text{reach } C), V \models \neg K \ i \ p) \longrightarrow (\neg K \ i \ p) \in V \rangle$ 
    using K.premis(1)  $\langle \text{consistency } C \rangle \langle \text{maximal } V \ C \rangle \text{ at-least-one-in-maximal}$ 
  calculation(1) by auto
  moreover have  $\langle (\neg K \ i \ p) \in V \longrightarrow Kripke \ pi \ (\text{reach } C), V \models \neg K \ i \ p \rangle$ 
    using K.premis(1)  $\langle \text{consistency } C \rangle \langle \text{maximal } V \ C \rangle \text{ calculation(2) exactly-one-in-maximal}$ 
  by auto
  ultimately show ?case
    by blast
  qed
  qed

```

## 9.1 Completeness

**lemma** *imply-completeness*:

**assumes** *valid*:  $\langle \forall (M :: ('i, ('i :: \text{countable}) \text{fm set}) \text{kripke}) \ s.$

*list-all*  $(\lambda q. M, s \models q) \ G \longrightarrow M, s \models p \rangle$

**shows**  $\langle \vdash \text{imply } G \ p \rangle$

**proof** (*rule* K-Boole, *rule* ccontr)

**assume**  $\langle \neg \vdash \text{imply } ((\neg p) \# G) \ \perp \rangle$

**let**  $?S = \langle \text{set } ((\neg p) \# G) \rangle$

**let**  $?C = \langle \text{mk-finite-char } (\text{close } \{\text{set } G \mid G. \neg \vdash \text{imply } G \ \perp\}) \rangle$

**let**  $?V = \langle \text{Extend } ?S \ ?C \ \text{from-nat} \rangle$

**let**  $?M = \langle Kripke \ pi \ (\text{reach } ?C) \rangle$

**have**  $\langle \text{consistency } ?C \rangle$

**by** (*simp add*: K-finite-consistency)

```

have ⟨subset-closed ?C⟩
  using finite-char finite-char-closed by blast
have ⟨finite-char ?C⟩
  using finite-char by blast

have ⟨?S ∈ ?C⟩
  using ⟨¬ ⊢ imply ((¬ p) # G) ⊥⟩ close-closed close-subset finite-char-subset
  by fast
then have ⟨?V ∈ ?C⟩
  using Extend-in-C K-finite-consistency ⟨finite-char ?C⟩ by blast

have ⟨∀ y :: 'i fm. ∃ n. y = from-nat n⟩
  by (metis from-nat-to-nat)
then have ⟨maximal ?V ?C⟩
  using Extend-maximal ⟨finite-char ?C⟩ by blast

{ fix x
  assume ⟨x ∈ ?S⟩
  then have ⟨x ∈ ?V⟩
    using Extend-subset by blast
  then have ⟨?M, ?V ⊨ x⟩
    using model-existence ⟨?V ∈ ?C⟩ ⟨maximal ?V ?C⟩ by blast }
then have ⟨?M, ?V ⊨ (¬ p)⟩ ⟨list-all (λp. ?M, ?V ⊨ p) G⟩
  unfolding list-all-def by fastforce+
then have ⟨?M, ?V ⊨ p⟩
  using valid by blast
then show False
  using ⟨?M, ?V ⊨ (¬ p)⟩ by simp
qed

theorem completeness:
  assumes ⟨∀ (M :: ('i :: countable, 'i fm set) kripke) s. M, s ⊨ p⟩
  shows ⟨⊢ p⟩
  using assms imply-completeness[where G=⟨[]⟩] by auto

```

## 10 Main Result

— System K is sound and complete

**abbreviation** ⟨valid p ≡ ∀ (M :: (nat, nat s-max) kripke) s. M, s ⊨ p⟩

**theorem** main: ⟨valid p ⟷ ⊢ p⟩

**proof**

```

assume ⟨valid p⟩
with completeness show ⟨⊢ p⟩
  by blast

```

**next**

```

assume ⟨⊢ p⟩
with soundness show ⟨valid p⟩

```

by (*intro allI*)  
**qed**  
  
**corollary**  $\langle \text{valid } p \longrightarrow M, s \models p \rangle$   
**proof**  
 assume  $\langle \text{valid } p \rangle$   
 then have  $\langle \vdash p \rangle$   
 unfolding *main* .  
 with *soundness* show  $\langle M, s \models p \rangle$  .  
**qed**

## 11 Acknowledgements

The definition of a consistency property, subset closure, finite character and maximally consistent sets is based on work by Berghofer, but has been adapted from first-order logic to epistemic logic.

- Stefan Berghofer: First-Order Logic According to Fitting. <https://www.isa-afp.org/entries/FOL-Fitting.shtml>

**end**

## References

- [1] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.