

Synthetic Completeness

Asta Halkjær From

February 6, 2026

Abstract

In this work, I provide an abstract framework for proving the completeness of a logical calculus using the synthetic method. The synthetic method is based on maximal consistent witnessed sets (MCSs). A set of formulas is consistent (with respect to the calculus) when we cannot derive a contradiction from it. It is maximally consistent when it contains every formula that is consistent with it. For logics where it is relevant, it is witnessed when it contains a witness for every existential formula. To prove completeness using these maximal consistent witnessed sets, we prove a truth lemma: every formula in an MCS has a satisfying model. Here, saturated sets provide a useful stepping stone. These can be seen as characterizations of the MCSs based on simple subformula conditions rather than via the calculus. We then prove that every saturated set gives rise to a satisfying model and that MCSs are saturated sets. Now, assume a valid formula cannot be derived. Then its negation must be consistent and therefore satisfiable. This contradicts validity and the original formula must be derivable.

To start, I build maximal consistent witnessed sets for any logic that satisfies a small set of assumptions. I do this using a transfinite version of Lindenbaum's lemma, which allows me to support languages of any cardinality. I then prove useful abstract results about derivations and refutations as they relate to MCSs. Finally, I show how saturated sets can be derived from the logic's semantics, outlining one way to prove the required truth lemma.

To demonstrate the versatility of the framework, I instantiate it with five different examples. The formalization contains soundness and completeness results for: a propositional tableau calculus, a propositional sequent calculus, an axiomatic system for modal logic, a labelled natural deduction system for hybrid logic and a natural deduction system for first-order logic. The tableau example uses custom Hintikka (downwards saturated) sets based on the calculus, but the other four examples derive their notion of saturation from the semantics in the style of the framework. The hybrid and first-order logic examples rely on witnessed MCSs. This places requirements on the cardinalities of their languages to ensure that there are enough witnesses available. In both cases, the type of witnesses must be infinite and have cardinality at least that of the type of propositional/predicate symbols.

Contents

Abstract	2
Contents	3
1 Maximal Consistent Sets	5
1.1 Utility	5
1.2 Base Locales	7
1.3 Ordinal Locale	8
1.3.1 Lindenbaum Extension	8
1.3.2 Consistency	9
1.3.3 Maximality	12
1.3.4 Witnessing	12
1.4 Locales for Universe Well-Order	13
1.5 Truth Lemma	14
2 Derivations	15
2.1 Derivations	15
2.2 MCSs and Explosion	15
2.3 MCSs and Derivability	16
2.4 Proof Rules	17
3 Refutations	23
3.1 Rearranging Refutations	23
3.2 MCSs and Refutability	23
4 Example: Propositional Tableau Calculus	25
4.1 Syntax	25
4.2 Semantics	25
4.3 Calculus	25
4.4 Soundness	25
4.5 Maximal Consistent Sets	26
4.6 Truth Lemma	26
4.7 Completeness	28
5 Example: Propositional Sequent Calculus	29
5.1 Syntax	29
5.2 Semantics	29
5.3 Calculus	29
5.4 Soundness	30
5.5 Maximal Consistent Sets	30

5.6	Truth Lemma	31
5.7	Completeness	31
6	Example: Modal Logic	33
6.1	Syntax	33
6.2	Semantics	33
6.3	Calculus	33
6.4	Soundness	34
6.5	Admissible rules	34
6.6	Maximal Consistent Sets	36
6.7	Truth Lemma	37
6.8	Completeness	39
7	Example: Hybrid Logic	40
7.1	Syntax	40
7.2	Semantics	40
7.3	Calculus	41
7.4	Soundness	41
7.5	Admissible Rules	42
7.6	Maximal Consistent Sets	44
7.7	Nominals	47
7.8	Truth Lemma	48
7.9	Cardinalities	50
7.10	Completeness	52
8	Example: First-Order Logic	54
8.1	Syntax	54
8.2	Semantics	54
8.3	Operations	54
8.4	Calculus	56
	8.4.1 Weakening	56
8.5	Soundness	59
8.6	Admissible Rules	59
8.7	Maximal Consistent Sets	60
8.8	Truth Lemma	62
8.9	Cardinalities	62
8.10	Completeness	64
	Bibliography	67

Chapter 1

Maximal Consistent Sets

theory *Maximal-Consistent-Sets* **imports** *HOL-Cardinals.Cardinal-Order-Relation* **begin**

1.1 Utility

lemma *Set-Diff-Un*: $\langle X - (Y \cup Z) = X - Y - Z \rangle$
by *blast*

lemma *infinite-Diff-fin-Un*: $\langle \text{infinite } (X - Y) \implies \text{finite } Z \implies \text{infinite } (X - (Z \cup Y)) \rangle$
by (*simp add: Set-Diff-Un Un-commute*)

lemma *infinite-Diff-subset*: $\langle \text{infinite } (X - A) \implies B \subseteq A \implies \text{infinite } (X - B) \rangle$
by (*meson Diff-cancel Diff-eq-empty-iff Diff-mono infinite-super*)

lemma *finite-bound*:
fixes $X :: \langle 'a :: \text{size} \rangle \text{ set}$
assumes $\langle \text{finite } X \rangle \langle X \neq \{\} \rangle$
shows $\langle \exists x \in X. \forall y \in X. \text{size } y \leq \text{size } x \rangle$
using *assms* **by** (*induct X rule: finite-induct*) *force+*

lemma *infinite-UNIV-size*:
fixes $f :: \langle 'a :: \text{size} \rangle \Rightarrow 'a$
assumes $\langle \bigwedge x. \text{size } x < \text{size } (f x) \rangle$
shows $\langle \text{infinite } (\text{UNIV} :: 'a \text{ set}) \rangle$

proof
assume $\langle \text{finite } (\text{UNIV} :: 'a \text{ set}) \rangle$
then obtain $x :: 'a$ **where** $\langle \forall y :: 'a. \text{size } y \leq \text{size } x \rangle$
using *finite-bound* **by** *fastforce*
moreover have $\langle \text{size } x < \text{size } (f x) \rangle$
using *assms* .
ultimately show *False*
using *leD* **by** *blast*
qed

context *wo-rel* **begin**

lemma *underS-bound*: $\langle a \in \text{underS } c \implies b \in \text{underS } c \implies a \in \text{under } b \vee b \in \text{under } a \rangle$
by (*meson BNF-Least-Fixpoint.underS-Field REFL Reft-under-in in-mono under-ofilter ofilter-linord*)

lemma *finite-underS-bound*:
assumes $\langle \text{finite } X \rangle \langle X \subseteq \text{underS } c \rangle \langle X \neq \{\} \rangle$

shows $\langle \exists a \in X. \forall b \in X. b \in \text{under } a \rangle$
using *assms*
proof (*induct X rule: finite-induct*)
case (*insert x F*)
then show *?case*
proof (*cases* $\langle F = \{\} \rangle$)
case *True*
then show *?thesis*
using *insert underS-bound by fast*
next
case *False*
then show *?thesis*
using *insert underS-bound by (metis TRANS insert-absorb insert-iff insert-subset under-trans)*
qed
qed *simp*

lemma *finite-bound-under:*
assumes $\langle \text{finite } p \rangle \langle p \subseteq (\bigcup a \in \text{Field } r. f a) \rangle$
shows $\langle \exists b. p \subseteq (\bigcup a \in \text{under } b. f a) \rangle$
using *assms*
proof (*induct rule: finite-induct*)
case (*insert x p*)
then obtain *b* **where** $\langle p \subseteq (\bigcup a \in \text{under } b. f a) \rangle$
by *fast*
moreover obtain *b'* **where** $\langle x \in f b' \rangle \langle b' \in \text{Field } r \rangle$
using *insert(4) by blast*
then have $\langle x \in (\bigcup a \in \text{under } b'. f a) \rangle$
using *REFL Reft-under-in by fast*
ultimately have $\langle \{x\} \cup p \subseteq (\bigcup a \in \text{under } b. f a) \cup (\bigcup a \in \text{under } b'. f a) \rangle$
by *fast*
then show *?case*
by (*metis SUP-union Un-commute insert-is-Un sup.absorb-iff2 ofilter-linord under-ofilter*)
qed *simp*

lemma *underS-trans:* $\langle a \in \text{underS } b \implies b \in \text{underS } c \implies a \in \text{underS } c \rangle$
by (*meson ANTISYM TRANS underS-underS-trans*)

end

lemma *card-of-infinite-smaller-Union:*
assumes $\langle \forall x. |f x| < o |X| \rangle \langle \text{infinite } X \rangle$
shows $\langle |\bigcup x \in X. f x| \leq o |X| \rangle$
using *assms by (metis (full-types) Field-card-of card-of-UNION-ordLeq-infinite card-of-well-order-on ordLeq-iff-ordLess-or-ordIso ordLess-or-ordLeq)*

lemma *card-of-params-marker-lists:*
assumes $\langle \text{infinite } (UNIV :: 'i \text{ set}) \rangle \langle |UNIV :: 'm \text{ set}| \leq o |UNIV :: \text{nat set}| \rangle$
shows $\langle |UNIV :: ('i + 'm \times \text{nat}) \text{ list set}| \leq o |UNIV :: 'i \text{ set}| \rangle$
proof –
have $\langle (UNIV :: 'm \text{ set}) \neq \{\} \rangle$
by *simp*
then have $\langle |UNIV :: 'm \text{ set}| * c |UNIV :: \text{nat set}| \leq o |UNIV :: \text{nat set}| \rangle$
using *assms(2) by (simp add: cfinite-def cprod-cfinite-bound ordLess-imp-ordLeq)*
then have $\langle |UNIV :: ('m \times \text{nat}) \text{ set}| \leq o |UNIV :: \text{nat set}| \rangle$
unfolding *cprod-def by simp*
moreover have $\langle |UNIV :: \text{nat set}| \leq o |UNIV :: 'i \text{ set}| \rangle$

```

    using assms infinite-iff-card-of-nat by blast
  ultimately have  $\langle |UNIV :: ('m \times nat) set| \leq_o |UNIV :: 'i set| \rangle$ 
    using ordLeq-transitive by blast
  moreover have  $\langle Cinfinit\ e\ |UNIV :: 'i set| \rangle$ 
    using assms by (simp add: cinfinit\ e-def)
  ultimately have  $\langle |UNIV :: 'i set| + c\ |UNIV :: ('m \times nat) set| =_o |UNIV :: 'i set| \rangle$ 
    using csum-absorb1 by blast
  then have  $\langle |UNIV :: ('i + 'm \times nat) set| =_o |UNIV :: 'i set| \rangle$ 
    unfolding csum-def by simp
  then have  $\langle |UNIV :: ('i + 'm \times nat) set| \leq_o |UNIV :: 'i set| \rangle$ 
    using ordIso-iff-ordLeq by blast
  moreover have  $\langle infinite\ (UNIV :: ('i + 'm \times nat) set) \rangle$ 
    using assms by simp
  then have  $\langle |UNIV :: ('i + 'm \times nat) list\ set| =_o |UNIV :: ('i + 'm \times nat) set| \rangle$ 
    by (metis card-of-lists-infinite lists-UNIV)
  ultimately have  $\langle |UNIV :: ('i + 'm \times nat) list\ set| \leq_o |UNIV :: 'i set| \rangle$ 
    using ordIso-ordLeq-trans by blast
  then show ?thesis
    using ordLeq-transitive by blast
qed

```

1.2 Base Locales

```

locale MCS-Base =
  fixes consistent ::  $\langle 'a\ set \Rightarrow bool \rangle$ 
  assumes consistent-hereditary:  $\langle \bigwedge S\ S'.\ consistent\ S \Longrightarrow S' \subseteq S \Longrightarrow consistent\ S' \rangle$ 
  and inconsistent-finite:  $\langle \bigwedge S.\ \neg\ consistent\ S \Longrightarrow \exists S' \subseteq S.\ finite\ S' \wedge \neg\ consistent\ S' \rangle$ 
begin

```

```

definition maximal ::  $\langle 'a\ set \Rightarrow bool \rangle$  where
   $\langle maximal\ S \equiv \forall p.\ consistent\ (\{p\} \cup S) \longrightarrow p \in S \rangle$ 

```

end

```

locale MCS-Witness = MCS-Base consistent
  for consistent ::  $\langle 'a\ set \Rightarrow bool \rangle$  +
  fixes witness ::  $\langle 'a \Rightarrow 'a\ set \Rightarrow 'a\ set \rangle$ 
  and params ::  $\langle 'a \Rightarrow 'i\ set \rangle$ 
  assumes finite-params:  $\langle \bigwedge p.\ finite\ (params\ p) \rangle$ 
  and finite-witness-params:  $\langle \bigwedge p\ S.\ finite\ (\bigcup q \in witness\ p\ S.\ params\ q) \rangle$ 
  and consistent-witness:  $\langle \bigwedge p\ S.\ consistent\ (\{p\} \cup S) \Longrightarrow infinite\ (UNIV - (\bigcup q \in S.\ params\ q)) \Longrightarrow consistent\ (\{p\} \cup S \cup witness\ p\ S) \rangle$ 
begin

```

```

definition witnessed ::  $\langle 'a\ set \Rightarrow bool \rangle$  where
   $\langle witnessed\ S \equiv \forall p \in S.\ \exists S'.\ witness\ p\ S' \subseteq S \rangle$ 

```

```

abbreviation MCS ::  $\langle 'a\ set \Rightarrow bool \rangle$  where
   $\langle MCS\ S \equiv consistent\ S \wedge maximal\ S \wedge witnessed\ S \rangle$ 

```

end

```

locale MCS-No-Witness = MCS-Base consistent for consistent ::  $\langle 'a\ set \Rightarrow bool \rangle$ 

```

sublocale $MCS\text{-}No\text{-}Witness \subseteq MCS\text{-}Witness$ *consistent* $\langle \lambda\text{-} \cdot, \{\} \rangle \langle \lambda\text{-}, \{\} \rangle$
proof *qed simp-all*

1.3 Ordinal Locale

locale $MCS\text{-}Lim\text{-}Ord = MCS\text{-}Witness$ *consistent witness params*
for *consistent* :: $\langle 'a\ set \Rightarrow bool \rangle$
and *witness* :: $\langle 'a \Rightarrow 'a\ set \Rightarrow 'a\ set \rangle$
and *params* :: $\langle 'a \Rightarrow 'i\ set \rangle +$
fixes $r :: \langle 'a\ rel \rangle$
assumes $Cinfinite\text{-}r: \langle Cinfinite\ r \rangle$
begin

lemma $WELL: \langle Well\text{-}order\ r \rangle$
using $Cinfinite\text{-}r$ **by** *simp*

lemma $wo\text{-}rel\text{-}r: \langle wo\text{-}rel\ r \rangle$
by (*simp add: WELL wo-rel.intro*)

lemma $isLimOrd\text{-}r: \langle isLimOrd\ r \rangle$
using $Cinfinite\text{-}r$ *card-order-infinite-isLimOrd cinfinite-def* **by** *blast*

lemma $nonempty\text{-}Field\text{-}r: \langle Field\ r \neq \{\} \rangle$
using $Cinfinite\text{-}r$ *cinfinite-def infinite-imp-nonempty* **by** *blast*

1.3.1 Lindenbaum Extension

abbreviation $paramss :: \langle 'a\ set \Rightarrow 'i\ set \rangle$ **where**
 $\langle paramss\ S \equiv \bigcup p \in S. params\ p \rangle$

definition $extendS :: \langle 'a \Rightarrow 'a\ set \Rightarrow 'a\ set \rangle$ **where**
 $\langle extendS\ a\ prev \equiv \text{if consistent } (\{a\} \cup prev) \text{ then } \{a\} \cup prev \cup witness\ a\ prev \text{ else } prev \rangle$

definition $extendL :: \langle ('a \Rightarrow 'a\ set) \Rightarrow 'a \Rightarrow 'a\ set \rangle$ **where**
 $\langle extendL\ rec\ a \equiv \bigcup b \in underS\ r\ a. rec\ b \rangle$

definition $extend :: \langle 'a\ set \Rightarrow 'a \Rightarrow 'a\ set \rangle$ **where**
 $\langle extend\ S\ a \equiv wrecZSL\ r\ S\ extendS\ extendL\ a \rangle$

lemma $adm\text{-}woL\text{-}extendL: \langle adm\text{-}woL\ r\ extendL \rangle$
unfolding $extendL\text{-}def\ wo\text{-}rel.adm\text{-}woL\text{-}def[OF\ wo\text{-}rel\text{-}r]$ **by** *blast*

definition $Extend :: \langle 'a\ set \Rightarrow 'a\ set \rangle$ **where**
 $\langle Extend\ S \equiv \bigcup a \in Field\ r. extend\ S\ a \rangle$

lemma $extend\text{-}subset: \langle a \in Field\ r \Longrightarrow S \subseteq extend\ S\ a \rangle$

proof (*induct a rule: wo-rel.well-order-inductZSL[OF wo-rel-r]*)

case 1

then show *?case*

unfolding $extend\text{-}def\ wo\text{-}rel.wrecZSL\text{-}zero[OF\ wo\text{-}rel\text{-}r\ adm\text{-}woL\text{-}extendL]$

by *simp*

next

case (2 *i*)

moreover from this have $\langle i \in Field\ r \rangle$

by (*meson FieldI1 wo-rel.succ-in wo-rel-r*)

ultimately show *?case*
unfolding *extend-def extendS-def wo-rel.worecZSL-succ[OF wo-rel-r adm-woL-extendL 2(1)]*
by *auto*
next
case (3 *i*)
then show *?case*
unfolding *extend-def extendL-def wo-rel.worecZSL-isLim[OF wo-rel-r adm-woL-extendL 3(1-2)]*
using *wo-rel.zero-in-Field[OF wo-rel-r] wo-rel.zero-smallest[OF wo-rel-r]*
by (*metis SUP-upper2 emptyE underS-I*)
qed

lemma *Extend-subset*: $\langle S \subseteq \text{Extend } S \rangle$
unfolding *Extend-def* **using** *extend-subset nonempty-Field-r* **by** *fast*

lemma *extend-underS*: $\langle b \in \text{underS } r \ a \implies \text{extend } S \ b \subseteq \text{extend } S \ a \rangle$
proof (*induct a rule: wo-rel.well-order-inductZSL[OF wo-rel-r]*)
case 1
then show *?case*
unfolding *extend-def* **using** *wo-rel.underS-zero[OF wo-rel-r]* **by** *fast*
next
case (2 *i*)
moreover from *this* **have** $\langle b = i \vee b \in \text{underS } r \ i \rangle$
by (*metis wo-rel.less-succ[OF wo-rel-r] underS-E underS-I*)
ultimately show *?case*
unfolding *extend-def extendS-def wo-rel.worecZSL-succ[OF wo-rel-r adm-woL-extendL 2(1)]* **by** *auto*
next
case (3 *i*)
then show *?case*
unfolding *extend-def extendL-def wo-rel.worecZSL-isLim[OF wo-rel-r adm-woL-extendL 3(1-2)]*
by *blast*
qed

lemma *extend-under*: $\langle b \in \text{under } r \ a \implies \text{extend } S \ b \subseteq \text{extend } S \ a \rangle$
using *extend-underS wo-rel.supr-greater[OF wo-rel-r] wo-rel.supr-under[OF wo-rel-r]*
by (*metis emptyE in-Above-under set-eq-subset underS-I under-empty*)

1.3.2 Consistency

lemma *params-origin*:
assumes $\langle x \in \text{paramss } (\text{extend } S \ a) \rangle$
shows $\langle x \in \text{paramss } S \vee (\exists b \in \text{underS } r \ a. x \in \text{paramss } (\{b\} \cup \text{witness } b \ (\text{extend } S \ b))) \rangle$
using *assms*
proof (*induct a rule: wo-rel.well-order-inductZSL[OF wo-rel-r]*)
case 1
then show *?case*
unfolding *extend-def wo-rel.worecZSL-zero[OF wo-rel-r adm-woL-extendL]*
by *blast*
next
case (2 *i*)
then consider (*here*) $\langle x \in \text{paramss } (\{i\} \cup \text{witness } i \ (\text{extend } S \ i)) \rangle$ | (*there*) $\langle x \in \text{paramss } (\text{extend } S \ i) \rangle$
using *wo-rel.worecZSL-succ[OF wo-rel-r adm-woL-extendL 2(1)] extendS-def extend-def*
by (*auto split: if-splits*)
then show *?case*
proof *cases*
case *here*

```

moreover have  $\langle i \in \text{Field } r \rangle$ 
  by (meson WELL 2(1) well-order-on-domain wo-rel.succ-in-diff[OF wo-rel-r])
ultimately show ?thesis
  using 2(1) by (metis Reft-under-in wo-rel.underS-succ[OF wo-rel-r] wo-rel.REFL[OF wo-rel-r])
next
  case there
  then show ?thesis
    using 2 by (metis in-mono underS-subset-under wo-rel.underS-succ[OF wo-rel-r])
  next
  qed
next
  case (3 i)
  then obtain j where  $\langle j \in \text{underS } r \ i \rangle \langle x \in \text{paramss } (\text{extend } S \ j) \rangle$ 
    unfolding extend-def extendL-def wo-rel.worecZSL-isLim[OF wo-rel-r adm-woL-extendL 3(1-2)]
    by blast
  then show ?case
    using 3 wo-rel.underS-trans[OF wo-rel-r, of - j i] by meson
qed

lemma consistent-extend:
  assumes  $\langle \text{consistent } S \rangle \langle r \leq o \mid \text{UNIV} - \text{paramss } S \mid \rangle$ 
  shows  $\langle \text{consistent } (\text{extend } S \ a) \rangle$ 
  using assms(1)
proof (induct a rule: wo-rel.well-order-inductZSL[OF wo-rel-r])
  case 1
  then show ?case
    unfolding extend-def wo-rel.worecZSL-zero[OF wo-rel-r adm-woL-extendL]
    by blast
next
  case (2 i)
  then have  $\langle i \in \text{Field } r \rangle$ 
    by (meson WELL well-order-on-domain wo-rel.succ-in-diff[OF wo-rel-r])
  then have  $\langle \mid \text{underS } r \ i \mid < o \ r \rangle$ 
    using card-of-underS by (simp add: Cinfinit-r)
  let ?paramss =  $\langle \lambda k. \text{paramss } (\{k\} \cup \text{witness } k \ (\text{extend } S \ k)) \rangle$ 
  let ?X =  $\langle \bigcup k \in \text{underS } r \ i. \ ?\text{paramss } k \rangle$ 
  have  $\langle \mid ?X \mid < o \ r \rangle$ 
  proof (cases  $\langle \text{finite } (\text{underS } r \ i) \rangle$ )
    case True
      then have  $\langle \text{finite } ?X \rangle$ 
        by (simp add: finite-params finite-witness-params)
      then show ?thesis
        using Cinfinit-r assms(2) unfolding cinfinit-def by (simp add: finite-ordLess-infinite)
    next
      case False
        moreover have  $\langle \forall k. \text{finite } (?paramss \ k) \rangle$ 
          by (simp add: finite-params finite-witness-params)
        then have  $\langle \forall k. \mid ?paramss \ k \mid < o \ \mid \text{underS } r \ i \mid \rangle$ 
          using False by simp
        ultimately have  $\langle \mid ?X \mid \leq o \ \mid \text{underS } r \ i \mid \rangle$ 
          using card-of-infinite-smaller-Union by fast
        then show ?thesis
          using  $\ast$  ordLeq-ordLess-trans by blast
    qed
  then have  $\langle \mid ?X \mid < o \ \mid \text{UNIV} - \text{paramss } S \mid \rangle$ 
    using assms(2) ordLess-ordLeq-trans by blast

```

moreover have $\langle \text{infinite } (UNIV - \text{paramss } S) \rangle$
using *assms(2) Cinfinit-r unfolding cinfinit-def* **by** (*metis Field-card-of ordLeq-finite-Field*)
ultimately have $\langle |UNIV - \text{paramss } S - ?X| = o \mid UNIV - \text{paramss } S \rangle$
using *card-of-Un-diff-infinite* **by** *blast*
moreover from this have $\langle \text{infinite } (UNIV - \text{paramss } S - ?X) \rangle$
using $\langle \text{infinite } (UNIV - \text{paramss } S) \rangle$ *card-of-ordIso-finite* **by** *blast*
moreover have $\langle \bigwedge a. a \in \text{paramss } (\text{extend } S \ i) \implies a \in \text{paramss } S \vee a \in ?X \rangle$
using *params-origin* **by** *simp*
then have $\langle \text{paramss } (\text{extend } S \ i) \subseteq \text{paramss } S \cup ?X \rangle$
by *fast*
ultimately have $\langle \text{infinite } (UNIV - \text{paramss } (\text{extend } S \ i)) \rangle$
using *infinite-Diff-subset* **by** (*metis (no-types, lifting) Set-Diff-Un*)
with 2 show *?case*
unfolding *extend-def extendS-def wo-rel.worecZSL-succ[OF wo-rel-r adm-woL-extendL 2(1)]*
using *consistent-witness* **by** *simp*
next
case (*3 i*)
show *?case*
proof (*rule ccontr*)
assume $\langle \neg \text{consistent } (\text{extend } S \ i) \rangle$
then obtain *S'* **where** $S': \langle \text{finite } S' \rangle \langle S' \subseteq (\bigcup a \in \text{under } S \ r \ i. \text{extend } S \ a) \rangle \langle \neg \text{consistent } S' \rangle$
unfolding *extend-def extendL-def wo-rel.worecZSL-isLim[OF wo-rel-r adm-woL-extendL 3(1-2)]*
using *inconsistent-finite* **by** *auto*
then obtain *as* **where** $as: \langle S' \subseteq (\bigcup a \in as. \text{extend } S \ a) \rangle \langle as \subseteq \text{under } S \ r \ i \rangle \langle \text{finite } as \rangle$
by (*metis finite-subset-Union finite-subset-image*)
moreover have $\langle as \neq \{\} \rangle$
using *S'(3) assms calculation(1) consistent-hereditary* **by** *auto*
ultimately obtain *j* **where** $\langle \forall a \in as. a \in \text{under } r \ j \rangle \langle j \in \text{under } S \ r \ i \rangle$
using *wo-rel.finite-underS-bound wo-rel-r as* **by** (*meson subset-iff*)
then have $\langle \forall a \in as. \text{extend } S \ a \subseteq \text{extend } S \ j \rangle$
using *extend-under* **by** *fast*
then have $\langle S' \subseteq \text{extend } S \ j \rangle$
using *S' as(1)* **by** *blast*
then show *False*
using *3(3-) ¬ consistent S' consistent-hereditary j ∈ underS r i*
by (*meson BNF-Least-Fixpoint.underS-Field*)
qed
qed

lemma *consistent-Extend:*

assumes $\langle \text{consistent } S \rangle \langle r \leq o \mid UNIV - \text{paramss } S \rangle$
shows $\langle \text{consistent } (\text{Extend } S) \rangle$
unfolding *Extend-def*
proof (*rule ccontr*)
assume $\langle \neg \text{consistent } (\bigcup a \in \text{Field } r. \text{extend } S \ a) \rangle$
then obtain *S'* **where** $\langle \text{finite } S' \rangle \langle S' \subseteq (\bigcup a \in \text{Field } r. \text{extend } S \ a) \rangle \langle \neg \text{consistent } S' \rangle$
using *inconsistent-finite* **by** *metis*
then obtain *b* **where** $\langle S' \subseteq (\bigcup a \in \text{under } r \ b. \text{extend } S \ a) \rangle \langle b \in \text{Field } r \rangle$
using *wo-rel.finite-bound-under[OF wo-rel-r] assms consistent-hereditary*
by (*metis Sup-empty emptyE image-empty subsetI under-empty*)
then have $\langle S' \subseteq \text{extend } S \ b \rangle$
using *extend-under* **by** *fast*
moreover have $\langle \text{consistent } (\text{extend } S \ b) \rangle$
using *assms consistent-extend b ∈ Field r* **by** *blast*
ultimately show *False*
using $\langle \neg \text{consistent } S' \rangle$ *consistent-hereditary* **by** *blast*

qed

lemma *Extend-bound*: $\langle a \in \text{Field } r \implies \text{extend } S \ a \subseteq \text{Extend } S \rangle$
unfolding *Extend-def* **by** *blast*

1.3.3 Maximality

definition *maximal'* :: $\langle 'a \text{ set} \implies \text{bool} \rangle$ **where**
 $\langle \text{maximal}' \ S \equiv \forall p \in \text{Field } r. \text{consistent } (\{p\} \cup S) \longrightarrow p \in S \rangle$

lemma *maximal'-Extend*: $\langle \text{maximal}' \ (\text{Extend } S) \rangle$
unfolding *maximal'-def*

proof *safe*

fix *p*

assume *: $\langle p \in \text{Field } r \rangle \langle \text{consistent } (\{p\} \cup \text{Extend } S) \rangle$

then have $\langle \{p\} \cup \text{extend } S \ p \subseteq \{p\} \cup \text{Extend } S \rangle$

unfolding *Extend-def* **by** *blast*

then have **: $\langle \text{consistent } (\{p\} \cup \text{extend } S \ p) \rangle$

using * *consistent-hereditary* **by** *blast*

moreover have *succ*: $\langle \text{above } S \ r \ p \neq \{\} \rangle$

using * *isLimOrd-r wo-rel.isLimOrd-aboveS[OF wo-rel-r]* **by** *blast*

then have $\langle \text{succ } r \ p \in \text{Field } r \rangle$

using *wo-rel.succ-in-Field[OF wo-rel-r]* **by** *blast*

moreover have $\langle p \in \text{extend } S \ (\text{succ } r \ p) \rangle$

using ** **unfolding** *extend-def extendsS-def wo-rel.worecZSL-succ[OF wo-rel-r adm-woL-extendL*

succ]

by *simp*

ultimately show $\langle p \in \text{Extend } S \rangle$

using *Extend-bound* **by** *fast*

qed

1.3.4 Witnessing

definition *witnessed'* :: $\langle 'a \text{ set} \implies \text{bool} \rangle$ **where**
 $\langle \text{witnessed}' \ S \equiv \forall p \in \text{Field } r. p \in S \longrightarrow (\exists S'. \text{witness } p \ S' \subseteq S) \rangle$

lemma *witnessed'-Extend*:

assumes $\langle \text{consistent } (\text{Extend } S) \rangle$

shows $\langle \text{witnessed}' \ (\text{Extend } S) \rangle$

unfolding *witnessed'-def*

proof *safe*

fix *p*

assume *: $\langle p \in \text{Field } r \rangle \langle p \in \text{Extend } S \rangle$

then have $\langle \text{extend } S \ p \subseteq \text{Extend } S \rangle$

unfolding *Extend-def* **by** *blast*

then have $\langle \text{consistent } (\{p\} \cup \text{extend } S \ p) \rangle$

using *assms(1) * consistent-hereditary* **by** *auto*

moreover have *succ*: $\langle \text{above } S \ r \ p \neq \{\} \rangle$

using * *isLimOrd-r wo-rel.isLimOrd-aboveS wo-rel-r* **by** *fast*

then have $\langle \text{succ } r \ p \in \text{Field } r \rangle$

using *wo-rel-r* **by** (*simp add: wo-rel.succ-in-Field*)

ultimately have $\langle \text{extend } S \ (\text{succ } r \ p) = \{p\} \cup \text{extend } S \ p \cup \text{witness } p \ (\text{extend } S \ p) \rangle$

unfolding *extend-def extendsS-def wo-rel.worecZSL-succ[OF wo-rel-r adm-woL-extendL succ]*

by *simp*

moreover have $\langle \text{extend } S \ (\text{succ } r \ p) \subseteq \text{Extend } S \rangle$

unfolding *Extend-def* **using** $\langle \text{succ } r \ p \in \text{Field } r \rangle$ **by** *blast*

ultimately show $\langle \exists a. \text{witness } p \ a \subseteq \text{Extend } S \rangle$
 by *fast*
 qed
 end

1.4 Locales for Universe Well-Order

locale *MCS-Witness-UNIV* = *MCS-Witness consistent witness params*
 for *consistent* :: $\langle 'a \text{ set} \Rightarrow \text{bool} \rangle$
 and *witness* :: $\langle 'a \Rightarrow 'a \text{ set} \Rightarrow 'a \text{ set} \rangle$
 and *params* :: $\langle 'a \Rightarrow 'i \text{ set} \rangle +$
 assumes *infinite-UNIV*: $\langle \text{infinite } (UNIV :: 'a \text{ set}) \rangle$

sublocale *MCS-Witness-UNIV* \subseteq *MCS-Lim-Ord consistent witness params* $\langle |UNIV| \rangle$
 proof
 show $\langle \text{Cinfinite } |UNIV :: 'a \text{ set}| \rangle$
 unfolding *cinfinite-def* using *infinite-UNIV* by *simp*
 qed

context *MCS-Witness-UNIV* begin

lemma *maximal-maximal'*: $\langle \text{maximal } S \longleftrightarrow \text{maximal}' S \rangle$
 unfolding *maximal-def* *maximal'-def* by *simp*

lemma *maximal-Extend*: $\langle \text{maximal } (\text{Extend } S) \rangle$
 using *maximal'-Extend* *maximal-maximal'* by *fast*

lemma *witnessed-witnessed'*: $\langle \text{witnessed } S \longleftrightarrow \text{witnessed}' S \rangle$
 unfolding *witnessed-def* *witnessed'-def* by *auto*

lemma *witnessed-Extend*:
 assumes $\langle \text{consistent } (\text{Extend } S) \rangle$
 shows $\langle \text{witnessed } (\text{Extend } S) \rangle$
 using *assms* *witnessed'-Extend* *witnessed-witnessed'* by *blast*

theorem *MCS-Extend*:
 assumes $\langle \text{consistent } S \rangle \langle |UNIV :: 'a \text{ set}| \leq o \ |UNIV - \text{params } S| \rangle$
 shows $\langle \text{MCS } (\text{Extend } S) \rangle$
 using *assms* *consistent-Extend* *maximal-Extend* *witnessed-Extend* by *blast*

end

locale *MCS-No-Witness-UNIV* = *MCS-No-Witness consistent*
 for *consistent* :: $\langle 'a \text{ set} \Rightarrow \text{bool} \rangle +$
 assumes *infinite-UNIV'* [*simp*]: $\langle \text{infinite } (UNIV :: 'a \text{ set}) \rangle$

sublocale *MCS-No-Witness-UNIV* \subseteq *MCS-Witness-UNIV consistent* $\langle \lambda -. \{\} \rangle \langle \lambda -. \{\} \rangle$
 proof qed *simp*

context *MCS-No-Witness-UNIV*
 begin

theorem *MCS-Extend'*:
 assumes $\langle \text{consistent } S \rangle$

shows $\langle \text{MCS } (\text{Extend } S) \rangle$
 unfolding *witnessed-def using assms consistent-Extend maximal-Extend*
 by (*metis Diff-empty UN-constant card-of-UNIV empty-subsetI*)

end

1.5 Truth Lemma

locale *Truth-Base* =

fixes *semics* :: $\langle 'model \Rightarrow ('model \Rightarrow 'fm \Rightarrow bool) \Rightarrow 'fm \Rightarrow bool \rangle$ ($\langle (- \llbracket - \rrbracket -) \rangle$ [55, 0, 55] 55)
 and *semantics* :: $\langle 'model \Rightarrow 'fm \Rightarrow bool \rangle$ (**infix** $\langle \models \rangle$ 50)
 and \mathcal{M} :: $\langle 'a \text{ set} \Rightarrow 'model \text{ set} \rangle$
 and \mathcal{R} :: $\langle 'a \text{ set} \Rightarrow 'model \Rightarrow 'fm \Rightarrow bool \rangle$
 assumes *semics-semantics*: $\langle M \models p \longleftrightarrow M \llbracket (\models) \rrbracket p \rangle$

begin

abbreviation *saturated* :: $\langle 'a \text{ set} \Rightarrow bool \rangle$ where

$\langle \text{saturated } S \equiv \forall p. \forall M \in \mathcal{M}(S). M \llbracket \mathcal{R}(S) \rrbracket p \longleftrightarrow \mathcal{R}(S) M p \rangle$

end

locale *Truth-Witness* = *Truth-Base semics semantics* $\mathcal{M} \mathcal{R}$ + *MCS-Witness consistent witness params*

for *semics* :: $\langle 'model \Rightarrow ('model \Rightarrow 'fm \Rightarrow bool) \Rightarrow 'fm \Rightarrow bool \rangle$ ($\langle (- \llbracket - \rrbracket -) \rangle$ [55, 0, 55] 55)
 and *semantics* :: $\langle 'model \Rightarrow 'fm \Rightarrow bool \rangle$ (**infix** $\langle \models \rangle$ 50)
 and \mathcal{M} :: $\langle 'a \text{ set} \Rightarrow 'model \text{ set} \rangle$
 and \mathcal{R} :: $\langle 'a \text{ set} \Rightarrow 'model \Rightarrow 'fm \Rightarrow bool \rangle$
 and *consistent* :: $\langle 'a \text{ set} \Rightarrow bool \rangle$
 and *witness* :: $\langle 'a \Rightarrow 'a \text{ set} \Rightarrow 'a \text{ set} \rangle$
 and *params* :: $\langle 'a \Rightarrow 'i \text{ set} \rangle$ +
 assumes *saturated-semantics*: $\langle \bigwedge S M p. \text{saturated } S \Longrightarrow M \in \mathcal{M}(S) \Longrightarrow M \models p \longleftrightarrow \mathcal{R}(S) M p \rangle$
 and *MCS-saturated*: $\langle \bigwedge S. \text{MCS } S \Longrightarrow \text{saturated } S \rangle$

begin

theorem *truth-lemma*:

assumes $\langle \text{MCS } S \rangle$ $\langle M \in \mathcal{M}(S) \rangle$
 shows $\langle M \models p \longleftrightarrow \mathcal{R}(S) M p \rangle$
 using *saturated-semantics MCS-saturated assms* by *blast*

end

locale *Truth-No-Witness* = *Truth-Witness semics semantics* $\mathcal{M} \mathcal{R}$ *consistent* $\langle \lambda -. \{\} \rangle$ $\langle \lambda -. \{\} \rangle$

for *semics* :: $\langle 'model \Rightarrow ('model \Rightarrow 'fm \Rightarrow bool) \Rightarrow 'fm \Rightarrow bool \rangle$
 and *semantics* :: $\langle 'model \Rightarrow 'fm \Rightarrow bool \rangle$
 and \mathcal{M} :: $\langle 'a \text{ set} \Rightarrow 'model \text{ set} \rangle$
 and \mathcal{R} :: $\langle 'a \text{ set} \Rightarrow 'model \Rightarrow 'fm \Rightarrow bool \rangle$
 and *consistent* :: $\langle 'a \text{ set} \Rightarrow bool \rangle$

end

Chapter 2

Derivations

theory *Derivations* **imports** *Maximal-Consistent-Sets* **begin**

lemma *split-finite-sets*:

assumes $\langle \text{finite } A \rangle \langle \text{finite } B \rangle$
and $\langle A \subseteq B \cup S \rangle$
shows $\langle \exists B' C. \text{finite } C \wedge A = B' \cup C \wedge B' = A \cap B \wedge C \subseteq S \rangle$
using *assms subset-UnE* **by** *auto*

lemma *split-list*:

assumes $\langle \text{set } A \subseteq \text{set } B \cup S \rangle$
shows $\langle \exists B' C. \text{set } (B' @ C) = \text{set } A \wedge \text{set } B' = \text{set } A \cap \text{set } B \wedge \text{set } C \subseteq S \rangle$
using *assms split-finite-sets* [**where** $A = \langle \text{set } A \rangle$ **and** $B = \langle \text{set } B \rangle$ **and** $S = S$]
by (*metis List.finite-set finite-Un finite-list set-append*)

2.1 Derivations

locale *Derivations* =

fixes *derive* :: $\langle 'fm \text{ list} \Rightarrow 'fm \Rightarrow \text{bool} \rangle$ (**infix** $\langle \vdash \rangle$ 50)
assumes *derive-assm* [*simp*]: $\langle \bigwedge A p. p \in \text{set } A \Longrightarrow A \vdash p \rangle$
and *derive-set*: $\langle \bigwedge A B r. A \vdash r \Longrightarrow \text{set } A = \text{set } B \Longrightarrow B \vdash r \rangle$

begin

theorem *derive-split*:

assumes $\langle \text{set } A \subseteq \text{set } B \cup X \rangle \langle A \vdash p \rangle$
shows $\langle \exists B' C. \text{set } B' = \text{set } A \cap \text{set } B \wedge \text{set } C \subseteq X \wedge B' @ C \vdash p \rangle$
using *assms derive-set split-list* [**where** $A = A$ **and** $B = B$] **by** *metis*

corollary *derive-split1*:

assumes $\langle \text{set } A \subseteq \{q\} \cup X \rangle \langle A \vdash p \rangle \langle q \in \text{set } A \rangle$
shows $\langle \exists C. \text{set } C \subseteq X \wedge q \# C \vdash p \rangle$
using *assms derive-split* [**where** $A = A$ **and** $X = X$ **and** $B = \langle [q] \rangle$ **and** $p = p$] *derive-set* [**where** $B = \langle q \#$
 $\rightarrow \rangle$]
by *auto*

end

2.2 MCSs and Explosion

locale *Derivations-MCS* = *MCS-Base consistent* + *Derivations derive*

for *consistent* :: $\langle 'fm \text{ set} \Rightarrow \text{bool} \rangle$

and *derive* :: $\langle 'fm\ list \Rightarrow 'fm \Rightarrow bool \rangle$ (**infix** $\langle \vdash \rangle$ 50) +
assumes *consistent-underivable*: $\langle \bigwedge S. consistent\ S \longleftrightarrow (\forall A. set\ A \subseteq S \longrightarrow (\exists q. \neg A \vdash q)) \rangle$
begin

theorem *MCS-explode*:

assumes $\langle consistent\ S \rangle$ $\langle maximal\ S \rangle$
shows $\langle p \notin S \longleftrightarrow (\exists A. set\ A \subseteq S \wedge (\forall q. p \# A \vdash q)) \rangle$
proof *safe*
assume $\langle p \notin S \rangle$
then obtain *B* **where** *B*: $\langle set\ B \subseteq \{p\} \cup S \rangle$ $\langle p \in set\ B \rangle$ $\langle \forall q. B \vdash q \rangle$
using *assms unfolding consistent-underivable maximal-def* **by** *blast*
moreover have $\langle set\ (p \# removeAll\ p\ B) = set\ B \rangle$
using *B(2)* **by** *auto*
ultimately have $\langle \forall q. p \# removeAll\ p\ B \vdash q \rangle$
using *derive-set* **by** *metis*
then show $\langle \exists A. set\ A \subseteq S \wedge (\forall q. p \# A \vdash q) \rangle$
using *B(1)* **by** (*metis Diff-subset-conv set-removeAll*)
next
fix *A*
assume $\langle set\ A \subseteq S \rangle$ $\langle \forall q. p \# A \vdash q \rangle$ $\langle p \in S \rangle$
then show *False*
using *assms unfolding consistent-underivable*
by (*metis (no-types, lifting) insert-subsetI list.simps(15)*)
qed
end

2.3 MCSs and Derivability

locale *Derivations-Cut-MCS = Derivations-MCS consistent derive*

for *consistent* :: $\langle 'fm\ set \Rightarrow bool \rangle$
and *derive* :: $\langle 'fm\ list \Rightarrow 'fm \Rightarrow bool \rangle$ (**infix** $\langle \vdash \rangle$ 50) +
assumes *derive-cut*: $\langle \bigwedge A\ B\ p\ q. A \vdash p \Longrightarrow p \# B \vdash q \Longrightarrow A @ B \vdash q \rangle$
begin

theorem *MCS-derive*:

assumes $\langle consistent\ S \rangle$ $\langle maximal\ S \rangle$
shows $\langle p \in S \longleftrightarrow (\exists A. set\ A \subseteq S \wedge A \vdash p) \rangle$
proof *safe*
assume $\langle p \in S \rangle$
then show $\langle \exists A. set\ A \subseteq S \wedge A \vdash p \rangle$
using *derive-assm* **by** (*metis List.set-insert empty-set empty-subsetI insert-subset singletonI*)
next
fix *A*
assume *A*: $\langle set\ A \subseteq S \rangle$ $\langle A \vdash p \rangle$

have *bot*: $\langle \forall A. set\ A \subseteq S \longrightarrow (\exists q. \neg A \vdash q) \rangle$
using *assms(1) unfolding consistent-underivable* **by** *blast*

have $\langle consistent\ (\{p\} \cup S) \rangle$
unfolding *consistent-underivable*
proof *safe*
fix *B*
assume *B*: $\langle set\ B \subseteq \{p\} \cup S \rangle$
show $\langle \exists q. \neg B \vdash q \rangle$

```

proof (rule ccontr)
  assume *:  $\langle \nexists q. \neg B \vdash q \rangle$ 
  then have  $\langle \forall q. B \vdash q \rangle$ 
    by blast
  show False
  proof (cases  $\langle p \in \text{set } B \rangle$ )
    case True
      then have  $\langle \text{set } (p \# \text{removeAll } p \ B) = \text{set } B \rangle$ 
        by auto
      then have  $\langle \forall q. p \# \text{removeAll } p \ B \vdash q \rangle$ 
        using  $\langle \forall q. B \vdash q \rangle$  derive-set by blast
      then have  $\langle \forall q. A @ \text{removeAll } p \ B \vdash q \rangle$ 
        using A(2) derive-cut by blast
      moreover have  $\langle \text{set } (A @ \text{removeAll } p \ B) \subseteq S \rangle$ 
        using A(1) B by auto
      ultimately show False
        using bot by blast
    next
      case False
      then show False
        using * B bot by auto
  qed
qed
qed
then show  $\langle p \in S \rangle$ 
  using assms unfolding maximal-def by auto
qed
end

```

2.4 Proof Rules

```

locale Derivations-Bot = Derivations-Cut-MCS consistent derive
  for consistent ::  $\langle 'fm \ \text{set} \Rightarrow \text{bool} \rangle$ 
  and derive ::  $\langle 'fm \ \text{list} \Rightarrow 'fm \Rightarrow \text{bool} \rangle$  (infix  $\langle \vdash \rangle$  50) +
  fixes bot ::  $\langle 'fm \ (\langle \perp \rangle) \rangle$ 
  assumes botE:  $\langle \bigwedge A \ p. A \vdash \perp \Longrightarrow A \vdash p \rangle$ 
begin

```

```

corollary MCS-botE [elim]:
  assumes  $\langle \text{consistent } S \rangle$   $\langle \text{maximal } S \rangle$ 
  and  $\langle \perp \in S \rangle$ 
  shows  $\langle p \in S \rangle$ 
  using assms botE MCS-derive by blast

```

```

corollary MCS-bot [simp]:
  assumes  $\langle \text{consistent } S \rangle$   $\langle \text{maximal } S \rangle$ 
  shows  $\langle \perp \notin S \rangle$ 
  using assms botE MCS-derive consistent-undervivable by blast

```

end

```

locale Derivations-Top = Derivations-Cut-MCS +
  fixes top ( $\langle \top \rangle$ )
  assumes topI:  $\langle \bigwedge A. A \vdash \top \rangle$ 

```

```

begin

corollary MCS-topI [simp]:
  assumes ⟨consistent S⟩ ⟨maximal S⟩
  shows ⟨ $\top \in S$ ⟩
  using assms topI MCS-derive by (metis empty-set empty-subsetI)

end

locale Derivations-Not = Derivations-Bot consistent derive bot
  for consistent :: ⟨'fm set  $\Rightarrow$  bool⟩
    and derive :: ⟨'fm list  $\Rightarrow$  'fm  $\Rightarrow$  bool⟩ (infix <math>\vdash</math> 50)
    and bot :: 'fm (⟨ $\perp$ ⟩) +
  fixes not :: ⟨'fm  $\Rightarrow$  'fm⟩ (⟨ $\neg$ ⟩)
  assumes
    notI: ⟨ $\bigwedge A p. p \# A \vdash \perp \Longrightarrow A \vdash \neg p$ ⟩ and
    notE: ⟨ $\bigwedge A p. A \vdash p \Longrightarrow A \vdash \neg p \Longrightarrow A \vdash \perp$ ⟩
begin

corollary MCS-not-xor:
  assumes ⟨consistent S⟩ ⟨maximal S⟩
  shows ⟨ $p \in S \longleftrightarrow \neg p \notin S$ ⟩
proof safe
  assume ⟨ $p \in S$ ⟩ ⟨ $\neg p \in S$ ⟩
  then have ⟨set [p,  $\neg p$ ]  $\subseteq S$ ⟩
    by simp
  moreover have ⟨[p,  $\neg p$ ]  $\vdash \perp$ ⟩
    using notE derive-assm by (meson list.set-intros(1) list.set-intros(2))
  ultimately have ⟨ $\perp \in S$ ⟩
    using assms MCS-derive by blast
  then show False
    using assms MCS-bot by blast
next
  assume *: ⟨ $\neg p \notin S$ ⟩
  show ⟨ $p \in S$ ⟩
  proof (rule ccontr)
    assume ⟨ $p \notin S$ ⟩
    then obtain A where A: ⟨set A  $\subseteq S$ ⟩ ⟨ $\forall q. p \# A \vdash q$ ⟩
      using assms MCS-explode by blast
    then have ⟨ $p \# A \vdash \perp$ ⟩
      by fast
    then have ⟨ $A \vdash \neg p$ ⟩
      using notI by blast
    then have ⟨ $\neg p \in S$ ⟩
      using A(1) assms MCS-derive by blast
    then show False
      using * by blast
  qed
qed

corollary MCS-not-both:
  assumes ⟨consistent S⟩ ⟨maximal S⟩
  shows ⟨ $p \notin S \vee \neg p \notin S$ ⟩
  using assms MCS-not-xor by blast

corollary MCS-not-neither:

```

```

assumes ⟨consistent S⟩ ⟨maximal S⟩
shows ⟨p ∈ S ∨ ¬ p ∈ S⟩
using assms MCS-not-xor by blast

end

locale Derivations-Con = Derivations-Cut-MCS consistent derive
for consistent :: ⟨'fm set ⇒ bool⟩
and derive :: ⟨'fm list ⇒ 'fm ⇒ bool⟩ (infix ⟨|⟩ 50) +
fixes con :: ⟨'fm ⇒ 'fm ⇒ 'fm⟩ (⟨- ∧ -⟩)
assumes
  conI: ⟨∧A p q. A ⊢ p ⇒ A ⊢ q ⇒ A ⊢ (p ∧ q)⟩ and
  conE: ⟨∧A p q r. A ⊢ (p ∧ q) ⇒ p # q # A ⊢ r ⇒ A ⊢ r⟩
begin

corollary MCS-conI [intro]:
assumes ⟨consistent S⟩ ⟨maximal S⟩
and ⟨p ∈ S⟩ ⟨q ∈ S⟩
shows ⟨(p ∧ q) ∈ S⟩
using assms MCS-derive derive-assm conI
by (metis (mono-tags, lifting) insert-subset list.set-intros(1) list.simps(15) set-subset-Cons)

corollary MCS-conE [dest]:
assumes ⟨consistent S⟩ ⟨maximal S⟩
and ⟨(p ∧ q) ∈ S⟩
shows ⟨p ∈ S ∧ q ∈ S⟩
proof –
have ⟨p # q # A ⊢ p⟩ ⟨p # q # A ⊢ q⟩ for A
using derive-assm by simp-all
then show ?thesis
using assms MCS-derive conE by blast
qed

corollary MCS-con:
assumes ⟨consistent S⟩ ⟨maximal S⟩
shows ⟨(p ∧ q) ∈ S ⟷ p ∈ S ∧ q ∈ S⟩
using assms MCS-conI MCS-conE by blast

end

locale Derivations-Dis = Derivations-Cut-MCS consistent derive
for consistent :: ⟨'fm set ⇒ bool⟩
and derive :: ⟨'fm list ⇒ 'fm ⇒ bool⟩ (infix ⟨|⟩ 50) +
fixes dis :: ⟨'fm ⇒ 'fm ⇒ 'fm⟩ (⟨- ∨ -⟩)
assumes
  disI1: ⟨∧A p q. A ⊢ p ⇒ A ⊢ (p ∨ q)⟩ and
  disI2: ⟨∧A p q. A ⊢ q ⇒ A ⊢ (p ∨ q)⟩ and
  disE: ⟨∧A p q r. A ⊢ (p ∨ q) ⇒ p # A ⊢ r ⇒ q # A ⊢ r ⇒ A ⊢ r⟩
begin

corollary MCS-disI1 [intro]:
assumes ⟨consistent S⟩ ⟨maximal S⟩
and ⟨p ∈ S⟩
shows ⟨(p ∨ q) ∈ S⟩
using assms disI1 MCS-derive by auto

```

corollary *MCS-disI2* [*intro*]:

assumes $\langle \text{consistent } S \rangle \langle \text{maximal } S \rangle$
and $\langle q \in S \rangle$
shows $\langle (p \vee q) \in S \rangle$
using *assms disI2 MCS-derive* **by** *auto*

corollary *MCS-disE* [*elim*]:

assumes $\langle \text{consistent } S \rangle \langle \text{maximal } S \rangle$
and $\langle (p \vee q) \in S \rangle$
shows $\langle p \in S \vee q \in S \rangle$

proof (*rule ccontr*)

have *bot*: $\langle \forall A. \text{set } A \subseteq S \longrightarrow (\exists q. \neg A \vdash q) \rangle$
using *assms(1) unfolding consistent-underivable* **by** *blast*

assume $\langle \neg (p \in S \vee q \in S) \rangle$

then obtain *P Q* **where**

P: $\langle \text{set } P \subseteq S \rangle \langle \forall r. p \# P \vdash r \rangle$ **and**
Q: $\langle \text{set } Q \subseteq S \rangle \langle \forall r. q \# Q \vdash r \rangle$
using *assms MCS-explode* **by** *auto*

have $\langle p \# (p \vee q) \# Q \vdash p \rangle$

by *simp*

then have $\langle p \# (p \vee q) \# Q @ P \vdash r \rangle$ **for** *r*

using *P derive-cut[of - p]* **by** (*metis append-Cons*)

then have $\langle p \# (p \vee q) \# P @ Q \vdash r \rangle$ **for** *r*

using *derive-set* **[where** *B* = $\langle p \# (p \vee q) \# P @ Q \rangle$ **by** *fastforce*

moreover have $\langle q \# (p \vee q) \# P \vdash q \rangle$

by *simp*

then have $\langle q \# (p \vee q) \# P @ Q \vdash r \rangle$ **for** *r*

using *Q derive-cut[of - q]* **by** (*metis append-Cons*)

moreover have $\langle (p \vee q) \# P @ Q \vdash (p \vee q) \rangle$

by *simp*

ultimately have $\langle (p \vee q) \# P @ Q \vdash r \rangle$ **for** *r*

using *disE* **by** *blast*

moreover have $\langle \text{set } ((p \vee q) \# P @ Q) \subseteq S \rangle$

using *assms(3) P Q* **by** *simp*

ultimately show *False*

using *assms(1) unfolding consistent-underivable* **by** *blast*

qed

corollary *MCS-dis*:

assumes $\langle \text{consistent } S \rangle \langle \text{maximal } S \rangle$
shows $\langle (p \vee q) \in S \longleftrightarrow p \in S \vee q \in S \rangle$
using *assms MCS-disI1 MCS-disI2 MCS-disE* **by** *blast*

end

locale *Derivations-Imp = Derivations-Cut-MCS* *consistent derive*

for *consistent* :: $\langle 'fm \text{ set} \Rightarrow \text{bool} \rangle$

and *derive* :: $\langle 'fm \text{ list} \Rightarrow 'fm \Rightarrow \text{bool} \rangle$ (**infix** $\langle \vdash \rangle$ 50) +

fixes *imp* :: $\langle 'fm \Rightarrow 'fm \Rightarrow 'fm \rangle$ ($\langle \vdash \rightarrow \vdash \rangle$)

assumes

impI: $\langle \bigwedge A p q. p \# A \vdash q \Longrightarrow A \vdash (p \rightarrow q) \rangle$ **and**

impE: $\langle \bigwedge A p q. A \vdash p \Longrightarrow A \vdash (p \rightarrow q) \Longrightarrow A \vdash q \rangle$

begin

corollary *MCS-impI* [*intro*]:

assumes $\langle \text{consistent } S \rangle \langle \text{maximal } S \rangle$
and $\langle p \in S \longrightarrow q \in S \rangle$
shows $\langle (p \rightarrow q) \in S \rangle$
using *assms impI derive-assm MCS-derive MCS-explode*
by (*metis insert-subset list.simps(15) subsetI*)

corollary *MCS-impE* [*dest*]:

assumes $\langle \text{consistent } S \rangle \langle \text{maximal } S \rangle$
and $\langle (p \rightarrow q) \in S \rangle \langle p \in S \rangle$
shows $\langle q \in S \rangle$
using *assms(3-4) impE MCS-derive[OF assms(1-2)] derive-assm*
by (*metis insert-subset list.set-intros(1) list.simps(15) set-subset-Cons*)

corollary *MCS-imp*:

assumes $\langle \text{consistent } S \rangle \langle \text{maximal } S \rangle$
shows $\langle (p \rightarrow q) \in S \longleftrightarrow (p \in S \longrightarrow q \in S) \rangle$
using *assms MCS-impI MCS-impE* **by** *blast*

end

locale *Derivations-Exi* = *MCS-Witness consistent witness params* + *Derivations-Cut-MCS consistent derive*

for *consistent* :: $\langle 'fm \text{ set} \Rightarrow \text{bool} \rangle$
and *witness params*
and *derive* :: $\langle 'fm \text{ list} \Rightarrow 'fm \Rightarrow \text{bool} \rangle$ (**infix** $\langle \vdash \rangle$ 50) +
fixes *exi* :: $\langle 'fm \Rightarrow 'fm \rangle$ ($\langle \exists \rangle$)
and *inst* :: $\langle 't \Rightarrow 'fm \Rightarrow 'fm \rangle$ ($\langle \{-} \rangle$)
assumes
exi-witness: $\langle \bigwedge S S' p. \text{MCS } S \Longrightarrow \text{witness } (\exists p) S' \subseteq S \Longrightarrow \exists t. \langle t \rangle p \in S \rangle$ **and**
exiI: $\langle \bigwedge A p t. A \vdash \langle t \rangle p \Longrightarrow A \vdash \exists p \rangle$

begin

corollary *MCS-exiI* [*intro*]:

assumes $\langle \text{consistent } S \rangle \langle \text{maximal } S \rangle$
and $\langle \langle t \rangle p \in S \rangle$
shows $\langle \exists p \in S \rangle$
using *assms MCS-derive exiI* **by** *blast*

corollary *MCS-exiE* [*dest*]:

assumes $\langle \text{consistent } S \rangle \langle \text{maximal } S \rangle \langle \text{witnessed } S \rangle$
and $\langle \exists p \in S \rangle$
shows $\langle \exists t. \langle t \rangle p \in S \rangle$
using *assms exi-witness unfolding witnessed-def* **by** *blast*

corollary *MCS-exi*:

assumes $\langle \text{consistent } S \rangle \langle \text{maximal } S \rangle \langle \text{witnessed } S \rangle$
shows $\langle \exists p \in S \longleftrightarrow (\exists t. \langle t \rangle p \in S) \rangle$
using *assms MCS-exiI MCS-exiE* **by** *blast*

end

locale *Derivations-Uni* = *MCS-Witness consistent witness params* + *Derivations-Not consistent derive bot not*

for *consistent* :: $\langle 'fm \text{ set} \Rightarrow \text{bool} \rangle$
and *witness params*

```

    and derive :: ⟨'fm list ⇒ 'fm ⇒ bool⟩ (infix ‹‹› 50)
    and bot :: 'fm (‹⊥›)
    and not :: ⟨'fm ⇒ 'fm⟩ (‹¬›) +
fixes uni :: ⟨'fm ⇒ 'fm⟩ (‹∀›)
    and inst :: ⟨'t ⇒ 'fm ⇒ 'fm⟩ (‹⟨-⟩›)
assumes
  uni-witness: ⟨∧ S S' p. MCS S ⇒ witness (¬ (∀ p)) S' ⊆ S ⇒ ∃ t. ¬ (⟨t⟩p) ∈ S⟩ and
  uniE: ⟨∧ A p t. A ⊢ ∀ p ⇒ A ⊢ ⟨t⟩p⟩
begin

corollary MCS-uniE [dest]:
  assumes ‹consistent S⟩ ‹maximal S⟩
    and ‹∀ p ∈ S⟩
  shows ‹⟨t⟩p ∈ S⟩
  using assms MCS-derive uniE by blast

corollary MCS-uniI [intro]:
  assumes ‹consistent S⟩ ‹maximal S⟩ ‹witnessed S⟩
    and ‹∀ t. ⟨t⟩p ∈ S⟩
  shows ‹∀ p ∈ S⟩
proof (rule ccontr)
  assume ‹∀ p ∉ S⟩
  then have ‹¬ (∀ p) ∈ S⟩
    using assms MCS-not-xor by blast
  then have ‹∃ t. ¬ (⟨t⟩p) ∈ S⟩
    using assms uni-witness unfolding witnessed-def by blast
  then show False
    using assms MCS-not-xor by blast
qed

corollary MCS-uni:
  assumes ‹consistent S⟩ ‹maximal S⟩ ‹witnessed S⟩
  shows ‹∀ p ∈ S ⟷ (∀ t. ⟨t⟩p ∈ S)⟩
  using assms MCS-uniI MCS-uniE by blast

end

end

```

Chapter 3

Refutations

theory *Refutations* **imports** *Maximal-Consistent-Sets* **begin**

lemma *split-finite-sets*:

assumes $\langle \text{finite } A \rangle \langle \text{finite } B \rangle$

and $\langle A \subseteq B \cup S \rangle$

shows $\langle \exists B' C. \text{finite } C \wedge A = B' \cup C \wedge B' = A \cap B \wedge C \subseteq S \rangle$

using *assms subset-UnE* **by** *auto*

lemma *split-list*:

assumes $\langle \text{set } A \subseteq \text{set } B \cup S \rangle$

shows $\langle \exists B' C. \text{set } (B' @ C) = \text{set } A \wedge \text{set } B' = \text{set } A \cap \text{set } B \wedge \text{set } C \subseteq S \rangle$

using *assms split-finite-sets* [**where** $A = \langle \text{set } A \rangle$ **and** $B = \langle \text{set } B \rangle$ **and** $S = S$]

by (*metis List.finite-set finite-Un finite-list set-append*)

3.1 Rearranging Refutations

locale *Refutations* =

fixes *refute* :: $\langle 'fm \text{ list} \Rightarrow \text{bool} \rangle$

assumes *refute-set*: $\langle \bigwedge A B. \text{refute } A \Longrightarrow \text{set } A = \text{set } B \Longrightarrow \text{refute } B \rangle$

begin

theorem *refute-split*:

assumes $\langle \text{set } A \subseteq \text{set } B \cup X \rangle \langle \text{refute } A \rangle$

shows $\langle \exists B' C. \text{set } B' = \text{set } A \cap \text{set } B \wedge \text{set } C \subseteq X \wedge \text{refute } (B' @ C) \rangle$

using *assms refute-set split-list* [**where** $A = A$ **and** $B = B$] **by** *metis*

corollary *refute-split1*:

assumes $\langle \text{set } A \subseteq \{q\} \cup X \rangle \langle \text{refute } A \rangle \langle q \in \text{set } A \rangle$

shows $\langle \exists C. \text{set } C \subseteq X \wedge \text{refute } (q \# C) \rangle$

using *assms refute-split* [**where** $A = A$ **and** $X = X$ **and** $B = \langle [q] \rangle$] *refute-set* **by** *auto*

end

3.2 MCSs and Refutability

locale *Refutations-MCS* = *MCS-Base* + *Refutations* +

assumes *consistent-refute*: $\langle \bigwedge S. \text{consistent } S \longleftrightarrow (\forall A. \text{set } A \subseteq S \longrightarrow \neg \text{refute } A) \rangle$

begin

theorem *MCS-refute*:

```

assumes  $\langle \text{consistent } S \rangle \langle \text{maximal } S \rangle$ 
shows  $\langle p \notin S \longleftrightarrow (\exists A. \text{set } A \subseteq S \wedge \text{refute } (p \# A)) \rangle$ 
proof safe
  assume  $\langle p \notin S \rangle$ 
  then obtain  $B$  where  $B: \langle \text{set } B \subseteq \{p\} \cup S \rangle \langle p \in \text{set } B \rangle \langle \text{refute } B \rangle$ 
    using assms unfolding consistent-refute maximal-def by blast
  moreover have  $\langle \text{set } (p \# \text{removeAll } p B) = \text{set } B \rangle$ 
    using  $B(2)$  by auto
  ultimately have  $\langle \text{refute } (p \# \text{removeAll } p B) \rangle$ 
    using refute-set by metis
  then show  $\langle \exists A. \text{set } A \subseteq S \wedge \text{refute } (p \# A) \rangle$ 
    using  $B(1)$  by (metis Diff-subset-conv set-removeAll)
next
  fix  $A$ 
  assume  $\langle \text{set } A \subseteq S \rangle \langle \text{refute } (p \# A) \rangle \langle p \in S \rangle$ 
  then show False
    using assms unfolding consistent-refute
    by (metis (no-types, lifting) insert-subsetI list.simps(15))
qed

end

end

```

Chapter 4

Example: Propositional Tableau Calculus

theory *Example-Propositional-Tableau* imports *Refutations* begin

4.1 Syntax

```
datatype 'p fm
  = Pro 'p (<·>)
  | Neg <'p fm> (<¬ -> [70] 70)
  | Imp <'p fm> <'p fm> (infixr <⟶> 55)
```

4.2 Semantics

```
type-synonym 'p model = <'p ⇒ bool>
```

```
fun semantics :: <'p model ⇒ 'p fm ⇒ bool> (infix <|=T> 50) where
  <I |=T ·P ⟷ I P>
| <I |=T ¬ p ⟷ ¬ I |=T p>
| <I |=T p ⟶ q ⟷ I |=T p ⟶ I |=T q>
```

4.3 Calculus

```
inductive Calculus :: <'p fm list ⇒ bool> (<⊢T -> [50] 50) where
  Axiom [simp]: <⊢T ·P # ¬ ·P # A>
| NegI [intro]: <⊢T p # A ⟹ ⊢T ¬ ¬ p # A>
| ImpP [intro]: <⊢T ¬ p # A ⟹ ⊢T q # A ⟹ ⊢T (p ⟶ q) # A>
| ImpN [intro]: <⊢T p # ¬ q # A ⟹ ⊢T ¬ (p ⟶ q) # A>
| Weak: <⊢T A ⟹ set A ⊆ set B ⟹ ⊢T B>
```

lemma *Weak2*:

```
  assumes <⊢T p # A> <⊢T q # B>
  shows <⊢T p # A @ B ∧ ⊢T q # A @ B>
  using assms Weak[where A=<- # -> and B=<- # A @ B>] by fastforce
```

4.4 Soundness

```
theorem soundness: <⊢T A ⟹ ∃ p ∈ set A. ¬ I |=T p>
  by (induct A rule: Calculus.induct) auto
```

corollary *soundness'*: $\langle \vdash_T [\neg p] \implies I \models_T p \rangle$
using *soundness by fastforce*

corollary $\langle \neg \vdash_T [] \rangle$
using *soundness by fastforce*

4.5 Maximal Consistent Sets

definition *consistent* :: $\langle 'p \text{ fm set} \implies \text{bool} \rangle$ **where**
 $\langle \text{consistent } S \equiv \forall A. \text{ set } A \subseteq S \longrightarrow \neg \vdash_T A \rangle$

interpretation *MCS-No-Witness-UNIV consistent*

proof

show $\langle \text{infinite } (UNIV :: 'p \text{ fm set}) \rangle$
using *infinite-UNIV-size*[of $\langle \lambda p. p \longrightarrow p \rangle$] **by** *simp*
qed (*auto simp: consistent-def*)

interpretation *Refutations-MCS consistent Calculus*

proof

fix $A B :: \langle 'p \text{ fm list} \rangle$
assume $\langle \vdash_T A \rangle \langle \text{set } A = \text{set } B \rangle$
then show $\langle \vdash_T B \rangle$
using *Weak by blast*
next
fix $S :: \langle 'p \text{ fm set} \rangle$
show $\langle \text{consistent } S \longleftrightarrow (\forall A. \text{ set } A \subseteq S \longrightarrow \neg \vdash_T A) \rangle$
unfolding *consistent-def ..*
qed

4.6 Truth Lemma

abbreviation (*input*) *canonical* :: $\langle 'p \text{ fm set} \implies 'p \text{ model} \rangle (\langle \mathcal{M}_T \rangle)$ **where**
 $\langle \mathcal{M}_T(S) \equiv \lambda P. \cdot P \in S \rangle$

locale *Hintikka* =

fixes $H :: \langle 'a \text{ fm set} \rangle$
assumes *AxiomH*: $\langle \bigwedge P. \cdot P \in H \implies \neg \cdot P \in H \implies \text{False} \rangle$
and *NegIH*: $\langle \bigwedge p. \neg \neg p \in H \implies p \in H \rangle$
and *ImpPH*: $\langle \bigwedge p q. p \longrightarrow q \in H \implies \neg p \in H \vee q \in H \rangle$
and *ImpNH*: $\langle \bigwedge p q. \neg (p \longrightarrow q) \in H \implies p \in H \wedge \neg q \in H \rangle$

lemma *Hintikka-model*:

assumes $\langle \text{Hintikka } H \rangle$
shows $\langle (p \in H \longrightarrow \mathcal{M}_T(H) \models_T p) \wedge (\neg p \in H \longrightarrow \neg \mathcal{M}_T(H) \models_T p) \rangle$
using *assms by (induct p) (unfold Hintikka-def semantics.simps; blast)+*

lemma *MCS-Hintikka*:

assumes $\langle \text{MCS } H \rangle$
shows $\langle \text{Hintikka } H \rangle$

proof

fix P
assume $\langle \cdot P \in H \rangle \langle \neg \cdot P \in H \rangle$
then have $\langle \text{set } [\cdot P, \neg \cdot P] \subseteq H \rangle$
by *simp*

```

moreover have  $\langle \vdash_T [\cdot P, \neg \cdot P] \rangle$ 
  by simp
ultimately show False
  using assms unfolding consistent-def by blast
next
fix p
assume  $\langle \neg \neg p \in H \rangle$ 
then show  $\langle p \in H \rangle$ 
  using assms MCS-refute by blast
next
fix p q
assume *:  $\langle p \longrightarrow q \in H \rangle$ 
show  $\langle \neg p \in H \vee q \in H \rangle$ 
proof (rule ccontr)
  assume  $\langle \neg (\neg p \in H \vee q \in H) \rangle$ 
  then have  $\langle \neg p \notin H \rangle \langle q \notin H \rangle$ 
    by blast+
  then have  $\langle \exists A. \text{set } A \subseteq H \wedge \vdash_T \neg p \# A \rangle \langle \exists A. \text{set } A \subseteq H \wedge \vdash_T q \# A \rangle$ 
    using assms MCS-refute by blast+
  then have  $\langle \exists A. \text{set } A \subseteq H \wedge \vdash_T \neg p \# A \wedge \vdash_T q \# A \rangle$ 
    using Weak2[where  $p = \neg p$  and  $q = q$ ] by (metis Un-least set-append)
  then have  $\langle \exists A. \text{set } A \subseteq H \wedge \vdash_T (p \longrightarrow q) \# A \rangle$ 
    by blast
  then have  $\langle p \longrightarrow q \notin H \rangle$ 
    using assms unfolding consistent-def by auto
  then show False
    using * ..
qed
next
fix p q
assume *:  $\langle \neg (p \longrightarrow q) \in H \rangle$ 
show  $\langle p \in H \wedge \neg q \in H \rangle$ 
proof (rule ccontr)
  assume  $\langle \neg (p \in H \wedge \neg q \in H) \rangle$ 
  then consider  $\langle p \notin H \rangle \mid \langle \neg q \notin H \rangle$ 
    by blast
  then show False
proof cases
  case 1
  then have  $\langle \exists A. \text{set } A \subseteq H \wedge \vdash_T p \# A \rangle$ 
    using assms MCS-refute by blast
  then have  $\langle \exists A. \text{set } A \subseteq H \wedge \vdash_T p \# \neg q \# A \rangle$ 
    using Weak[where  $B = \langle p \# \neg q \# - \rangle$ ] by fastforce
  then have  $\langle \exists A. \text{set } A \subseteq H \wedge \vdash_T \neg (p \longrightarrow q) \# A \rangle$ 
    by fast
  then have  $\langle \neg (p \longrightarrow q) \notin H \rangle$ 
    using assms unfolding consistent-def by auto
  then show False
    using * ..
  next
  case 2
  then have  $\langle \exists A. \text{set } A \subseteq H \wedge \vdash_T \neg q \# A \rangle$ 
    using assms MCS-refute by blast
  then have  $\langle \exists A. \text{set } A \subseteq H \wedge \vdash_T p \# \neg q \# A \rangle$ 
    using Weak by (metis set-subset-Cons)
  then have  $\langle \exists A. \text{set } A \subseteq H \wedge \vdash_T \neg (p \longrightarrow q) \# A \rangle$ 

```

by *fast*
 then have $\langle \neg (p \longrightarrow q) \notin H \rangle$
 using *assms unfolding consistent-def* by *auto*
 then show *False*
 using * ..
 qed
 qed
 qed

lemma *truth-lemma*:
 assumes $\langle MCS\ H \rangle \langle p \in H \rangle$
 shows $\langle \mathcal{M}_T(H) \models_T p \rangle$
 using *Hintikka-model MCS-Hintikka assms* by *blast*

4.7 Completeness

theorem *strong-completeness*:
 assumes $\langle \forall M. (\forall q \in X. M \models_T q) \longrightarrow M \models_T p \rangle$
 shows $\langle \exists A. set\ A \subseteq X \wedge \vdash_T \neg p \# A \rangle$
proof (*rule ccontr*)
 assume $\langle \nexists A. set\ A \subseteq X \wedge \vdash_T \neg p \# A \rangle$
 then have *: $\langle \forall A. set\ A \subseteq \{\neg p\} \cup X \longrightarrow \neg \vdash_T A \rangle$
 using *refute-split1* by (*metis Weak insert-is-Un set-subset-Cons subset-insert*)

let $?S = \langle \{\neg p\} \cup X \rangle$
 let $?H = \langle Extend\ ?S \rangle$

have $\langle consistent\ ?S \rangle$
 unfolding *consistent-def* using * by *blast*
 then have $\langle MCS\ ?H \rangle$
 using *MCS-Extend'* by *blast*
 then have $\langle p \in ?H \longrightarrow \mathcal{M}_T(?H) \models_T p \rangle$ for p
 using *truth-lemma* by *blast*
 then have $\langle p \in ?S \longrightarrow \mathcal{M}_T(?H) \models_T p \rangle$ for p
 using *Extend-subset* by *blast*
 then have $\langle \mathcal{M}_T(?H) \models_T \neg p \rangle \langle \forall q \in X. \mathcal{M}_T(?H) \models_T q \rangle$
 by *blast+*
 moreover from *this* have $\langle \mathcal{M}_T(?H) \models_T p \rangle$
 using *assms(1)* by *blast*
 ultimately show *False*
 by *simp*
 qed

abbreviation *valid* :: $\langle 'p\ fm \Rightarrow bool \rangle$ where
 $\langle valid\ p \equiv \forall M. M \models_T p \rangle$

theorem *completeness*:
 assumes $\langle valid\ p \rangle$
 shows $\langle \vdash_T [\neg p] \rangle$
 using *assms strong-completeness* [where $X = \{\}$] by *auto*

theorem *main*: $\langle valid\ p \longleftrightarrow \vdash_T [\neg p] \rangle$
 using *completeness soundness'* by *blast*

end

Chapter 5

Example: Propositional Sequent Calculus

theory *Example-Propositional-SC* **imports** *Derivations* **begin**

5.1 Syntax

datatype *'p fm*
= *Fls* ($\langle \perp \rangle$)
| *Pro* *'p* ($\langle \cdot \rangle$)
| *Imp* *'p fm* *'p fm* (**infixr** $\langle \longrightarrow \rangle$ 55)

abbreviation *Neg* ($\langle \neg \rightarrow [70] 70 \rangle$) **where** $\langle \neg p \equiv p \longrightarrow \perp \rangle$

5.2 Semantics

type-synonym *'p model* = $\langle 'p \Rightarrow \text{bool} \rangle$

fun *semantics* :: $\langle 'p \text{ model} \Rightarrow 'p \text{ fm} \Rightarrow \text{bool} \rangle$ (**infix** $\langle \models_S \rangle$ 50) **where**
 $\langle \cdot \models_S \perp \longleftrightarrow \text{False} \rangle$
| $\langle I \models_S \cdot P \longleftrightarrow I P \rangle$
| $\langle I \models_S p \longrightarrow q \longleftrightarrow I \models_S p \longrightarrow I \models_S q \rangle$

5.3 Calculus

inductive *Calculus* :: $\langle 'p \text{ fm list} \Rightarrow 'p \text{ fm list} \Rightarrow \text{bool} \rangle$ (**infix** $\langle \vdash_S \rangle$ 50) **where**
Axiom [*simp*]: $\langle p \# A \vdash_S p \# B \rangle$
| *FlsL* [*simp*]: $\langle \perp \# A \vdash_S B \rangle$
| *FlsR* [*elim*]: $\langle A \vdash_S \perp \# B \Longrightarrow A \vdash_S B \rangle$
| *ImpL* [*intro*]: $\langle A \vdash_S p \# B \Longrightarrow q \# A \vdash_S B \Longrightarrow (p \longrightarrow q) \# A \vdash_S B \rangle$
| *ImpR* [*intro*]: $\langle p \# A \vdash_S q \# B \Longrightarrow A \vdash_S (p \longrightarrow q) \# B \rangle$
| *Cut*: $\langle A \vdash_S [p] \Longrightarrow p \# A \vdash_S B \Longrightarrow A \vdash_S B \rangle$
| *WeakL*: $\langle A \vdash_S B \Longrightarrow \text{set } A \subseteq \text{set } A' \Longrightarrow A' \vdash_S B \rangle$
| *WeakR*: $\langle A \vdash_S B \Longrightarrow \text{set } B \subseteq \text{set } B' \Longrightarrow A \vdash_S B' \rangle$

lemma *Boole*: $\langle \neg p \# A \vdash_S [] \Longrightarrow A \vdash_S [p] \rangle$
by (*meson Axiom Cut ImpL ImpR WeakR set-subset-Cons*)

5.4 Soundness

theorem *soundness*: $\langle A \vdash_S B \implies \forall q \in \text{set } A. I \models_S q \implies \exists p \in \text{set } B. I \models_S p \rangle$
 by (*induct A B rule: Calculus.induct*) *auto*

corollary *soundness'*: $\langle [] \vdash_S [p] \implies I \models_S p \rangle$
 using *soundness* by *fastforce*

corollary $\langle \neg [] \vdash_S [] \rangle$
 using *soundness* by *fastforce*

5.5 Maximal Consistent Sets

definition *consistent* :: $\langle 'p \text{ fm set} \implies \text{bool} \rangle$ **where**
 $\langle \text{consistent } S \equiv \forall A. \text{set } A \subseteq S \longrightarrow \neg A \vdash_S [\perp] \rangle$

interpretation *MCS-No-Witness-UNIV consistent*

proof

show $\langle \text{infinite } (\text{UNIV} :: 'p \text{ fm set}) \rangle$
 using *infinite-UNIV-size*[of $\langle \lambda p. p \longrightarrow p \rangle$] by *simp*
qed (*auto simp: consistent-def*)

interpretation *Derivations-Cut-MCS consistent* $\langle \lambda A p. A \vdash_S [p] \rangle$

proof

fix $A B$ **and** $p :: 'p \text{ fm}$
assume $\langle A \vdash_S [p] \rangle \langle \text{set } A = \text{set } B \rangle$
then show $\langle B \vdash_S [p] \rangle$
 using *WeakL* by *blast*

next

fix $S :: 'p \text{ fm set}$
show $\langle \text{consistent } S \iff (\forall A. \text{set } A \subseteq S \longrightarrow (\exists q. \neg A \vdash_S [q])) \rangle$
 unfolding *consistent-def* using *Cut FlsL* by *blast*

next

fix A **and** $p :: 'p \text{ fm}$
assume $\langle p \in \text{set } A \rangle$
then show $\langle A \vdash_S [p] \rangle$
 by (*metis Axiom WeakL set-ConsD subsetI*)

next

fix $A B$ **and** $p q :: 'p \text{ fm}$
assume $\langle A \vdash_S [p] \rangle \langle p \# B \vdash_S [q] \rangle$
then have $\langle A @ B \vdash_S [p] \rangle \langle p \# A @ B \vdash_S [q] \rangle$
 by (*fastforce intro: WeakL*)+
then show $\langle A @ B \vdash_S [q] \rangle$
 using *Cut* by *blast*

qed

interpretation *Derivations-Bot consistent* $\langle \lambda A p. A \vdash_S [p] \rangle \langle \perp \rangle$

proof

show $\langle \bigwedge A r. A \vdash_S [\perp] \implies A \vdash_S [r] \rangle$
 using *Cut FlsL* by *blast*

qed

interpretation *Derivations-Imp consistent* $\langle \lambda A p. A \vdash_S [p] \rangle \langle \lambda p q. p \longrightarrow q \rangle$

proof

show $\langle \bigwedge A p q. A \vdash_S [p] \implies A \vdash_S [p \longrightarrow q] \implies A \vdash_S [q] \rangle$

by (*meson Axiom Cut ImpL*)
qed *fast+*

5.6 Truth Lemma

abbreviation *canonical* :: $\langle 'p \text{ fm set} \Rightarrow 'p \text{ model} \rangle (\langle \mathcal{M}_S \rangle)$ **where**
 $\langle \mathcal{M}_S(S) \equiv \lambda P. \cdot P \in S \rangle$

fun *semics* :: $\langle 'p \text{ model} \Rightarrow ('p \text{ model} \Rightarrow 'p \text{ fm} \Rightarrow \text{bool}) \Rightarrow 'p \text{ fm} \Rightarrow \text{bool} \rangle$
 $(\langle \cdot \llbracket \cdot \rrbracket_S \rightarrow [55, 0, 55] 55 \rangle)$ **where**
 $\langle \cdot \llbracket \cdot \rrbracket_S \perp \longleftrightarrow \text{False} \rangle$
 $| \langle I \llbracket \cdot \rrbracket_S \cdot P \longleftrightarrow I P \rangle$
 $| \langle I \llbracket \mathcal{R} \rrbracket_S p \longrightarrow q \longleftrightarrow \mathcal{R} I p \longrightarrow \mathcal{R} I q \rangle$

fun *rel* :: $\langle 'p \text{ fm set} \Rightarrow 'p \text{ model} \Rightarrow 'p \text{ fm} \Rightarrow \text{bool} \rangle (\langle \mathcal{R}_S \rangle)$ **where**
 $\langle \mathcal{R}_S(S) - p \longleftrightarrow p \in S \rangle$

theorem *saturated-model*:

assumes $\langle \bigwedge p. \forall M \in \{\mathcal{M}_S(S)\}. M \llbracket \mathcal{R}_S(S) \rrbracket_S p = \mathcal{R}_S(S) M p \rangle$ $\langle M \in \{\mathcal{M}_S(S)\} \rangle$
shows $\langle \mathcal{R}_S(S) M p \longleftrightarrow M \models_S p \rangle$
proof (*induct p rule: wf-induct[where r= $\langle \text{measure size} \rangle$]*)
case 1
then show ?*case* ..
next
case (2 *x*)
then show ?*case*
using *assms(1)[of x] assms(2) by (cases x) simp-all*
qed

theorem *saturated-MCS*:

assumes $\langle \text{MCS } S \rangle$ $\langle M \in \{\mathcal{M}_S(S)\} \rangle$
shows $\langle M \llbracket \mathcal{R}_S(S) \rrbracket_S p \longleftrightarrow \mathcal{R}_S(S) M p \rangle$
using *assms by (cases p) auto*

interpretation *Truth-No-Witness semics semantics* $\langle \lambda S. \{\mathcal{M}_S(S)\} \rangle$ *rel consistent*

proof

fix *p* **and** *M* :: $\langle 'p \text{ model} \rangle$
show $\langle (M \models_S p) = M \llbracket (\models_S) \rrbracket_S p \rangle$
by (*induct p*) *auto*
qed (*use saturated-model saturated-MCS in blast*)**+**

5.7 Completeness

theorem *strong-completeness*:

assumes $\langle \forall M. (\forall q \in X. M \models_S q) \longrightarrow M \models_S p \rangle$
shows $\langle \exists A. \text{set } A \subseteq X \wedge A \vdash_S [p] \rangle$
proof (*rule ccontr*)
assume $\langle \nexists A. \text{set } A \subseteq X \wedge A \vdash_S [p] \rangle$
then have *: $\langle \forall A. \text{set } A \subseteq \{\neg p\} \cup X \longrightarrow \neg A \vdash_S [\perp] \rangle$
using *derive-split1 botE Boole FlsR by (metis (full-types) insert-is-Un subset-insert-iff)*

let ?*X* = $\langle \{\neg p\} \cup X \rangle$
let ?*S* = $\langle \text{Extend } ?X \rangle$

have $\langle \text{consistent } ?X \rangle$

unfolding *consistent-def* **using** * .
then have $\langle MCS \ ?S \rangle$
using *MCS-Extend'* **by** *blast*
then have $\langle p \in ?S \longleftrightarrow \mathcal{M}_S \ ?S \models_S p \rangle$ **for** p
using *truth-lemma* **by** *fastforce*
then have $\langle p \in ?X \longrightarrow \mathcal{M}_S \ ?S \models_S p \rangle$ **for** p
using *Extend-subset* **by** *blast*
then have $\langle \mathcal{M}_S \ ?S \models_S \neg p \rangle \langle \forall q \in X. \mathcal{M}_S \ ?S \models_S q \rangle$
by *blast+*
moreover from *this* **have** $\langle \mathcal{M}_S \ ?S \models_S p \rangle$
using *assms(1)* **by** *blast*
ultimately show *False*
by *simp*
qed

abbreviation *valid* :: $\langle 'p \text{ fm} \Rightarrow \text{bool} \rangle$ **where**
 $\langle \text{valid } p \equiv \forall M. M \models_S p \rangle$

theorem *completeness*:
assumes $\langle \text{valid } p \rangle$
shows $\langle [] \vdash_S [p] \rangle$
using *assms strong-completeness* [**where** $X = \langle \{ \} \rangle$] **by** *auto*

theorem *main*: $\langle \text{valid } p \longleftrightarrow [] \vdash_S [p] \rangle$
using *completeness soundness'* **by** *blast*

end

Chapter 6

Example: Modal Logic

theory *Example-Modal-Logic* imports *Derivations* begin

6.1 Syntax

```
datatype ('i, 'p) fm
  = Fls (<⊥>)
  | Pro 'p (<⋅>)
  | Imp (<'i, 'p> fm) (<'i, 'p> fm) (infixr <⟶> 55)
  | Box 'i (<'i, 'p> fm) (<□>)
```

abbreviation *Neg* (<¬ → [70] 70) where
 <¬ p ≡ p ⟶ ⊥>

6.2 Semantics

```
datatype ('i, 'p, 'w) model =
  Model (W: <'w set>) (R: <'i ⇒ 'w ⇒ 'w set>) (V: <'w ⇒ 'p ⇒ bool>)
```

type-synonym ('i, 'p, 'w) ctx = <'i, 'p, 'w> model × 'w

```
fun semantics :: <'i, 'p, 'w> ctx ⇒ ('i, 'p) fm ⇒ bool (infix <⊨□> 50) where
  <⊥ ⊨□ False>
  | <(M, w) ⊨□ ⋅P ⟷ V M w P>
  | <(M, w) ⊨□ p ⟶ q ⟷ (M, w) ⊨□ p ⟶ (M, w) ⊨□ q>
  | <(M, w) ⊨□ □ i p ⟷ (∀ v ∈ W M ∩ R M i w. (M, v) ⊨□ p)>
```

6.3 Calculus

```
primrec eval :: <'p ⇒ bool> ⇒ (('i, 'p) fm ⇒ bool) ⇒ ('i, 'p) fm ⇒ bool where
  <eval - - ⊥ = False>
  | <eval g - (⋅P) = g P>
  | <eval g h (p ⟶ q) = (eval g h p ⟶ eval g h q)>
  | <eval - h (□ i p) = h (□ i p)>
```

abbreviation <tautology p ≡ ∀ g h. eval g h p>

```
inductive Calculus :: <'i, 'p> fm ⇒ bool (⟨⊢□ → [50] 50) where
  A1: <tautology p ⟹ ⊢□ p>
  | A2: <⊢□ □ i (p ⟶ q) ⟶ □ i p ⟶ □ i q>
```

| *R1*: $\langle \vdash_{\square} p \implies \vdash_{\square} p \longrightarrow q \implies \vdash_{\square} q \rangle$
| *R2*: $\langle \vdash_{\square} p \implies \vdash_{\square} \square i p \rangle$

primrec *imply* :: $\langle ('i, 'p) \text{ fm list} \Rightarrow ('i, 'p) \text{ fm} \Rightarrow ('i, 'p) \text{ fm} \rangle$ (**infixr** $\langle \rightsquigarrow \rangle$ 56) **where**
 $\langle (\square \rightsquigarrow p) = p \rangle$
| $\langle (q \# A \rightsquigarrow p) = (q \longrightarrow A \rightsquigarrow p) \rangle$

abbreviation *Calculus-assms* (**infix** $\langle \vdash_{\square} \rangle$ 50) **where**
 $\langle A \vdash_{\square} p \equiv \vdash_{\square} A \rightsquigarrow p \rangle$

6.4 Soundness

lemma *eval-semantic*: $\langle \text{eval } (g w) (\lambda q. (\text{Model } Ws r g, w) \models_{\square} q) p = ((\text{Model } Ws r g, w) \models_{\square} p) \rangle$
by (*induct p simp-all*)

lemma *tautology*:

assumes $\langle \text{tautology } p \rangle$
shows $\langle (M, w) \models_{\square} p \rangle$

proof –

from *assms* **have** $\langle \text{eval } (g w) (\lambda q. (\text{Model } Ws r g, w) \models_{\square} q) p \rangle$ **for** $Ws g r$
by *simp*
then **have** $\langle (\text{Model } Ws r g, w) \models_{\square} p \rangle$ **for** $Ws g r$
using *eval-semantic* **by** *fast*
then **show** $\langle (M, w) \models_{\square} p \rangle$
by (*metis model.exhaust*)

qed

theorem *soundness*: $\langle \vdash_{\square} p \implies w \in W M \implies (M, w) \models_{\square} p \rangle$
by (*induct p arbitrary: w rule: Calculus.induct*) (*auto simp: tautology*)

6.5 Admissible rules

lemma *K-imply-head*: $\langle p \# A \vdash_{\square} p \rangle$

proof –

have $\langle \text{tautology } (p \# A \rightsquigarrow p) \rangle$
by (*induct A simp-all*)
then **show** *?thesis*
using *A1* **by** *blast*

qed

lemma *K-imply-Cons*:

assumes $\langle A \vdash_{\square} q \rangle$
shows $\langle p \# A \vdash_{\square} q \rangle$
using *assms* **by** (*auto simp: A1 intro: R1*)

lemma *K-right-mp*:

assumes $\langle A \vdash_{\square} p \rangle \langle A \vdash_{\square} p \longrightarrow q \rangle$
shows $\langle A \vdash_{\square} q \rangle$

proof –

have $\langle \text{tautology } (A \rightsquigarrow p \longrightarrow A \rightsquigarrow (p \longrightarrow q) \longrightarrow A \rightsquigarrow q) \rangle$
by (*induct A simp-all*)
with *A1* **have** $\langle \vdash_{\square} A \rightsquigarrow p \longrightarrow A \rightsquigarrow (p \longrightarrow q) \longrightarrow A \rightsquigarrow q \rangle$.
then **show** *?thesis*
using *assms R1* **by** *blast*

qed

lemma *deduct1*: $\langle A \vdash_{\square} p \longrightarrow q \implies p \# A \vdash_{\square} q \rangle$
 by (*meson K-right-mp K-imp-Cons K-imp-head*)

lemma *imp-append* [*iff*]: $\langle (A @ B \rightsquigarrow r) = (A \rightsquigarrow B \rightsquigarrow r) \rangle$
 by (*induct A simp-all*)

lemma *imp-swap-append*: $\langle A @ B \vdash_{\square} r \implies B @ A \vdash_{\square} r \rangle$

proof (*induct B arbitrary: A*)

case *Cons*

then show *?case*

by (*metis deduct1 imp.simps(2) imp-append*)

qed *simp*

lemma *K-ImpI*: $\langle p \# A \vdash_{\square} q \implies A \vdash_{\square} p \longrightarrow q \rangle$

by (*metis imp.simps imp-append imp-swap-append*)

lemma *imp-mem* [*simp*]: $\langle p \in \text{set } A \implies A \vdash_{\square} p \rangle$

using *K-imp-head K-imp-Cons* by (*induct A fastforce+*)

lemma *add-imp* [*simp*]: $\langle \vdash_{\square} q \implies A \vdash_{\square} q \rangle$

using *K-imp-head R1* by *auto*

lemma *K-imp-weaken*: $\langle A \vdash_{\square} q \implies \text{set } A \subseteq \text{set } A' \implies A' \vdash_{\square} q \rangle$

by (*induct A arbitrary: q*) (*simp, metis K-right-mp K-ImpI imp-mem insert-subset list.set(2)*)

lemma *K-Boole*:

assumes $\langle (\neg p) \# A \vdash_{\square} \perp \rangle$

shows $\langle A \vdash_{\square} p \rangle$

proof –

have $\langle A \vdash_{\square} \neg \neg p \rangle$

using *assms K-ImpI* by *blast*

moreover have $\langle \text{tautology } (A \rightsquigarrow \neg \neg p \longrightarrow A \rightsquigarrow p) \rangle$

by (*induct A simp-all*)

then have $\langle \vdash_{\square} (A \rightsquigarrow \neg \neg p \longrightarrow A \rightsquigarrow p) \rangle$

using *A1* by *blast*

ultimately show *?thesis*

using *R1* by *blast*

qed

lemma *K-distrib-K-imp*:

assumes $\langle \vdash_{\square} \square i (A \rightsquigarrow q) \rangle$

shows $\langle \text{map } (\square i) A \vdash_{\square} \square i q \rangle$

proof –

have $\langle \vdash_{\square} \square i (A \rightsquigarrow q) \longrightarrow \text{map } (\square i) A \rightsquigarrow \square i q \rangle$

proof (*induct A*)

case *Nil*

then show *?case*

by (*simp add: A1*)

next

case (*Cons a A*)

have $\langle \vdash_{\square} \square i (a \# A \rightsquigarrow q) \longrightarrow \square i a \longrightarrow \square i (A \rightsquigarrow q) \rangle$

by (*simp add: A2*)

moreover have

$\langle \vdash_{\square} ((\square i (a \# A \rightsquigarrow q) \longrightarrow \square i a \longrightarrow \square i (A \rightsquigarrow q)) \longrightarrow$
 $(\square i (A \rightsquigarrow q) \longrightarrow \text{map } (\square i) A \rightsquigarrow \square i q) \longrightarrow$

```

    (□ i (a # A ~ q) → □ i a → map (□ i) A ~ □ i q)
  by (simp add: A1)
  ultimately have ‹⊢□ □ i (a # A ~ q) → □ i a → map (□ i) A ~ □ i q›
    using Cons R1 by blast
  then show ?case
    by simp
qed
then show ?thesis
  using assms R1 by blast
qed

```

6.6 Maximal Consistent Sets

definition *consistent* :: ‹('i, 'p) fm set ⇒ bool› where
 ‹consistent S ≡ ∀ A. set A ⊆ S → ¬ A ⊢□ ⊥›

interpretation *MCS-No-Witness-UNIV consistent*

proof

```

  show ‹infinite (UNIV :: ('i, 'p) fm set)›
    using infinite-UNIV-size[of ‹λp. p → p›] by simp
qed (auto simp: consistent-def)

```

interpretation *Derivations-Cut-MCS consistent Calculus-assms*

proof

```

  fix A B and p :: ('i, 'p) fm
  assume ‹⊢□ A ~ p› ‹set A = set B›
  then show ‹⊢□ B ~ p›
    using K-imply-weaken by blast
next
  fix S :: ('i, 'p) fm set
  show ‹consistent S = (∀ A. set A ⊆ S → (∃ q. ¬ A ⊢□ q))›
    unfolding consistent-def using K-Boole K-imply-Cons by blast
next
  fix A B and p q :: ('i, 'p) fm
  assume ‹A ⊢□ p› ‹p # B ⊢□ q›
  then show ‹A @ B ⊢□ q›
    by (metis K-right-mp add-imply imply.simps(2) imply-append)
qed simp

```

interpretation *Derivations-Bot consistent Calculus-assms ‹⊥›*

proof

```

  show ‹∧ A r. A ⊢□ ⊥ ⇒ A ⊢□ r›
    using K-Boole K-imply-Cons by blast
qed

```

interpretation *Derivations-Imp consistent Calculus-assms ‹λp q. p → q›*

proof

```

  show ‹∧ A p q. p # A ⊢□ q ⇒ A ⊢□ p → q›
    using K-ImpI by blast
  show ‹∧ A p q. A ⊢□ p ⇒ A ⊢□ p → q ⇒ A ⊢□ q›
    using K-right-mp by blast
qed

```

theorem *deriv-in-maximal*:

assumes ‹consistent S› ‹maximal S› ‹⊢□ p›

shows $\langle p \in S \rangle$
 using *assms MCS-derive by fastforce*

6.7 Truth Lemma

abbreviation *known* :: $\langle ('i, 'p) \text{ fm set} \Rightarrow 'i \Rightarrow ('i, 'p) \text{ fm set} \rangle$ **where**
 $\langle \text{known } S \ i \equiv \{p. \Box i \ p \in S\} \rangle$

abbreviation *reach* :: $\langle 'i \Rightarrow ('i, 'p) \text{ fm set} \Rightarrow ('i, 'p) \text{ fm set set} \rangle$ **where**
 $\langle \text{reach } i \ S \equiv \{S'. \text{known } S \ i \subseteq S' \wedge \text{MCS } S'\} \rangle$

abbreviation *canonical* :: $\langle ('i, 'p) \text{ fm set} \Rightarrow ('i, 'p, ('i, 'p) \text{ fm set}) \text{ ctx} \rangle$ ($\langle \mathcal{M}_{\Box} \rangle$) **where**
 $\langle \mathcal{M}_{\Box}(S) \equiv (\text{Model } \{S. \text{MCS } S\} \text{ reach } (\lambda S \ P. \bullet P \in S), S) \rangle$

fun *semics* ::

$\langle ('i, 'p, 'w) \text{ ctx} \Rightarrow (('i, 'p, 'w) \text{ ctx} \Rightarrow ('i, 'p) \text{ fm} \Rightarrow \text{bool}) \Rightarrow ('i, 'p) \text{ fm} \Rightarrow \text{bool} \rangle$
 $\langle \langle \text{[-]}_{\Box} \text{ -} \rangle [55, 0, 55] 55 \rangle$ **where**
 $\langle \text{[-]}_{\Box} \perp \longleftrightarrow \text{False} \rangle$
 $| \langle (M, w) \text{ [-]}_{\Box} \bullet P \longleftrightarrow V \ M \ w \ P \rangle$
 $| \langle (M, w) \text{ [R]}_{\Box} p \longrightarrow q \longleftrightarrow \mathcal{R} (M, w) \ p \longrightarrow \mathcal{R} (M, w) \ q \rangle$
 $| \langle (M, w) \text{ [R]}_{\Box} \Box i \ p \longleftrightarrow (\forall v \in W \ M \cap R \ M \ i \ w. \mathcal{R} (M, v) \ p) \rangle$

fun *rel* :: $\langle ('i, 'p) \text{ fm set} \Rightarrow ('i, 'p, ('i, 'p) \text{ fm set}) \text{ ctx} \Rightarrow ('i, 'p) \text{ fm} \Rightarrow \text{bool} \rangle$ ($\langle \mathcal{R}_{\Box} \rangle$) **where**
 $\langle \mathcal{R}_{\Box}(-) (-, w) \ p \longleftrightarrow p \in w \rangle$

theorem *saturated-model*:

fixes *S* :: $\langle ('i, 'p) \text{ fm set} \rangle$
assumes $\langle \bigwedge (S :: ('i, 'p) \text{ fm set}) \ p. \text{MCS } S \Longrightarrow \mathcal{M}_{\Box}(S) \text{ [R]}_{\Box}(S') \Box p \longleftrightarrow p \in S \rangle$
shows $\langle \text{MCS } S \Longrightarrow \mathcal{M}_{\Box}(S) \models_{\Box} p \longleftrightarrow p \in S \rangle$

proof (*induct p arbitrary: S rule: wf-induct[where r= \langle measure size \rangle]*)

case 1

then show ?*case ..*

next

case (2 *x*)

then show ?*case*

using *assms[of S x] by (cases x) auto*

qed

theorem *saturated-MCS*:

assumes $\langle \text{MCS } S \rangle$
shows $\langle \mathcal{M}_{\Box}(S) \text{ [R]}_{\Box}(S') \Box p \longleftrightarrow \mathcal{R}_{\Box}(S') (\mathcal{M}_{\Box}(S)) \ p \rangle$

proof (*cases p*)

case *Fls*

have $\langle \perp \notin S \rangle$

using *assms MCS-derive unfolding consistent-def by blast*

then show ?*thesis*

using *Fls by simp*

next

case (*Imp p q*)

then show ?*thesis*

using *assms by auto*

next

case (*Box i p*)

have $\langle (\forall S' \in \text{reach } i \ S. p \in S') \longleftrightarrow \Box i \ p \in S \rangle$

proof

```

assume  $\langle \Box i p \in S \rangle$ 
then show  $\langle \forall S' \in \text{reach } i S. p \in S' \rangle$ 
  by auto
next
assume *:  $\langle \forall S' \in \text{reach } i S. p \in S' \rangle$ 
have  $\langle \neg \text{consistent } (\{\neg p\} \cup \text{known } S i) \rangle$ 
proof
  assume  $\langle \text{consistent } (\{\neg p\} \cup \text{known } S i) \rangle$ 
  then obtain  $S'$  where  $S'$ :  $\langle \{\neg p\} \cup \text{known } S i \subseteq S' \rangle \langle \text{MCS } S' \rangle$ 
    using  $\langle \text{MCS } S \rangle$  MCS-Extend' Extend-subset by metis
  then show False
    using * MCS-impE MCS-bot by force
qed
then obtain  $A$  where  $A$ :  $\langle \neg p \# A \vdash_{\Box} \perp \rangle \langle \text{set } A \subseteq \text{known } S i \rangle$ 
  unfolding consistent-def using derive-split1 K-imply-Cons
  by (metis (no-types, lifting) insert-is-Un subset-insert)
then have  $\langle \vdash_{\Box} A \rightsquigarrow p \rangle$ 
  using K-Boole by blast
then have  $\langle \vdash_{\Box} \Box i (A \rightsquigarrow p) \rangle$ 
  using R2 by fast
then have  $\langle \text{map } (\Box i) A \vdash_{\Box} \Box i p \rangle$ 
  using K-distrib-K-imp by fast
then have  $\langle (\text{map } (\Box i) A \rightsquigarrow \Box i p) \in S \rangle$ 
  using deriv-in-maximal  $\langle \text{MCS } S \rangle$  by blast
then show  $\langle \Box i p \in S \rangle$ 
  using A(2)
proof (induct A)
  case (Cons a L)
  then have  $\langle \Box i a \in S \rangle$ 
    by auto
  then have  $\langle (\text{map } (\Box i) L \rightsquigarrow \Box i p) \in S \rangle$ 
    using Cons(2)  $\langle \text{MCS } S \rangle$  MCS-impE by auto
  then show ?case
    using Cons by simp
qed simp
qed
then show ?thesis
  using Box by auto
qed simp

```

interpretation *Truth-No-Witness semics semantics* $\langle \lambda-. \{\mathcal{M}_{\Box}(S) \mid S. \text{MCS } S\} \text{ rel consistent}$

proof

```

fix  $p$  and  $M$  ::  $\langle ('i, 'p, ('i, 'p) \text{ fm set}) \text{ ctx} \rangle$ 
show  $\langle (M \models_{\Box} p) = M \llbracket \text{semantics} \rrbracket_{\Box} p \rangle$ 
  by (cases M, induct p) simp-all

```

next

```

fix  $p$  and  $S$  ::  $\langle ('i, 'p) \text{ fm set} \rangle$  and  $M$  ::  $\langle ('i, 'p, ('i, 'p) \text{ fm set}) \text{ ctx} \rangle$ 
assume  $\langle \forall p. \forall M \in \{\mathcal{M}_{\Box}(S) \mid S. \text{MCS } S\}. M \llbracket \mathcal{R}_{\Box}(S) \rrbracket_{\Box} p \longleftrightarrow \mathcal{R}_{\Box}(S) M p \rangle \langle M \in \{\mathcal{M}_{\Box}(S) \mid S. \text{MCS } S\} \rangle$ 

```

```

then show  $\langle M \models_{\Box} p \longleftrightarrow \mathcal{R}_{\Box}(S) M p \rangle$ 
  using saturated-model[of S - p] by auto

```

next

```

fix  $S$  ::  $\langle ('i, 'p) \text{ fm set} \rangle$  and  $M$  ::  $\langle ('i, 'p, ('i, 'p) \text{ fm set}) \text{ ctx} \rangle$ 
assume  $\langle \text{MCS } S \rangle$ 
then show  $\langle \forall p. \forall M \in \{\mathcal{M}_{\Box}(S) \mid S. \text{MCS } S\}. M \llbracket \mathcal{R}_{\Box}(S) \rrbracket_{\Box} p \longleftrightarrow \mathcal{R}_{\Box}(S) M p \rangle$ 
  using saturated-MCS by blast

```

qed

lemma *Truth-lemma*:

assumes $\langle \text{MCS } S \rangle$

shows $\langle \mathcal{M}_\square(S) \models_\square p \iff p \in S \rangle$

using *assms truth-lemma* **by** *fastforce*

6.8 Completeness

theorem *strong-completeness*:

assumes $\langle \forall M :: ('i, 'p, ('i, 'p) \text{ fm set}) \text{ model}. \forall w \in W M.$

$(\forall q \in X. (M, w) \models_\square q) \longrightarrow (M, w) \models_\square p \rangle$

shows $\langle \exists A. \text{ set } A \subseteq X \wedge A \vdash_\square p \rangle$

proof (*rule ccontr*)

assume $\langle \nexists A. \text{ set } A \subseteq X \wedge A \vdash_\square p \rangle$

then have $*$: $\langle \forall A. \text{ set } A \subseteq \{\neg p\} \cup X \longrightarrow \neg A \vdash_\square \perp \rangle$

using *K-Boole botE* **by** (*metis derive-split1 insert-is-Un subset-insert*)

let $?X = \langle \{\neg p\} \cup X \rangle$

let $?S = \langle \text{Extend } ?X \rangle$

have $\langle \text{consistent } ?X \rangle$

using $*$ **unfolding** *consistent-def* .

then have $\langle \text{MCS } ?S \rangle$

using *MCS-Extend'* **by** *blast*

moreover have $\langle \neg p \in ?S \rangle \langle X \subseteq ?S \rangle$

using *Extend-subset* **by** *fast+*

ultimately have $\langle \mathcal{M}_\square ?S \models_\square (\neg p) \rangle \langle \forall q \in X. \mathcal{M}_\square ?S \models_\square q \rangle$

using *assms Truth-lemma* **by** *fast+*

then have $\langle \mathcal{M}_\square ?S \models_\square p \rangle$

using *assms* $\langle \text{MCS } ?S \rangle$ **by** *simp*

then show *False*

using $\langle \mathcal{M}_\square ?S \models_\square (\neg p) \rangle$ **by** *simp*

qed

abbreviation *valid* $:: \langle ('i, 'p) \text{ fm} \Rightarrow \text{bool} \rangle$ **where**

$\langle \text{valid } p \equiv \forall (M :: ('i, 'p, ('i, 'p) \text{ fm set}) \text{ model}). \forall w \in W M. (M, w) \models_\square p \rangle$

corollary *completeness*: $\langle \text{valid } p \implies \vdash_\square p \rangle$

using *strong-completeness* [**where** $X = \langle \{\} \rangle$] **by** *simp*

theorem *main*: $\langle \text{valid } p \iff \vdash_\square p \rangle$

using *soundness completeness* **by** *meson*

end

Chapter 7

Example: Hybrid Logic

theory *Example-Hybrid-Logic* imports *Derivations* begin

7.1 Syntax

datatype (*nominals-fm*: 'i, 'p) *fm*
= *Fls* (\perp)
| *Pro* 'p ($\langle \cdot \rangle$)
| *Nom* 'i ($\langle \cdot \rangle$)
| *Imp* $\langle ('i, 'p) \text{ fm} \rangle \langle ('i, 'p) \text{ fm} \rangle$ (**infix** $\langle \longrightarrow \rangle$ 55)
| *Dia* $\langle ('i, 'p) \text{ fm} \rangle \langle \langle \cdot \rangle \rangle$
| *Sat* 'i $\langle ('i, 'p) \text{ fm} \rangle \langle \langle \textcircled{\cdot} \rangle \rangle$
| *All* $\langle ('i, 'p) \text{ fm} \rangle \langle \langle \mathbf{A} \rangle \rangle$

abbreviation *Neg* ($\langle \neg \rightarrow [70] 70 \rangle$) **where** $\langle \neg p \equiv p \longrightarrow \perp \rangle$

abbreviation *Con* (**infix** $\langle \wedge \rangle$ 35) **where** $\langle p \wedge q \equiv \neg (p \longrightarrow \neg q) \rangle$

type-synonym ('i, 'p) *lbd* = $\langle 'i \times ('i, 'p) \text{ fm} \rangle$

primrec *nominals-lbd* :: $\langle ('i, 'p) \text{ lbd set} \Rightarrow 'i \text{ set} \rangle$ **where**
 $\langle \text{nominals-lbd } (i, p) = \{i\} \cup \text{nominals-fm } p \rangle$

abbreviation *nominals* :: $\langle ('i, 'p) \text{ lbd set} \Rightarrow 'i \text{ set} \rangle$ **where**
 $\langle \text{nominals } S \equiv \bigcup ip \in S. \text{nominals-lbd } ip \rangle$

lemma *finite-nominals-fm* [*simp*]: $\langle \text{finite } (\text{nominals-fm } p) \rangle$
by (*induct* *p*) *simp-all*

lemma *finite-nominals-lbd*: $\langle \text{finite } (\text{nominals-lbd } p) \rangle$
by (*cases* *p*) *simp*

7.2 Semantics

datatype ('w, 'p) *model* =
Model (*W*: $\langle 'w \text{ set} \rangle$) (*R*: $\langle 'w \Rightarrow 'w \text{ set} \rangle$) (*V*: $\langle 'w \Rightarrow 'p \Rightarrow \text{bool} \rangle$)

type-synonym ('i, 'p, 'w) *ctx* = $\langle ('w, 'p) \text{ model} \times ('i \Rightarrow 'w) \times 'w \rangle$

fun *semantics* :: $\langle ('i, 'p, 'w) \text{ ctx} \Rightarrow ('i, 'p) \text{ fm} \Rightarrow \text{bool} \rangle$ (**infix** $\langle \models_{\textcircled{\cdot}} \rangle$ 50) **where**
 $\langle (M, g, w) \models_{\textcircled{\cdot}} \perp \longleftrightarrow \text{False} \rangle$

$\langle (M, -, w) \models_{\mathbb{Q}} \cdot P \longleftrightarrow V M w P \rangle$
 $\langle (-, g, w) \models_{\mathbb{Q}} \cdot i \longleftrightarrow w = g i \rangle$
 $\langle (M, g, w) \models_{\mathbb{Q}} p \longrightarrow q \longleftrightarrow (M, g, w) \models_{\mathbb{Q}} p \longrightarrow (M, g, w) \models_{\mathbb{Q}} q \rangle$
 $\langle (M, g, w) \models_{\mathbb{Q}} \diamond p \longleftrightarrow (\exists v \in W M \cap R M w. (M, g, v) \models_{\mathbb{Q}} p) \rangle$
 $\langle (M, g, -) \models_{\mathbb{Q}} @i p \longleftrightarrow (M, g, g i) \models_{\mathbb{Q}} p \rangle$
 $\langle (M, g, -) \models_{\mathbb{Q}} \mathbf{A} p \longleftrightarrow (\forall v \in W M. (M, g, v) \models_{\mathbb{Q}} p) \rangle$

lemma semantics-fresh: $\langle i \notin \text{nominals-fm } p \implies (M, g, w) \models_{\mathbb{Q}} p \longleftrightarrow (M, g(i := v), w) \models_{\mathbb{Q}} p \rangle$
by (*induct p arbitrary: w*) *auto*

lemma semantics-fresh-lbd:

$\langle k \notin \text{nominals-lbd } (i, p) \implies (M, g, w) \models_{\mathbb{Q}} p \longleftrightarrow (M, g(k := v), w) \models_{\mathbb{Q}} p \rangle$
by (*induct p arbitrary: w*) *auto*

7.3 Calculus

inductive Calculus :: $\langle ('i, 'p) \text{ lbd list} \implies ('i, 'p) \text{ lbd} \implies \text{bool} \rangle$ (**infix** $\langle \vdash_{\mathbb{Q}} \rangle$ 50) **where**

Assm [simp]: $\langle (i, p) \in \text{set } A \implies A \vdash_{\mathbb{Q}} (i, p) \rangle$
Ref [simp]: $\langle A \vdash_{\mathbb{Q}} (i, \cdot i) \rangle$
Nom [dest]: $\langle A \vdash_{\mathbb{Q}} (i, \cdot k) \implies A \vdash_{\mathbb{Q}} (i, p) \implies A \vdash_{\mathbb{Q}} (k, p) \rangle$
FlsE [elim]: $\langle A \vdash_{\mathbb{Q}} (i, \perp) \implies A \vdash_{\mathbb{Q}} (k, p) \rangle$
ImpI [intro]: $\langle (i, p) \# A \vdash_{\mathbb{Q}} (i, q) \implies A \vdash_{\mathbb{Q}} (i, p \longrightarrow q) \rangle$
ImpE [dest]: $\langle A \vdash_{\mathbb{Q}} (i, p \longrightarrow q) \implies A \vdash_{\mathbb{Q}} (i, p) \implies A \vdash_{\mathbb{Q}} (i, q) \rangle$
SatI [intro]: $\langle A \vdash_{\mathbb{Q}} (i, p) \implies A \vdash_{\mathbb{Q}} (k, @i p) \rangle$
SatE [dest]: $\langle A \vdash_{\mathbb{Q}} (i, @k p) \implies A \vdash_{\mathbb{Q}} (k, p) \rangle$
DiaI [intro]: $\langle A \vdash_{\mathbb{Q}} (i, \diamond (\cdot k)) \implies A \vdash_{\mathbb{Q}} (k, p) \implies A \vdash_{\mathbb{Q}} (i, \diamond p) \rangle$
DiaE [elim]: $\langle A \vdash_{\mathbb{Q}} (i, \diamond p) \implies k \notin \text{nominals } (\{(i, p), (j, q)\} \cup \text{set } A) \implies$
 $(k, p) \# (i, \diamond (\cdot k)) \# A \vdash_{\mathbb{Q}} (j, q) \implies A \vdash_{\mathbb{Q}} (j, q) \rangle$
AllI [intro]: $\langle A \vdash_{\mathbb{Q}} (k, p) \implies k \notin \text{nominals } (\{(i, p)\} \cup \text{set } A) \implies A \vdash_{\mathbb{Q}} (i, \mathbf{A} p) \rangle$
Alle [dest]: $\langle A \vdash_{\mathbb{Q}} (i, \mathbf{A} p) \implies A \vdash_{\mathbb{Q}} (k, p) \rangle$
Clas: $\langle (i, p \longrightarrow q) \# A \vdash_{\mathbb{Q}} (i, p) \implies A \vdash_{\mathbb{Q}} (i, p) \rangle$
Cut: $\langle A \vdash_{\mathbb{Q}} (k, q) \implies (k, q) \# B \vdash_{\mathbb{Q}} (i, p) \implies A @ B \vdash_{\mathbb{Q}} (i, p) \rangle$

7.4 Soundness

theorem soundness: $\langle A \vdash_{\mathbb{Q}} (i, p) \implies \text{list-all } (\lambda(i, p). (M, g, g i) \models_{\mathbb{Q}} p) A \implies \text{range } g \subseteq W M \implies$
 $(M, g, g i) \models_{\mathbb{Q}} p \rangle$

proof (*induct* $\langle (i, p) \rangle$ *arbitrary: i p g rule: Calculus.induct*)

case (*Nom A i k p*)

then show *?case*

by (*metis semantics.simps(3)*)

next

case (*DiaE A i p k j q*)

then have $\langle (M, g, g i) \models_{\mathbb{Q}} \diamond p \rangle$

by *blast*

then obtain *v* **where** *v*: $\langle v \in W M \cap R M (g i) \rangle \langle (M, g, v) \models_{\mathbb{Q}} p \rangle$

by *auto*

let *?g* = $\langle g(k := v) \rangle$

have $\langle (M, ?g, ?g k) \models_{\mathbb{Q}} p \rangle \langle (M, ?g, ?g i) \models_{\mathbb{Q}} \diamond (\cdot k) \rangle$

using *v fun-upd-same DiaE(3) semantics-fresh-lbd* **by** *fastforce+*

moreover have $\langle \text{list-all } (\lambda(i, p). (M, ?g, ?g i) \models_{\mathbb{Q}} p) A \rangle$

using *DiaE.prem(1) DiaE.hyps(3) semantics-fresh-lbd* **by** (*fastforce simp: list-all-iff*)

ultimately have $\langle \text{list-all } (\lambda(i, p). (M, ?g, ?g i) \models_{\mathbb{Q}} p) ((k, p) \# (i, \diamond (\cdot k)) \# A) \rangle$

by *simp*

moreover have $\langle \text{range } ?g \subseteq W M \rangle$

```

    using DiaE.premis v by auto
  ultimately have  $\langle (M, ?g, ?g j) \models_{\text{@}} q \rangle$ 
    using DiaE.hyps by blast
  then show ?case
    using DiaE.hyps(3) semantics-fresh-lbd by fastforce
next
case (AllI A k p i)
{
  fix v
  assume  $\langle v \in W M \rangle$ 
  let  $?g = \langle g(k := v) \rangle$ 
  have  $\langle \forall v. \text{list-all } (\lambda(i, p). (M, ?g, ?g i) \models_{\text{@}} p) A \rangle$ 
    using AllI.premis(1) AllI.hyps(3) semantics-fresh-lbd by (fastforce simp: list-all-iff)
  moreover have  $\langle \text{range } ?g \subseteq W M \rangle$ 
    using AllI.premis  $\langle v \in W M \rangle$  by auto
  ultimately have  $\langle (M, ?g, ?g k) \models_{\text{@}} p \rangle$ 
    using AllI.hyps by fast
}
then have  $\langle \forall v \in W M. (M, g(k := v), v) \models_{\text{@}} p \rangle$ 
  by simp
then have  $\langle \forall v \in W M. (M, g, v) \models_{\text{@}} p \rangle$ 
  using AllI.hyps(3) semantics-fresh-lbd by fast
then show ?case
  by simp
next
case (AllE A i p k)
then show ?case
  by fastforce
qed (auto simp: list-all-iff)

```

corollary *soundness'*:

```

  assumes  $\langle [] \vdash_{\text{@}} (i, p) \rangle \langle i \notin \text{nominals-fm } p \rangle$ 
  and  $\langle \text{range } g \subseteq W M \rangle \langle w \in W M \rangle$ 
  shows  $\langle (M, g, w) \models_{\text{@}} p \rangle$ 

```

proof –

```

  let  $?g = \langle g(i := w) \rangle$ 
  from assms(1) have  $\langle (M, ?g, ?g i) \models_{\text{@}} p \rangle$ 
    by (rule soundness) (use assms(3–4) in auto)
  then have  $\langle (M, ?g, w) \models_{\text{@}} p \rangle$ 
    by simp
  then show ?thesis
    using assms(2) semantics-fresh by fast

```

qed

corollary $\langle \neg (\ [] \vdash_{\text{@}} (i, \perp)) \rangle$

by (metis list.pred-inject(1) model.sel(1) semantics.simps(1) soundness subset-refl)

7.5 Admissible Rules

lemma *Assm-head* [simp]: $\langle (p, i) \# A \vdash_{\text{@}} (p, i) \rangle$
by auto

lemma *SatE'*:

```

  assumes  $\langle (k, q) \# A \vdash_{\text{@}} (i, p) \rangle$ 
  shows  $\langle (j, @k q) \# A \vdash_{\text{@}} (i, p) \rangle$ 

```

proof –
 have $\langle (j, @k\ q) \vdash_{@} (k, q) \rangle$
 by (*meson Assm-head SatE*)
 then show *?thesis*
 using *assms* by (*auto dest: Cut*)
qed

lemma *ImpI'*:
 assumes $\langle (k, q) \# A \vdash_{@} (i, p) \rangle$
 shows $\langle A \vdash_{@} (i, (@k\ q) \longrightarrow p) \rangle$
 using *assms SatE'* by *fast*

lemma *Weak'*: $\langle A \vdash_{@} (i, p) \Longrightarrow A @ B \vdash_{@} (i, p) \rangle$
 by (*simp add: Cut*)

lemma *Weaken*: $\langle A \vdash_{@} (i, p) \Longrightarrow \text{set } A \subseteq \text{set } B \Longrightarrow B \vdash_{@} (i, p) \rangle$

proof (*induct A arbitrary: p*)

case *Nil*
 then show *?case*
 by (*metis Weak' append-Nil*)

next

case (*Cons kq A*)
 then show *?case*
proof (*cases kq*)
 case (*Pair k q*)
 then have $\langle B \vdash_{@} (i, @k\ q \longrightarrow p) \rangle$
 using *Cons* by (*simp add: ImpI'*)
 then show *?thesis*
 using *Pair Cons(3)* by *fastforce*

qed

qed

lemma *Weak*: $\langle A \vdash_{@} (i, p) \Longrightarrow (k, q) \# A \vdash_{@} (i, p) \rangle$
 using *Weaken*[**where** $A=A$ **and** $B=\langle (k, q) \# A \rangle$] by *auto*

lemma *deduct1*: $\langle A \vdash_{@} (i, p \longrightarrow q) \Longrightarrow (i, p) \# A \vdash_{@} (i, q) \rangle$
 by (*meson ImpE Weak Assm-head*)

lemma *Boole*: $\langle (i, \neg p) \# A \vdash_{@} (i, \perp) \Longrightarrow A \vdash_{@} (i, p) \rangle$
 using *Clas FlsE* by *meson*

interpretation *Derivations Calculus*

proof

fix *A* and *p* :: $\langle ('i, 'p) \text{ lbd} \rangle$
 show $\langle p \in \text{set } A \Longrightarrow A \vdash_{@}(p) \rangle$
 by (*cases p*) *simp*

next

fix *A B* and *p* :: $\langle ('i, 'p) \text{ lbd} \rangle$
 assume $\langle A \vdash_{@}(p) \rangle \langle \text{set } A = \text{set } B \rangle$
 then show $\langle B \vdash_{@}(p) \rangle$
 by (*cases p*) (*simp add: Weaken*)

qed

7.6 Maximal Consistent Sets

definition *consistent* :: $\langle (i, p) \text{ lbd set} \Rightarrow \text{bool} \rangle$ **where**
 $\langle \text{consistent } S \equiv \forall A \ a. \ \text{set } A \subseteq S \longrightarrow \neg A \vdash_{\text{@}} (a, \perp) \rangle$

lemma *consistent-add-diamond-witness*:

assumes $\langle \text{consistent } S \rangle \langle (i, \diamond p) \in S \rangle \langle k \notin \text{nominals } S \rangle$
shows $\langle \text{consistent } (\{(k, p), (i, \diamond (\cdot k))\} \cup S) \rangle$
unfolding *consistent-def*
proof *safe*
fix $A \ a$
assume $A: \langle \text{set } A \subseteq \{(k, p), (i, \diamond (\cdot k))\} \cup S \rangle \langle A \vdash_{\text{@}} (a, \perp) \rangle$
then obtain $A' \ a \ B$ **where** $\langle \text{set } A' \subseteq S \rangle \langle B \ @ \ A' \vdash_{\text{@}} (a, \perp) \rangle \langle \text{set } B = \{(k, p), (i, \diamond (\cdot k))\} \cap \text{set } A \rangle$
using *assms derive-split* [**where** $p = \langle (a, \perp) \rangle$ **and** $X = S$ **and** $B = \langle [(k, p), (i, \diamond (\cdot k))] \rangle$]
by (*metis Int-commute empty-set list.simps(15)*)
then have $\langle (k, p) \# (i, \diamond (\cdot k)) \# A' \vdash_{\text{@}} (a, \perp) \rangle$
by (*auto intro: Weaken*)
then have $\langle (k, p) \# (i, \diamond (\cdot k)) \# A' \vdash_{\text{@}} (i, \perp) \rangle$
by *fast*
then have $\langle (k, p) \# (i, \diamond (\cdot k)) \# (i, \diamond p) \# A' \vdash_{\text{@}} (i, \perp) \rangle$
by (*fastforce intro: Weaken*)
moreover have $\langle k \notin \text{nominals } (\{(i, p), (i, \perp)\} \cup \text{set } ((i, \diamond p) \# A')) \rangle$
using $\langle \text{set } A' \subseteq S \rangle$ *assms(2-3)* **by** *auto*
moreover have $\langle (i, \diamond p) \# A' \vdash_{\text{@}} (i, \diamond p) \rangle$
by *auto*
ultimately have $\langle (i, \diamond p) \# A' \vdash_{\text{@}} (i, \perp) \rangle$
by *fast*
moreover have $\langle \text{set } ((i, \diamond p) \# A') \subseteq S \rangle$
using $\langle \text{set } A' \subseteq S \rangle$ *assms(2)* **by** *simp*
ultimately show *False*
using *assms(1)* **unfolding** *consistent-def* **by** *blast*
qed

lemma *consistent-add-global-witness*:

assumes $\langle \text{consistent } S \rangle \langle (i, \neg \mathbf{A} \ p) \in S \rangle \langle k \notin \text{nominals } S \rangle$
shows $\langle \text{consistent } (\{(k, \neg p)\} \cup S) \rangle$
unfolding *consistent-def*
proof *safe*
fix $A \ a$
assume $\langle \text{set } A \subseteq \{(k, \neg p)\} \cup S \rangle \langle A \vdash_{\text{@}} (a, \perp) \rangle$
then obtain A' **where** $\langle \text{set } A' \subseteq S \rangle \langle (k, \neg p) \# A' \vdash_{\text{@}} (a, \perp) \rangle$
using *assms derive-split1* **by** (*metis consistent-def insert-is-Un subset-insert*)
then have $\langle (k, \neg p) \# A' \vdash_{\text{@}} (k, \perp) \rangle$
by *fast*
then have $\langle A' \vdash_{\text{@}} (k, p) \rangle$
by (*meson Boole*)
moreover have $\langle k \notin \text{nominals } (\{(i, p), (i, \perp)\} \cup \text{set } ((i, \mathbf{A} \ p) \# A')) \rangle$
using $\langle \text{set } A' \subseteq S \rangle$ *assms(2-3)* **by** *auto*
ultimately have $\langle A' \vdash_{\text{@}} (i, \mathbf{A} \ p) \rangle$
by *fastforce*
then have $\langle (i, \neg \mathbf{A} \ p) \# A' \vdash_{\text{@}} (i, \perp) \rangle$
by (*meson Assm-head ImpE Weak*)
moreover have $\langle \text{set } ((i, \neg \mathbf{A} \ p) \# A') \subseteq S \rangle$
using $\langle \text{set } A' \subseteq S \rangle$ *assms(2)* **by** *simp*
ultimately show *False*
using *assms(1)* **unfolding** *consistent-def* **by** *blast*

qed

fun *witness* :: $\langle ('i, 'p) \text{ lbd} \Rightarrow ('i, 'p) \text{ lbd set} \Rightarrow ('i, 'p) \text{ lbd set} \rangle$ **where**
 $\langle \text{witness } (i, \diamond p) S = (\text{let } k = \text{SOME } k. k \notin \text{nominals } (\{(i, p)\} \cup S) \text{ in } \{(k, p), (i, \diamond (\cdot k))\}) \rangle$
 $| \langle \text{witness } (i, \neg \mathbf{A} p) S = (\text{let } k = \text{SOME } k. k \notin \text{nominals } (\{(i, p)\} \cup S) \text{ in } \{(k, \neg p)\}) \rangle$
 $| \langle \text{witness } (-, -) - = \{\} \rangle$

lemma *consistent-witness'*:

assumes $\langle \text{consistent } (\{(i, p)\} \cup S) \rangle$ $\langle \text{infinite } (\text{UNIV} - \text{nominals } S) \rangle$
shows $\langle \text{consistent } (\text{witness } (i, p) S \cup \{(i, p)\} \cup S) \rangle$
using *assms*

proof (*induct* $\langle (i, p) \rangle$ *S arbitrary: i p rule: witness.induct*)

case $(1 \ i \ p \ S)$

have $\langle \text{infinite } (\text{UNIV} - \text{nominals } (\{(i, p)\} \cup S)) \rangle$

using $1(2)$ *finite-nominals-lbd*

by (*metis UN-Un finite.emptyI finite.insertI finite-UN-I infinite-Diff-fin-Un*)

then have $\langle \exists k. k \notin \text{nominals } (\{(i, p)\} \cup S) \rangle$

by (*simp add: not-finite-existsD set-diff-eq*)

then have $\langle (\text{SOME } k. k \notin \text{nominals } (\{(i, p)\} \cup S)) \notin \text{nominals } (\{(i, p)\} \cup S) \rangle$

by (*rule someI-ex*)

then obtain *k* **where** $\langle \text{witness } (i, \diamond p) S = \{(k, p), (i, \diamond (\cdot k))\} \rangle$

$\langle k \notin \text{nominals } (\{(i, \diamond p)\} \cup S) \rangle$

by (*simp add: Let-def*)

then show *?case*

using $1(1)$ *consistent-add-diamond-witness* [**where** $S = \{(i, \diamond p)\} \cup S$] **by** *simp*

next

case $(2 \ i \ p \ S)$

have $\langle \text{infinite } (\text{UNIV} - \text{nominals } (\{(i, p)\} \cup S)) \rangle$

using $2(2)$ *finite-nominals-lbd*

by (*metis UN-Un finite.emptyI finite.insertI finite-UN-I infinite-Diff-fin-Un*)

then have $\langle \exists k. k \notin \text{nominals } (\{(i, p)\} \cup S) \rangle$

by (*simp add: not-finite-existsD set-diff-eq*)

then have $\langle (\text{SOME } k. k \notin \text{nominals } (\{(i, p)\} \cup S)) \notin \text{nominals } (\{(i, p)\} \cup S) \rangle$

by (*rule someI-ex*)

then obtain *k* **where** $\langle \text{witness } (i, \neg \mathbf{A} p) S = \{(k, \neg p)\} \rangle$ $\langle k \notin \text{nominals } (\{(i, \neg \mathbf{A} p)\} \cup S) \rangle$

by (*simp add: Let-def*)

then show *?case*

using $2(1)$ *consistent-add-global-witness* [**where** $S = \{(i, \neg \mathbf{A} p)\} \cup S$] **by** *auto*

qed (*auto simp: assms*)

interpretation *MCS-Witness-UNIV consistent witness nominals-lbd*

proof

fix *ip* :: $\langle ('i, 'p) \text{ lbd} \rangle$ **and** *S* :: $\langle ('i, 'p) \text{ lbd set} \rangle$

show $\langle \text{finite } (\text{nominals } (\text{witness } ip \ S)) \rangle$

by (*induct ip S rule: witness.induct*) (*auto simp: Let-def*)

next

fix *ip* **and** *S* :: $\langle ('i, 'p) \text{ lbd set} \rangle$

assume $\langle \text{consistent } (\{ip\} \cup S) \rangle$ $\langle \text{infinite } (\text{UNIV} - \text{nominals } S) \rangle$

then show $\langle \text{consistent } (\{ip\} \cup S \cup \text{witness } ip \ S) \rangle$

using *consistent-witness'* **by** (*cases ip*) (*simp add: sup-commute*)

next

have $\langle \text{infinite } (\text{UNIV} :: ('i, 'p) \text{ fm set}) \rangle$

using *infinite-UNIV-size* [*of* $\langle \diamond \rangle$] **by** *simp*

then show $\langle \text{infinite } (\text{UNIV} :: ('i, 'p) \text{ lbd set}) \rangle$

using *finite-prod* **by** *blast*

qed (*auto simp: consistent-def*)

lemma *witnessed-diamond*: $\langle \text{witnessed } S \implies (i, \diamond p) \in S \implies \exists k. (i, \diamond (\cdot k)) \in S \wedge (k, p) \in S \rangle$
unfolding *witnessed-def* **by** (*metis insert-subset witness.simps(1)*)

lemma *witnessed-global*: $\langle \text{witnessed } S \implies (i, \neg \mathbf{A} p) \in S \implies \exists k. (k, \neg p) \in S \rangle$
unfolding *witnessed-def* **by** (*metis insert-subset witness.simps(2)*)

interpretation *Derivations-Cut-MCS consistent Calculus*

proof

show $\langle \bigwedge S. \text{consistent } S = (\forall A. \text{set } A \subseteq S \longrightarrow (\exists q. \neg A \vdash_{\text{@}}(q))) \rangle$

unfolding *consistent-def* **using** *FlsE* **by** *fast*

next

fix $A B$ **and** $p q :: \langle ('i, 'p) \text{ lbd} \rangle$

assume $\langle A \vdash_{\text{@}}(p) \rangle \langle p \# B \vdash_{\text{@}}(q) \rangle$

then show $\langle A @ B \vdash_{\text{@}}(q) \rangle$

by (*cases p, cases q*) (*meson Cut*)

qed

interpretation *Derivations-Bot consistent Calculus* $\langle (i, \perp) \rangle$

proof *qed auto*

interpretation *Derivations-Not consistent Calculus* $\langle (i, \perp) \rangle \langle \lambda(i, p). (i, \neg p) \rangle$

proof *qed auto*

lemma *MCS-impE'*: $\langle \text{consistent } S \implies \text{maximal } S \implies (i, p \longrightarrow q) \in S \implies (i, p) \in S \longrightarrow (i, q) \in S \rangle$
by (*metis MCS-derive deduct1 insert-subset list.simps(15)*)

interpretation *Derivations-Uni consistent witness nominals-lbd Calculus* $\langle (i, \perp) \rangle \langle \lambda(i, p). (i, \neg p) \rangle$
 $\langle \lambda(i, p). (i, \mathbf{A} p) \rangle \langle \lambda k (i, p). (k, p) \rangle$

proof

have $\langle \bigwedge S S' i p. \text{MCS } S \implies \text{witness } (i, \neg \mathbf{A} p) S' \subseteq S \implies \exists k. (k, \neg p) \in S \rangle$

by *auto*

then show $\langle \bigwedge S S' p. \text{MCS } S \implies$

$\text{witness } (\text{case case } p \text{ of } (i, p) \Rightarrow (i, \mathbf{A} p) \text{ of } (i, p) \Rightarrow (i, \neg p)) S' \subseteq S \implies$

$\exists t. (\text{case case } p \text{ of } (i, p) \Rightarrow (t, p) \text{ of } (i, p) \Rightarrow (i, \neg p)) \in S \rangle$

by *fast*

next

have $\langle \bigwedge A i p k. A \vdash_{\text{@}} (i, \mathbf{A} p) \implies A \vdash_{\text{@}} (k, p) \rangle$

..

then show $\langle \bigwedge A p t. A \vdash_{\text{@}} (\text{case } p \text{ of } (i, p) \Rightarrow (i, \mathbf{A} p)) \implies A \vdash_{\text{@}} (\text{case } p \text{ of } (i, p) \Rightarrow (t, p)) \rangle$

by *fast*

qed

lemma *conE1* [*elim*]: $\langle A \vdash_{\text{@}} (i, p \wedge q) \implies A \vdash_{\text{@}} (i, p) \rangle$
by (*meson Clas FlsE deduct1*)

lemma *conE2* [*elim*]: $\langle A \vdash_{\text{@}} (i, p \wedge q) \implies A \vdash_{\text{@}} (i, q) \rangle$
by (*meson Assm-head Boole ImpE ImpI Weak*)

lemma *conI* [*intro*]: $\langle A \vdash_{\text{@}} (i, p) \implies A \vdash_{\text{@}} (i, q) \implies A \vdash_{\text{@}} (i, p \wedge q) \rangle$
by (*meson Assm-head ImpE ImpI Weak*)

lemma *MCS-con*:

assumes $\langle \text{MCS } S \rangle$

shows $\langle (i, p \wedge q) \in S \longleftrightarrow (i, p) \in S \wedge (i, q) \in S \rangle$

using *assms MCS-derive conE1 conE2*

by (metis Boole MCS-explode MCS-impE' derive-assm list.set-intros(1))

interpretation *Derivations-Exi consistent witness nominals-lbd Calculus*

$\langle \lambda(i, p). (i, \diamond p) \rangle \langle \lambda k (i, p). (i, @k p \wedge \diamond(\cdot k)) \rangle$

proof

have $\langle \bigwedge S S' i p. MCS S \implies witness (i, \diamond p) S' \subseteq S \implies \exists k. (i, @k p \wedge \diamond(\cdot k)) \in S \rangle$

unfolding *witness.simps* using *MCS-con* by (metis *MCS-derive SatI insert-subset*)

then show $\langle \bigwedge S S' p. MCS S \implies witness (case p of (i, p) \Rightarrow (i, \diamond p)) S' \subseteq S \implies \exists t. (case p of (i, p) \Rightarrow (i, @t p \wedge \diamond(\cdot t))) \in S \rangle$

by *fast*

next

have $\langle \bigwedge A i k p. A \vdash_{@} (i, @k p \wedge \diamond(\cdot k)) \implies A \vdash_{@} (i, \diamond p) \rangle$

by (metis *DiaI SatE conE1 conE2*)

then show $\langle \bigwedge A p t. A \vdash_{@} (case p of (i, p) \Rightarrow (i, @t p \wedge \diamond(\cdot t))) \implies A \vdash_{@} (case p of (i, p) \Rightarrow (i, \diamond p)) \rangle$

by *fast*

qed

corollary *MCS-uni'*:

assumes $\langle MCS S \rangle \langle witnessed S \rangle$

shows $\langle (i, \mathbf{A} p) \in S \longleftrightarrow (\forall k. (k, p) \in S) \rangle$

using *assms MCS-uni* by *fastforce*

corollary *MCS-exi'*:

assumes $\langle MCS S \rangle \langle witnessed S \rangle$

shows $\langle (i, \diamond p) \in S \longleftrightarrow (\exists k. (i, @k p \wedge \diamond(\cdot k)) \in S) \rangle$

using *assms MCS-exi* by *fastforce*

7.7 Nominals

lemma *MCS-Nom-refl*:

assumes $\langle consistent S \rangle \langle maximal S \rangle$

shows $\langle (i, \cdot i) \in S \rangle$

using *assms Ref* by (metis *MCS-derive MCS-explode*)

lemma *MCS-Nom-sym*:

assumes $\langle consistent S \rangle \langle maximal S \rangle \langle (i, \cdot k) \in S \rangle$

shows $\langle (k, \cdot i) \in S \rangle$

using *assms Nom Ref* by (metis *MCS-derive*)

lemma *MCS-Nom-trans*:

assumes $\langle consistent S \rangle \langle maximal S \rangle \langle (i, \cdot j) \in S \rangle \langle (j, \cdot k) \in S \rangle$

shows $\langle (i, \cdot k) \in S \rangle$

proof –

have $\langle [(i, \cdot j), (j, \cdot k)] \vdash_{@} (i, \cdot j) \rangle \langle [(i, \cdot j), (j, \cdot k)] \vdash_{@} (j, \cdot k) \rangle$

by *simp-all*

then have $\langle [(i, \cdot j), (j, \cdot k)] \vdash_{@} (i, \cdot k) \rangle$

using *Nom Ref* by *metis*

then show *?thesis*

using *assms MCS-derive*

by (metis *bot.extremum insert-subset list.set(1) list.simps(15)*)

qed

7.8 Truth Lemma

```

fun semics :: ⟨('i, 'p, 'w) ctx ⇒ (('i, 'p, 'w) ctx ⇒ ('i, 'p) fm ⇒ bool) ⇒ ('i, 'p) fm ⇒ bool⟩
  (⟨(- [[-]]@ -) [55, 0, 55] 55) where
  ⟨- [[-]]@ ⊥ ⟷ False⟩
| ⟨(M, -, w) [[-]]@ ·P ⟷ V M w P⟩
| ⟨(-, g, w) [[-]]@ ·i ⟷ w = g i⟩
| ⟨(M, g, w) [[R]]@ (p) ⟶ q ⟷ R (M, g, w) p ⟶ R (M, g, w) q⟩
| ⟨(M, g, w) [[R]]@ ◇ p ⟷ (∃ v ∈ W M ∩ R M w. R (M, g, v) p)⟩
| ⟨(M, g, -) [[R]]@ @i p ⟷ R (M, g, g i) p⟩
| ⟨(M, g, -) [[R]]@ A p ⟷ (∀ v ∈ W M. R (M, g, v) p)⟩

```

```

fun rel :: ⟨('i, 'p) lbd set ⇒ ('i, 'p, 'i) ctx ⇒ ('i, 'p) fm ⇒ bool⟩ (⟨R@⟩) where
  ⟨R@(S) (-, -, i) p ⟷ (i, p) ∈ S⟩

```

```

definition equiv-nom :: ⟨('i, 'p) lbd set ⇒ 'i ⇒ 'i ⇒ bool⟩ where
  ⟨equiv-nom S i k ≡ (i, ·k) ∈ S⟩

```

```

lemma equiv-nom-reflp:
  assumes ⟨consistent S⟩ ⟨maximal S⟩
  shows ⟨reflp (equiv-nom S)⟩
  unfolding equiv-nom-def reflp-def using assms MCS-Nom-refl by fast

```

```

lemma equiv-nom-symp:
  assumes ⟨consistent S⟩ ⟨maximal S⟩
  shows ⟨symp (equiv-nom S)⟩
  unfolding equiv-nom-def symp-def using assms MCS-Nom-sym by fast

```

```

lemma equiv-nom-transp:
  assumes ⟨consistent S⟩ ⟨maximal S⟩
  shows ⟨transp (equiv-nom S)⟩
  unfolding equiv-nom-def transp-def using assms MCS-Nom-trans by fast

```

```

lemma equiv-nom-equivp:
  assumes ⟨consistent S⟩ ⟨maximal S⟩
  shows ⟨equivp (equiv-nom S)⟩
  using assms by (simp add: equivpI equiv-nom-reflp equiv-nom-symp equiv-nom-transp)

```

```

definition assign :: ⟨'i ⇒ ('i, 'p) lbd set ⇒ 'i⟩ (⟨[-]⟩ [0, 100] 100) where
  ⟨[i]_S ≡ minim ( |UNIV| ) {k. equiv-nom S i k}⟩

```

```

lemma equiv-nom-ne:
  assumes ⟨consistent S⟩ ⟨maximal S⟩
  shows ⟨{k. equiv-nom S i k} ≠ {}⟩
  unfolding equiv-nom-def using assms MCS-Nom-refl by fast

```

```

lemma equiv-nom-assign:
  assumes ⟨consistent S⟩ ⟨maximal S⟩
  shows ⟨equiv-nom S i ([i]_S)⟩
  unfolding assign-def using assms equiv-nom-ne wo-rel.minim-in
  by (metis Field-card-of card-of-well-order-on mem-Collect-eq top.extremum wo-rel-def)

```

```

lemma equiv-nom-Nom:
  assumes ⟨consistent S⟩ ⟨maximal S⟩ ⟨equiv-nom S i k⟩ ⟨(i, p) ∈ S⟩
  shows ⟨(k, p) ∈ S⟩
proof -

```

have $\langle [(i, \cdot k), (i, p)] \vdash_{\text{@}} (k, p) \rangle$
by (*meson Assm-head Nom Weak*)
then show *?thesis*
using *assms MCS-derive unfolding equiv-nom-def by force*
qed

definition *reach* :: $\langle ('i, 'p) \text{ lbd set} \Rightarrow 'i \Rightarrow 'i \text{ set} \rangle$ **where**
 $\langle \text{reach } S \ i \equiv \{[k]_S \mid k. (i, \diamond (\cdot k)) \in S\} \rangle$

primrec *canonical* :: $\langle ('i, 'p) \text{ lbd set} \times 'i \Rightarrow ('i, 'p, 'i) \text{ ctx} \rangle$ ($\langle \mathcal{M}_{\text{@}} \rangle$) **where**
 $\langle \mathcal{M}_{\text{@}}(S, i) = (\text{Model } \{[k]_S \mid k. \text{True}\} (\text{reach } S) (\lambda i P. (i, \cdot P) \in S), \lambda i. [i]_S, [i]_S) \rangle$

theorem *saturated-model*:

assumes $\langle \bigwedge p \ i. \mathcal{M}_{\text{@}}(S, i) \llbracket \mathcal{R}_{\text{@}}(S) \rrbracket_{\text{@}}(p) \longleftrightarrow \mathcal{R}_{\text{@}}(S) (\mathcal{M}_{\text{@}}(S, i)) \ p \rangle$ $\langle M \in \{\mathcal{M}_{\text{@}}(S, i) \mid i. \text{True}\} \rangle$
shows $\langle \mathcal{R}_{\text{@}}(S) (\mathcal{M}_{\text{@}}(S, i)) \ p \longleftrightarrow \mathcal{M}_{\text{@}}(S, i) \models_{\text{@}} p \rangle$

proof (*induct p arbitrary: i rule: wf-induct[where r= $\langle \text{measure size} \rangle$]*)

case 1

then show *?case ..*

next

case (2 x)

then show *?case*

using *assms(1)[of i x] assms(2)*

by (*cases x*) (*auto simp: reach-def*)

qed

lemma *reach-assign*: $\langle \text{reach } S \ ([i]_S) \subseteq \{[k]_S \mid k. \text{True}\} \rangle$
unfolding *reach-def assign-def by blast*

theorem *saturated-MCS*:

assumes $\langle \text{MCS } S \rangle$

shows $\langle \mathcal{M}_{\text{@}}(S, i) \llbracket \mathcal{R}_{\text{@}}(S) \rrbracket_{\text{@}}(p) \longleftrightarrow \mathcal{R}_{\text{@}}(S) (\mathcal{M}_{\text{@}}(S, i)) \ p \rangle$

proof (*cases p*)

case (*Nom k*)

have $\langle ([i]_S = [k]_S) \longleftrightarrow ([i]_S, \cdot k) \in S \rangle$

using *assms equiv-nom-equivp equiv-nom-assign by (metis assign-def equivp-def equiv-nom-def)*

then show *?thesis*

using *Nom by simp*

next

case (*Imp p q*)

have $\langle (([i]_S, p) \in S \longrightarrow ([i]_S, q) \in S) \longleftrightarrow ([i]_S, p \longrightarrow q) \in S \rangle$

using *assms MCS-derive MCS-explode MCS-impE' by (metis ImpI Weaken set-subset-Cons)*

then show *?thesis*

using *Imp by simp*

next

case (*Dia p*)

have $\langle ([i]_S, \diamond p) \in S \longleftrightarrow (\exists k. ([i]_S, @k \ p \wedge \diamond(\cdot k)) \in S) \rangle$

using *assms MCS-exi by fastforce*

moreover have $\langle ([i]_S, @k \ p \wedge \diamond(\cdot k)) \in S \longleftrightarrow ([i]_S, @k \ p) \in S \wedge ([i]_S, \diamond(\cdot k)) \in S \rangle$ **for** *k*

using *assms MCS-con by fast*

moreover have $\langle ([i]_S, @k \ p) \in S \longleftrightarrow (k, p) \in S \rangle$ **for** *k*

using *assms by (meson MCS-derive SatE SatI)*

moreover have $\langle (k, p) \in S \longleftrightarrow ([k]_S, p) \in S \rangle$ **for** *k*

using *assms by (meson MCS-Nom-refl equiv-nom-Nom equiv-nom-assign equiv-nom-def)*

moreover have $\langle ([i]_S, \diamond(\cdot k)) \in S \Longrightarrow [k]_S \in \text{reach } S \ ([i]_S) \rangle$ **for** *k*

unfolding *reach-def by blast*

ultimately have $\langle ([i]_S, \diamond p) \in S \longleftrightarrow (\exists k \in \text{reach } S ([i]_S). (k, p) \in S) \rangle$
unfolding *reach-def* **by** *blast*
then show *?thesis*
using *Dia reach-assign* **by** *fastforce*
next
case (*Sat k p*)
have $\langle ([k]_S, p) \in S \longleftrightarrow ([i]_S, @k p) \in S \rangle$
by (*metis SatE SatI assms MCS-derive equiv-nom-Nom equiv-nom-assign equiv-nom-symp sympD*)
then show *?thesis*
using *Sat* **by** *simp*
next
case (*All p*)
have $\langle ([i]_S, \mathbf{A} p) \in S \longleftrightarrow (\forall k. (k, p) \in S) \rangle$
using *assms MCS-uni* **by** *fastforce*
then have $\langle ([i]_S, \mathbf{A} p) \in S \longleftrightarrow (\forall k. ([k]_S, p) \in S) \rangle$
by (*meson MCS-Nom-sym assms equiv-nom-Nom equiv-nom-assign equiv-nom-def*)
then show *?thesis*
using *All* **by** *auto*
qed (*use assms in auto*)

interpretation *Truth-Witness semics semantics* $\langle \lambda S. \{\mathcal{M}_{@}(S, i) \mid i. \text{True}\} \rangle$ *rel consistent witness nominals-lbd*

proof

fix *p* **and** *M* **::** $\langle ('i, 'p, 'w) \text{ ctx} \rangle$
show $\langle (M \models_{@} p) = M \llbracket \text{semantics} \rrbracket_{@}(p) \rangle$
by (*cases M, induct p*) *auto*
next
fix *p M* **and** *S* **::** $\langle ('i, 'p) \text{ lbd set} \rangle$
assume $\langle \forall p. \forall M \in \{\mathcal{M}_{@}(S, i) \mid i. \text{True}\}. M \llbracket \mathcal{R}_{@}(S) \rrbracket_{@} p \longleftrightarrow \mathcal{R}_{@}(S) M p \rangle$ $\langle M \in \{\mathcal{M}_{@}(S, i) \mid i. \text{True}\} \rangle$
then show $\langle M \models_{@} p \longleftrightarrow \mathcal{R}_{@}(S) M p \rangle$
using *saturated-model* **by** *fast*
next
fix *S* **::** $\langle ('i, 'p) \text{ lbd set} \rangle$
assume $\langle \text{MCS } S \rangle$
then show $\langle \forall p. \forall M \in \{\mathcal{M}_{@}(S, i) \mid i. \text{True}\}. M \llbracket \mathcal{R}_{@}(S) \rrbracket_{@} p \longleftrightarrow \mathcal{R}_{@}(S) M p \rangle$
using *saturated-MCS* **by** *fastforce*
qed

lemma *Truth-lemma:*

assumes $\langle \text{MCS } S \rangle$
shows $\langle \mathcal{M}_{@}(S, i) \models_{@} p \longleftrightarrow (i, p) \in S \rangle$
proof –
have $\langle \mathcal{M}_{@}(S, i) \models_{@} p \longleftrightarrow ([i]_S, p) \in S \rangle$
using *assms truth-lemma* **by** *fastforce*
then show *?thesis*
using *assms* **by** (*meson MCS-Nom-sym equiv-nom-Nom equiv-nom-assign equiv-nom-def*)
qed

7.9 Cardinalities

datatype *marker* = *FlsM* | *ImpM* | *DiaM* | *SatM* | *AllM*

type-synonym $\langle 'i, 'p \rangle$ *enc* = $\langle ('i + 'p) + \text{marker} \times \text{nat} \rangle$

abbreviation $\langle \text{NOM } i \equiv \text{Inl } (\text{Inl } i) \rangle$
abbreviation $\langle \text{PRO } x \equiv \text{Inl } (\text{Inr } x) \rangle$
abbreviation $\langle \text{FLS} \equiv \text{Inr } (\text{FlsM}, 0) \rangle$
abbreviation $\langle \text{IMP } n \equiv \text{Inr } (\text{FlsM}, n) \rangle$
abbreviation $\langle \text{DIA} \equiv \text{Inr } (\text{DiaM}, 0) \rangle$
abbreviation $\langle \text{SAT} \equiv \text{Inr } (\text{SatM}, 0) \rangle$
abbreviation $\langle \text{GLO} \equiv \text{Inr } (\text{AllM}, 0) \rangle$

primrec $\text{encode} :: \langle ('i, 'p) \text{ fm} \Rightarrow ('i, 'p) \text{ enc list} \rangle$ **where**
 $\langle \text{encode } \perp = [\text{FLS}] \rangle$
 $| \langle \text{encode } (\cdot P) = [\text{PRO } P] \rangle$
 $| \langle \text{encode } (\cdot i) = [\text{NOM } i] \rangle$
 $| \langle \text{encode } (p \longrightarrow q) = \text{IMP } (\text{length } (\text{encode } p)) \# \text{encode } p @ \text{encode } q \rangle$
 $| \langle \text{encode } (\diamond p) = \text{DIA} \# \text{encode } p \rangle$
 $| \langle \text{encode } (@ i p) = \text{SAT} \# \text{NOM } i \# \text{encode } p \rangle$
 $| \langle \text{encode } (\mathbf{A} p) = \text{GLO} \# \text{encode } p \rangle$

lemma encode-ne [*simp*]: $\langle \text{encode } p \neq [] \rangle$
by (*induct p*) *auto*

lemma $\text{inj-encode}'$: $\langle \text{encode } p = \text{encode } q \Longrightarrow p = q \rangle$

proof (*induct p arbitrary: q*)

case *Fls*
then show *?case*
by (*cases q*) *auto*

next

case (*Pro P*)
then show *?case*
by (*cases q*) *auto*

next

case (*Nom i*)
then show *?case*
by (*cases q*) *auto*

next

case (*Imp p1 p2*)
then show *?case*
by (*cases q*) *auto*

next

case (*Dia p*)
then show *?case*
by (*cases q*) *auto*

next

case (*Sat i p*)
then show *?case*
by (*cases q*) *auto*

next

case (*All p*)
then show *?case*
by (*cases q*) *auto*

qed

primrec $\text{encode-lbd} :: \langle ('i, 'p) \text{ lbd} \Rightarrow ('i, 'p) \text{ enc list} \rangle$ **where**
 $\langle \text{encode-lbd } (i, p) = \text{NOM } i \# \text{encode } p \rangle$

lemma $\text{inj-encode-lbd}'$: $\langle \text{encode-lbd } (i, p) = \text{encode-lbd } (k, q) \Longrightarrow i = k \wedge p = q \rangle$

using *inj-encode'* **by** *auto*

lemma *inj-encode-lbd*: $\langle \text{inj encode-lbd} \rangle$
unfolding *inj-def* **using** *inj-encode-lbd'* **by** *auto*

lemma *finite-marker*: $\langle \text{finite (UNIV :: marker set)} \rangle$

proof –

have $\langle p \in \{FlsM, ImpM, DiaM, SatM, AllM\} \rangle$ **for** *p*
by (*cases p*) *auto*
then show *?thesis*
by (*meson ex-new-if-finite finite.emptyI finite-insert*)

qed

lemma *card-of-lbd*:

assumes $\langle \text{infinite (UNIV :: 'i set)} \rangle$
shows $\langle |UNIV :: ('i, 'p) \text{ lbd set}| \leq o |UNIV :: 'i \text{ set}| + c |UNIV :: 'p \text{ set}| \rangle$

proof –

have $\langle |UNIV :: \text{marker set}| \leq o |UNIV :: \text{nat set}| \rangle$
using *finite-marker* **by** (*simp add: ordLess-imp-ordLeq*)
moreover have $\langle \text{infinite (UNIV :: ('i + 'p) set)} \rangle$
using *assms* **by** *simp*
ultimately have $\langle |UNIV :: ('i, 'p) \text{ enc list set}| \leq o |UNIV :: ('i + 'p) \text{ set}| \rangle$
using *card-of-params-marker-lists* **by** *blast*
moreover have $\langle |UNIV :: ('i, 'p) \text{ lbd set}| \leq o |UNIV :: ('i, 'p) \text{ enc list set}| \rangle$
using *card-of-ordLeq inj-encode-lbd* **by** *blast*
ultimately have $\langle |UNIV :: ('i, 'p) \text{ lbd set}| \leq o |UNIV :: ('i + 'p) \text{ set}| \rangle$
using *ordLeq-transitive* **by** *blast*
then show *?thesis*
unfolding *csum-def* **by** *simp*

qed

7.10 Completeness

theorem *strong-completeness*:

fixes *p* :: $\langle ('i, 'p) \text{ fm} \rangle$
assumes $\langle \forall M :: ('i, 'p) \text{ model. } \forall g. \forall w \in W M. \text{range } g \subseteq W M \longrightarrow$
 $(\forall (k, q) \in X. (M, g, g k) \models_{@} q) \longrightarrow (M, g, w) \models_{@} p \rangle$
 $\langle \text{infinite (UNIV :: 'i set)} \rangle$
 $\langle |UNIV :: 'i \text{ set}| + c |UNIV :: 'p \text{ set}| \leq o |UNIV - \text{nominals } X| \rangle$
shows $\langle \exists A. \text{set } A \subseteq X \wedge A \vdash_{@} (i, p) \rangle$

proof (*rule ccontr*)

assume $\langle \nexists A. \text{set } A \subseteq X \wedge A \vdash_{@} (i, p) \rangle$
then have *: $\langle \forall A a. \text{set } A \subseteq \{(i, \neg p)\} \cup X \longrightarrow \neg A \vdash_{@} (a, \perp) \rangle$
using *Boole FlsE* **by** (*metis derive-split1 insert-is-Un subset-insert*)

let *?X* = $\langle \{(i, \neg p)\} \cup X \rangle$

let *?S* = $\langle \text{Extend } ?X \rangle$

have $\langle \text{consistent } ?X \rangle$

unfolding *consistent-def* **using** * **by** *blast*

moreover have $\langle \text{infinite (UNIV - nominals } X) \rangle$

using *assms(2-3)*

by (*metis Cinfiniteness-csum Cnotzero-UNIV Field-card-of cinfiniteness-def cinfiniteness-mono*)

then have $\langle |UNIV :: 'i \text{ set}| + c |UNIV :: 'p \text{ set}| \leq o |UNIV - \text{nominals } X - \text{nominals-lbd } (i, \neg p)| \rangle$

using *assms(3) finite-nominals-lbd card-of-infinite-diff-finite*

by (*metis ordIso-iff-ordLeq ordLeq-transitive*)
then have $\langle |UNIV :: 'i\ set| + c\ |UNIV :: 'p\ set| \leq o\ |UNIV - (nominals\ X \cup nominals\ lbd\ (i, \neg p))| \rangle$
 by (*metis Set-Diff-Un*)
then have $\langle |UNIV :: 'i\ set| + c\ |UNIV :: 'p\ set| \leq o\ |UNIV - nominals\ ?X| \rangle$
 by (*metis UN-insert insert-is-Un sup-commute*)
then have $\langle |UNIV :: ('i, 'p)\ lbd\ set| \leq o\ |UNIV - nominals\ ?X| \rangle$
 using *assms card-of-lbd ordLeq-transitive* **by blast**
ultimately have $\langle MCS\ ?S \rangle$
 using *MCS-Extend* **by fast**
then have $\langle \mathcal{M}_{@}(\ ?S, i) \models_{@} p \longleftrightarrow (i, p) \in ?S \rangle$ **for** $i\ p$
 using *Truth-lemma* **by fast**
then have $\langle (i, p) \in ?X \implies \mathcal{M}_{@}(\ ?S, i) \models_{@} p \rangle$ **for** $i\ p$
 using *Extend-subset* **by blast**
then have $\langle \mathcal{M}_{@}(\ ?S, i) \models_{@} \neg p \rangle \langle \forall (k, q) \in X. \mathcal{M}_{@}(\ ?S, k) \models_{@} q \rangle$
 by *blast+*
moreover from this have $\langle \mathcal{M}_{@}(\ ?S, i) \models_{@} p \rangle$
 using *assms(1)* **by force**
ultimately show *False*
 by *simp*
qed

abbreviation *valid* :: $\langle ('i, 'p)\ fm \implies bool \rangle$ **where**
 $\langle valid\ p \equiv \forall (M :: ('i, 'p)\ model)\ g. \forall w \in WM. range\ g \subseteq WM \longrightarrow (M, g, w) \models_{@} p \rangle$

theorem *completeness*:

fixes $p :: \langle ('i, 'p)\ fm \rangle$

assumes $\langle valid\ p \rangle \langle infinite\ (UNIV :: 'i\ set) \rangle \langle |UNIV :: 'p\ set| \leq o\ |UNIV :: 'i\ set| \rangle$

shows $\langle [] \vdash_{@} (i, p) \rangle$

proof –

have $\langle |UNIV :: 'i\ set| + c\ |UNIV :: 'p\ set| \leq o\ |UNIV :: 'i\ set| \rangle$

using *assms(2–3)* **by** (*simp add: cinfinite-def csum-absorb1 ordIso-imp-ordLeq*)

then show *?thesis*

using *assms strong-completeness*[**where** $X = \langle \{ \} \rangle$ **and** $p = p$] *infinite-UNIV-listI* **by auto**

qed

corollary *completeness'*:

fixes $p :: \langle ('i, 'i)\ fm \rangle$

assumes $\langle valid\ p \rangle \langle infinite\ (UNIV :: 'i\ set) \rangle$

shows $\langle [] \vdash_{@} (i, p) \rangle$

using *assms completeness*[*of p*] **by simp**

theorem *main*:

fixes $p :: \langle ('i, 'p)\ fm \rangle$

assumes $\langle i \notin nominals\ fm\ p \rangle \langle infinite\ (UNIV :: 'i\ set) \rangle \langle |UNIV :: 'p\ set| \leq o\ |UNIV :: 'i\ set| \rangle$

shows $\langle valid\ p \longleftrightarrow [] \vdash_{@} (i, p) \rangle$

using *assms completeness soundness'* **by metis**

corollary *main'*:

fixes $p :: \langle ('i, 'i)\ fm \rangle$

assumes $\langle i \notin nominals\ fm\ p \rangle \langle infinite\ (UNIV :: 'i\ set) \rangle$

shows $\langle valid\ p \longleftrightarrow [] \vdash_{@} (i, p) \rangle$

using *assms completeness' soundness'* **by metis**

end

Chapter 8

Example: First-Order Logic

theory *Example-First-Order-Logic* imports *Derivations* begin

8.1 Syntax

datatype (*params-tm*: 'f) *tm*
= *Var nat* (<#>)
| *Fun 'f* <'f *tm list*> (<·>)

abbreviation *Const* (<★>) **where** <★*a* ≡ ·*a* []>

datatype (*params-fm*: 'f, 'p) *fm*
= *Fls* (<⊥>)
| *Pre 'p* <'f *tm list*> (<·>)
| *Imp* <'f, 'p> *fm*> <'f, 'p> *fm*> (**infixr** <—> 55)
| *Exi* <'f, 'p> *fm*> (<∃>)

abbreviation *Neg* (<¬ -> [70] 70) **where** <¬ *p* ≡ *p* —> ⊥>

8.2 Semantics

type-synonym ('a, 'f, 'p) *model* = <(nat ⇒ 'a) × ('f ⇒ 'a list ⇒ 'a) × ('p ⇒ 'a list ⇒ bool)>

fun *semantics-tm* :: <(nat ⇒ 'a) × ('f ⇒ 'a list ⇒ 'a) ⇒ 'f *tm* ⇒ 'a> (<[-]>) **where**
| <[(*E*, -)] (#*n*) = *E n*>
| <[(*E*, *F*)] (·*f ts*) = *F f* (*map* [(*E*, *F*)] *ts*)>

primrec *add-env* :: <'a ⇒ (nat ⇒ 'a) ⇒ nat ⇒ 'a> (**infix** <§> 0) **where**
| <(t § s) 0 = *t*>
| <(t § s) (*Suc n*) = *s n*>

fun *semantics-fm* :: <'a, 'f, 'p> *model* ⇒ ('f, 'p) *fm* ⇒ bool (**infix** <|=₃> 50) **where**
| <- |=₃ ⊥ ↔ *False*>
| <(E, F, G) |=₃ ·*P ts* ↔ *G P* (*map* [(E, F)] *ts*)>
| <(E, F, G) |=₃ *p* —> *q* ↔ (E, F, G) |=₃ *p* —> (E, F, G) |=₃ *q*>
| <(E, F, G) |=₃ ∃*p* ↔ (∃*x*. (*x* § E, F, G) |=₃ *p*)>

8.3 Operations

primrec *lift-tm* :: <'f *tm* ⇒ 'f *tm*> **where**

$\langle \text{lift-tm } (\#n) = \#(n+1) \rangle$
 $\mid \langle \text{lift-tm } (\cdot f \text{ ts}) = \cdot f (\text{map lift-tm ts}) \rangle$

primrec *sub-tm* :: $\langle (\text{nat} \Rightarrow 'f \text{ tm}) \Rightarrow 'f \text{ tm} \Rightarrow 'f \text{ tm} \rangle$ **where**

$\langle \text{sub-tm } s (\#n) = s \ n \rangle$
 $\mid \langle \text{sub-tm } s (\cdot f \text{ ts}) = \cdot f (\text{map } (\text{sub-tm } s) \text{ ts}) \rangle$

primrec *sub-fm* :: $\langle (\text{nat} \Rightarrow 'f \text{ tm}) \Rightarrow ('f, 'p) \text{ fm} \Rightarrow ('f, 'p) \text{ fm} \rangle$ **where**

$\langle \text{sub-fm } - \perp = \perp \rangle$
 $\mid \langle \text{sub-fm } s (\cdot P \text{ ts}) = \cdot P (\text{map } (\text{sub-tm } s) \text{ ts}) \rangle$
 $\mid \langle \text{sub-fm } s (p \longrightarrow q) = \text{sub-fm } s \ p \longrightarrow \text{sub-fm } s \ q \rangle$
 $\mid \langle \text{sub-fm } s (\exists p) = \exists (\text{sub-fm } (\#0 \circ \lambda n. \text{lift-tm } (s \ n)) \ p) \rangle$

abbreviation *inst-single* :: $\langle 'f \text{ tm} \Rightarrow ('f, 'p) \text{ fm} \Rightarrow ('f, 'p) \text{ fm} \rangle$ ($\langle \langle - \rangle \rangle$) **where**

$\langle \langle t \rangle \equiv \text{sub-fm } (t \circ \#) \rangle$

abbreviation $\langle \text{params } S \equiv \bigcup p \in S. \text{params-fm } p \rangle$

abbreviation $\langle \text{params}' l \equiv \text{params } (\text{set } l) \rangle$

lemma *upd-params-tm [simp]*: $\langle f \notin \text{params-tm } t \implies \llbracket (E, F(f := x)) \rrbracket t = \llbracket (E, F) \rrbracket t \rangle$
by (*induct t*) (*auto cong: map-cong*)

lemma *upd-params-fm [simp]*: $\langle f \notin \text{params-fm } p \implies (E, F(f := x), G) \models_{\exists} p \longleftrightarrow (E, F, G) \models_{\exists} p \rangle$
by (*induct p arbitrary: E*) (*auto cong: map-cong*)

lemma *finite-params-tm [simp]*: $\langle \text{finite } (\text{params-tm } t) \rangle$
by (*induct t*) *simp-all*

lemma *finite-params-fm [simp]*: $\langle \text{finite } (\text{params-fm } p) \rangle$
by (*induct p*) *simp-all*

lemma *env [simp]*: $\langle P ((x \circ E) \ n) = (P \ x \circ \lambda n. P (E \ n)) \ n \rangle$
by (*induct n*) *simp-all*

lemma *lift-lemma*: $\langle \llbracket (x \circ E, F) \rrbracket (\text{lift-tm } t) = \llbracket (E, F) \rrbracket t \rangle$
by (*induct t*) (*auto cong: map-cong*)

lemma *sub-tm-semantics*: $\langle \llbracket (E, F) \rrbracket (\text{sub-tm } s \ t) = \llbracket (\lambda n. \llbracket (E, F) \rrbracket (s \ n), F) \rrbracket t \rangle$
by (*induct t*) (*auto cong: map-cong*)

lemma *sub-fm-semantics [simp]*: $\langle (E, F, G) \models_{\exists} \text{sub-fm } s \ p \longleftrightarrow (\lambda n. \llbracket (E, F) \rrbracket (s \ n), F, G) \models_{\exists} p \rangle$
by (*induct p arbitrary: E s*) (*auto cong: map-cong simp: sub-tm-semantics lift-lemma*)

lemma *sub-tm-Var [simp]*: $\langle \text{sub-tm } \# \ t = t \rangle$
by (*induct t*) (*auto cong: map-cong*)

lemma *reduce-Var [simp]*: $\langle (\# \ 0 \circ \lambda n. \# (Suc \ n)) = \# \rangle$
proof (*rule ext*)

fix *n*

show $\langle (\# \ 0 \circ \lambda n. \# (Suc \ n)) \ n = \# \ n \rangle$

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

qed

lemma *sub-fm-Var [simp]*:
fixes *p* :: $\langle ('f, 'p) \text{ fm} \rangle$

shows $\langle \text{sub-fm } \# \ p = p \rangle$
proof $(\text{induct } p)$
case $(\text{Pre } P \ ts)$
then show $?case$
by $(\text{auto cong: map-cong})$
qed simp-all

lemma semantics-tm-id $[\text{simp}]$: $\langle \llbracket (\#, \cdot) \rrbracket t = t \rangle$
by $(\text{induct } t) (\text{auto cong: map-cong})$

lemma $\text{semantics-tm-id-map}$ $[\text{simp}]$: $\langle \text{map } \llbracket (\#, \cdot) \rrbracket \ ts = ts \rangle$
by $(\text{auto cong: map-cong})$

The built-in *size* is not invariant under substitution.

primrec $\text{size-fm} :: \langle ('f, 'p) \text{ fm} \Rightarrow \text{nat} \rangle$ **where**
 $\langle \text{size-fm } \perp = 1 \rangle$
 $|\ \langle \text{size-fm } (\cdot -) = 1 \rangle$
 $|\ \langle \text{size-fm } (p \longrightarrow q) = 1 + \text{size-fm } p + \text{size-fm } q \rangle$
 $|\ \langle \text{size-fm } (\exists p) = 1 + \text{size-fm } p \rangle$

lemma size-sub-fm $[\text{simp}]$: $\langle \text{size-fm } (\text{sub-fm } s \ p) = \text{size-fm } p \rangle$
by $(\text{induct } p \text{ arbitrary: } s) \text{ simp-all}$

8.4 Calculus

inductive $\text{Calculus} :: \langle ('f, 'p) \text{ fm list} \Rightarrow ('f, 'p) \text{ fm} \Rightarrow \text{bool} \rangle$ (**infix** $\langle \vdash_{\exists} \rangle$ 50) **where**
 Assm $[\text{simp}]$: $\langle p \in \text{set } A \Longrightarrow A \vdash_{\exists} p \rangle$
 $|\ \text{FlsE}$ $[\text{elim}]$: $\langle A \vdash_{\exists} \perp \Longrightarrow A \vdash_{\exists} p \rangle$
 $|\ \text{ImpI}$ $[\text{intro}]$: $\langle p \# A \vdash_{\exists} q \Longrightarrow A \vdash_{\exists} p \longrightarrow q \rangle$
 $|\ \text{ImpE}$ $[\text{dest}]$: $\langle A \vdash_{\exists} p \longrightarrow q \Longrightarrow A \vdash_{\exists} p \Longrightarrow A \vdash_{\exists} q \rangle$
 $|\ \text{ExiI}$ $[\text{intro}]$: $\langle A \vdash_{\exists} \langle t \rangle p \Longrightarrow A \vdash_{\exists} \exists p \rangle$
 $|\ \text{ExiE}$ $[\text{elim}]$: $\langle A \vdash_{\exists} \exists p \Longrightarrow a \notin \text{params } (\text{set } (p \# q \# A)) \Longrightarrow \langle \star a \rangle p \# A \vdash_{\exists} q \Longrightarrow A \vdash_{\exists} q \rangle$
 $|\ \text{Clas}$: $\langle (p \longrightarrow q) \# A \vdash_{\exists} p \Longrightarrow A \vdash_{\exists} p \rangle$

8.4.1 Weakening

abbreviation $\langle \text{psub } f \equiv \text{map-fm } f \ \text{id} \rangle$

lemma map-tm-sub-tm $[\text{simp}]$: $\langle \text{map-tm } f \ (\text{sub-tm } g \ t) = \text{sub-tm } (\text{map-tm } f \ o \ g) \ (\text{map-tm } f \ t) \rangle$
by $(\text{induct } t) \text{ simp-all}$

lemma map-tm-lift-tm $[\text{simp}]$: $\langle \text{map-tm } f \ (\text{lift-tm } t) = \text{lift-tm } (\text{map-tm } f \ t) \rangle$
by $(\text{induct } t) \text{ simp-all}$

lemma psub-sub-fm : $\langle \text{psub } f \ (\text{sub-fm } g \ p) = \text{sub-fm } (\text{map-tm } f \ o \ g) \ (\text{psub } f \ p) \rangle$
by $(\text{induct } p \text{ arbitrary: } g) (\text{simp-all add: comp-def})$

lemma $\text{map-tm-inst-single}$: $\langle (\text{map-tm } f \ o \ (u \ \S \ \#)) \ t = (\text{map-tm } f \ u \ \S \ \#) \ t \rangle$
by $(\text{induct } t) \text{ auto}$

lemma psub-inst-single $[\text{simp}]$: $\langle \text{psub } f \ (\langle t \rangle p) = \langle \text{map-tm } f \ t \rangle (\text{psub } f \ p) \rangle$
unfolding $\text{psub-sub-fm map-tm-inst-single ..}$

lemma map-tm-upd $[\text{simp}]$: $\langle a \notin \text{params-tm } t \Longrightarrow \text{map-tm } (f(a := b)) \ t = \text{map-tm } f \ t \rangle$
by $(\text{induct } t) \text{ auto}$

lemma *psub-upd [simp]*: $\langle a \notin \text{params-fm } p \implies \text{psub } (f(a := b)) \text{ } p = \text{psub } f \text{ } p \rangle$
 by (*induct p*) *auto*

class *inf-univ* =
 fixes *itself* :: $\langle 'a \text{ itself} \rangle$
 assumes *infinite-UNIV*: $\langle \text{infinite } (UNIV :: 'a \text{ set}) \rangle$

lemma *Calculus-psub*:
 fixes *f* :: $\langle 'f \Rightarrow 'g :: \text{inf-univ} \rangle$
 shows $\langle A \vdash_{\exists} p \implies \text{map } (\text{psub } f) \text{ } A \vdash_{\exists} \text{psub } f \text{ } p \rangle$

proof (*induct A p arbitrary: f pred: Calculus*)

case (*Assm p A*)

then show *?case*

by *simp*

next

case (*FlsE A p*)

then show *?case*

by *force*

next

case (*ImpI p A q*)

then show *?case*

by *auto*

next

case (*ImpE A p q*)

then show *?case*

by *auto*

next

case (*ExiI A t p*)

then show *?case*

by (*metis Calculus.ExiI fm.simps(27) psub-inst-single*)

next

case (*ExiE A p a q*)

let *?params* = $\langle \text{params}' (p \# q \# A) \rangle$

have $\langle \text{finite } ?\text{params} \rangle$

by *simp*

then obtain *b* **where** *b*: $\langle b \notin \{f \text{ } a\} \cup f \text{ } '?\text{params} \rangle$

using *ex-new-if-finite infinite-UNIV*

by (*metis finite.emptyI finite.insertI finite-UnI finite-imageI*)

define *g* **where** $\langle g \equiv f(a := b) \rangle$

have $\langle a \notin \text{params}' (p \# q \# A) \rangle$

using *ExiE* by *simp*

then have *b'*: $\langle b \notin \text{params}' (\text{map } (\text{psub } g) (p \# q \# A)) \rangle$

unfolding *g-def* **using** *b ExiE(3)* **by** (*auto simp: fm.set-map(1)*)

have $\langle \text{map } (\text{psub } g) \text{ } A \vdash_{\exists} \text{psub } g \text{ } (\exists p) \rangle$

using *ExiE* by *blast*

then have $\langle \text{map } (\text{psub } g) \text{ } A \vdash_{\exists} \exists (\text{psub } g \text{ } p) \rangle$

by *simp*

moreover have $\langle \text{map } (\text{psub } g) ((\star a)p \# A) \vdash_{\exists} \text{psub } g \text{ } q \rangle$

using *ExiE* by *blast*

then have $\langle \langle \star b \rangle (\text{psub } g \text{ } p) \# \text{map } (\text{psub } g) \text{ } A \vdash_{\exists} \text{psub } g \text{ } q \rangle$

unfolding *g-def* **by** *simp*

ultimately have $\langle \text{map } (\text{psub } g) \text{ } A \vdash_{\exists} \text{psub } g \text{ } q \rangle$

```

    using b' by fastforce
  moreover have ⟨psub g q = psub f q⟩ ⟨map (psub g) A = map (psub f) A⟩
    unfolding g-def using ExiE.hyps(3) by simp-all
  ultimately show ?case
    by metis
next
case (Clas p q A)
then show ?case
  using Calculus.Clas by auto
qed

```

lemma *Weaken*:

```

  fixes p :: ⟨('f :: inf-univ, 'p) fm⟩
  shows ⟨A ⊢∃ p ⟹ set A ⊆ set B ⟹ B ⊢∃ p⟩
proof (induct A p arbitrary: B pred: Calculus)
  case (Assm p A)
  then show ?case
    by auto
next
  case (FlsE A p)
  then show ?case
    using Calculus.FlsE by blast
next
  case (ImpI p A q)
  then show ?case
    by (simp add: Calculus.ImpI subset-code(1))
next
  case (ImpE A p q)
  then show ?case
    by blast
next
  case (ExiI A t p)
  then show ?case
    by blast
next
  case (ExiE A p a q)
  let ?params = ⟨params' (p # q # B)⟩
  have ⟨finite ?params⟩
    by simp
  then obtain b where b: ⟨b ∉ ?params⟩
    using ex-new-if-finite infinite-UNIV by blast
  then have b': ⟨b ∉ params' A⟩
    using ExiE by auto

  define f where ⟨f ≡ id(a := b, b := a)⟩
  let ?B = ⟨map (psub f) B⟩

  have f: ⟨∀ p ∈ set A. psub f p = p⟩
    using ExiE(3) b' by (simp add: fm.map-id f-def)
  then have ⟨set A ⊆ set ?B⟩
    using ExiE.prem by force
  then have ⟨?B ⊢∃ ∃ p⟩
    using ExiE.hyps by blast

  moreover have ⟨⟨★a⟩p ∈ set (⟨★a⟩p # ?B)⟩
    using ExiE(3) b by (auto simp: fm.map-id0)

```

then have $\langle \text{set } (\star a) p \# A \subseteq \text{set } (\star a)p \# ?B \rangle$
using $\langle \text{set } A \subseteq \text{set } ?B \rangle$ **by** *auto*
then have $\langle (\star a)p \# ?B \vdash_{\exists} q \rangle$
using *ExiE(5)* **by** *blast*

moreover have $\langle a \notin \text{params}' (p \# q \# ?B) \rangle$
using *ExiE(3)* *b* **by** (*simp add: fm.set-map(1) image-iff f-def*)

ultimately have $\langle ?B \vdash_{\exists} q \rangle$
by *fast*
then have $\langle \text{map } (\text{psub } f) ?B \vdash_{\exists} \text{psub } f q \rangle$
using *Calculus-psub* **by** *blast*
moreover have $\langle \text{psub } f q = q \rangle$
using *ExiE.hyps(3)* *b* *fm.map-id* **unfolding** *f-def* **by** *auto*
moreover have $\langle f \circ f = \text{id} \rangle$
unfolding *f-def* **by** *auto*
then have $\langle \text{psub } f \circ \text{psub } f = \text{id} \rangle$
by (*auto simp: fm.map-comp fm.map-id0*)
then have $\langle \text{map } (\text{psub } f) ?B = B \rangle$
unfolding *map-map* **by** (*metis list.map-id*)
ultimately show *?case*
by *simp*
next
case (*Clas p q A*)
then show *?case*
using *Calculus.Clas*
by (*metis insert-mono list.simps(15)*)
qed

8.5 Soundness

theorem *soundness*: $\langle A \vdash_{\exists} p \implies \forall q \in \text{set } A. (E, F, G) \models_{\exists} q \implies (E, F, G) \models_{\exists} p \rangle$
proof (*induct p arbitrary: F pred: Calculus*)
case (*ExiE A p a q*)
then obtain *x* **where** $\langle x \circ E, F, G \rangle \models_{\exists} p$
by *fastforce*
then have $\langle (E, F(a := \lambda-. x), G) \models_{\exists} (\star a)p \rangle$
using *ExiE(3)* **by** *simp*
moreover have $\langle \forall q \in \text{set } A. (E, F(a := \lambda-. x), G) \models_{\exists} q \rangle$
using *ExiE(3, 6)* **by** *simp*
ultimately have $\langle (E, F(a := \lambda-. x), G) \models_{\exists} q \rangle$
using *ExiE(5)* **by** *simp*
then show *?case*
using *ExiE(3)* **by** *simp*
qed *auto*

corollary *soundness'*: $\langle [] \vdash_{\exists} p \implies M \models_{\exists} p \rangle$
using *soundness* **by** (*cases M*) *fastforce*

corollary $\langle \neg ([] \vdash_{\exists} \perp) \rangle$
using *soundness'* **by** *fastforce*

8.6 Admissible Rules

lemma *Assm-head*: $\langle p \# A \vdash_{\exists} p \rangle$

by auto

lemma *Boole*: $\langle (\neg p) \# A \vdash_{\exists} \perp \implies A \vdash_{\exists} p \rangle$
using *Clas FlsE* by *blast*

corollary *Weak*:

fixes $p :: \langle ('f :: \text{inf-univ}, 'p) \text{fm} \rangle$
shows $\langle A \vdash_{\exists} p \implies q \# A \vdash_{\exists} p \rangle$
using *Weaken*[**where** $B = \langle q \# A \rangle$] by *auto*

lemma *deduct1*:

fixes $p :: \langle ('f :: \text{inf-univ}, 'p) \text{fm} \rangle$
shows $\langle A \vdash_{\exists} p \longrightarrow q \implies p \# A \vdash_{\exists} q \rangle$
using *Assm-head Weak* by *blast*

lemma *Weak'*:

fixes $p :: \langle ('f :: \text{inf-univ}, 'p) \text{fm} \rangle$
shows $\langle A \vdash_{\exists} p \implies B @ A \vdash_{\exists} p \rangle$
by (*induct B*) (*simp-all add: Weak*)

interpretation *Derivations* $\langle \text{Calculus} :: ('f :: \text{inf-univ}, 'p) \text{fm list} \Rightarrow - \rangle$

proof

show $\langle \bigwedge A p. p \in \text{set } A \implies A \vdash_{\exists} p \rangle$
by *simp*

next

show $\langle \bigwedge A B. A \vdash_{\exists} r \implies \text{set } A = \text{set } B \implies B \vdash_{\exists} r \rangle$ **for** $r :: \langle ('f, 'p) \text{fm} \rangle$
using *Weaken* by *blast*

qed

8.7 Maximal Consistent Sets

definition *consistent* :: $\langle ('f, 'p) \text{fm set} \Rightarrow \text{bool} \rangle$ **where**

$\langle \text{consistent } S \equiv \forall A. \text{set } A \subseteq S \longrightarrow \neg A \vdash_{\exists} \perp \rangle$

fun *witness* :: $\langle ('f, 'p) \text{fm} \Rightarrow ('f, 'p) \text{fm set} \Rightarrow ('f, 'p) \text{fm set} \rangle$ **where**

$\langle \text{witness } (\exists p) S = (\text{let } a = \text{SOME } a. a \notin \text{params } (\{p\} \cup S) \text{ in } \{\langle \star a \rangle p\}) \rangle$

| $\langle \text{witness } - - = \{\} \rangle$

lemma *consistent-add-witness*:

fixes $p :: \langle ('f :: \text{inf-univ}, 'p) \text{fm} \rangle$
assumes $\langle \text{consistent } S \rangle \langle \exists p \in S \rangle \langle a \notin \text{params } S \rangle$
shows $\langle \text{consistent } (\{\langle \star a \rangle p\} \cup S) \rangle$
unfolding *consistent-def*

proof *safe*

fix A

assume $\langle \text{set } A \subseteq \{\langle \star a \rangle p\} \cup S \rangle \langle A \vdash_{\exists} \perp \rangle$

then obtain A' **where** $\langle \text{set } A' \subseteq S \rangle \langle \langle \star a \rangle p \# A' \vdash_{\exists} \perp \rangle$

using *assms derive-split1* by (*metis consistent-def insert-is-Un subset-insert*)

then have $\langle \langle \star a \rangle p \# A' \vdash_{\exists} \perp \rangle$

using *Boole* by *blast*

then have $\langle \langle \star a \rangle p \# \exists p \# A' \vdash_{\exists} \perp \rangle$

using *Weak deduct1* by *blast*

moreover have $\langle a \notin \text{params-fm } p \rangle \langle \forall p \in \text{set } (\exists p \# A'). a \notin \text{params-fm } p \rangle$

using $\langle \text{set } A' \subseteq S \rangle$ *assms(2-3)* by *auto*

then have $\langle a \notin \text{params } (\{p\} \cup \{\perp\} \cup \text{set } (\exists p \# A')) \rangle$

```

  using calculation by simp
moreover have ⟨ $\exists p \# A' \vdash_{\exists} \exists p$ ⟩
  by simp
ultimately have ⟨ $\exists p \# A' \vdash_{\exists} \perp$ ⟩
  by fastforce
moreover have ⟨ $\text{set } (\exists p \# A') \subseteq S$ ⟩
  using ⟨ $\text{set } A' \subseteq S$ ⟩ assms(2) by simp
ultimately show False
  using assms(1) unfolding consistent-def by blast
qed

```

lemma *consistent-witness'*:

```

fixes p :: ⟨('f :: inf-univ, 'p) fm⟩
assumes ⟨consistent ({p} ∪ S)⟩ ⟨infinite (UNIV - params S)⟩
shows ⟨consistent (witness p S ∪ {p} ∪ S)⟩
using assms
proof (induct p S rule: witness.induct)
  case (1 p S)
  have ⟨infinite (UNIV - params ({p} ∪ S))⟩
    using 1(2) finite-params-fm by (simp add: infinite-Diff-fin-Un)
  then have ⟨ $\exists a. a \notin \text{params } (\{p\} \cup S)$ ⟩
    by (simp add: not-finite-existsD set-diff-eq)
  then have ⟨(SOME a. a ∉ params ({p} ∪ S)) ∉ params ({p} ∪ S)⟩
    by (rule someI-ex)
  then obtain a where a: ⟨witness (∃ p) S = {⟨ $\star a$ ⟩p}⟩ ⟨a ∉ params ({∃ p} ∪ S)⟩
    by simp
  then show ?case
    using 1(1-2) a(1) consistent-add-witness[where S=⟨{∃ p} ∪ S⟩] by auto
qed (auto simp: assms)

```

interpretation *MCS-Witness-UNIV consistent witness* ⟨*params-fm* :: ('f :: inf-univ, 'p) fm ⇒ -⟩

proof

```

fix p and S :: ⟨('f, 'p) fm set⟩
show ⟨finite (params (witness p S))⟩
  by (induct p S rule: witness.induct) simp-all
next
fix p and S :: ⟨('f, 'p) fm set⟩
assume ⟨consistent ({p} ∪ S)⟩ ⟨infinite (UNIV - params S)⟩
then show ⟨consistent ({p} ∪ S ∪ witness p S)⟩
  using consistent-witness' by (simp add: sup-commute)
next
show ⟨infinite (UNIV :: ('f, 'p) fm set)⟩
  using infinite-UNIV-size[of ⟨ $\lambda p. p \longrightarrow p$ ⟩] by simp
qed (auto simp: consistent-def)

```

interpretation *Derivations-Cut-MCS consistent* ⟨*Calculus* :: ('f :: inf-univ, 'p) fm list ⇒ -⟩

proof

```

show ⟨ $\bigwedge S. \text{consistent } S = (\forall A. \text{set } A \subseteq S \longrightarrow (\exists q. \neg A \vdash_{\exists} q))$ ⟩
  unfolding consistent-def using FlsE by blast
next
show ⟨ $\bigwedge A B p. A \vdash_{\exists} p \implies p \# B \vdash_{\exists} q \implies A @ B \vdash_{\exists} q$ ⟩ for q :: ⟨('f, 'p) fm⟩
  by (metis ImpE ImpI Un-upper1 Weak' Weaken set-append)
qed

```

interpretation *Derivations-Bot consistent Calculus* ⟨ \perp :: ('f :: inf-univ, 'p) fm⟩

proof

qed fast

interpretation *Derivations-Imp consistent Calculus* $\langle \lambda p q. p \longrightarrow q :: (f :: \text{inf-univ}, 'p) \text{ fm} \rangle$
proof qed fast+

interpretation *Derivations-Exi consistent witness params-fm Calculus* $\langle \exists \rangle \langle \lambda t p. \langle t \rangle p :: (f :: \text{inf-univ}, 'p) \text{ fm} \rangle$
proof qed auto

8.8 Truth Lemma

abbreviation *canonical* :: $\langle (f, 'p) \text{ fm set} \Rightarrow (f \text{ tm}, 'f, 'p) \text{ model} \rangle \langle \mathcal{M}_\exists \rangle$ **where**
 $\langle \mathcal{M}_\exists(S) \equiv (\#, \cdot, \lambda P \text{ ts. } \cdot P \text{ ts} \in S) \rangle$

fun *semics* ::
 $\langle (a, 'f, 'p) \text{ model} \Rightarrow ((a, 'f, 'p) \text{ model} \Rightarrow (f, 'p) \text{ fm} \Rightarrow \text{bool}) \Rightarrow (f, 'p) \text{ fm} \Rightarrow \text{bool} \rangle$
 $\langle (- \llbracket - \rrbracket_\exists -) \rangle$ [55, 0, 55] 55) **where**
 $\langle - \llbracket - \rrbracket_\exists \perp \longleftrightarrow \text{False} \rangle$
 $| \langle (E, F, G) \llbracket - \rrbracket_\exists \cdot P \text{ ts} \longleftrightarrow G P (\text{map } \llbracket (E, F) \rrbracket \text{ ts}) \rangle$
 $| \langle (E, F, G) \llbracket \mathcal{R} \rrbracket_\exists p \longrightarrow q \longleftrightarrow \mathcal{R} (E, F, G) p \longrightarrow \mathcal{R} (E, F, G) q \rangle$
 $| \langle (E, F, G) \llbracket \mathcal{R} \rrbracket_\exists \exists p \longleftrightarrow (\exists x. \mathcal{R} (x \circ E, F, G) p) \rangle$

fun *rel* :: $\langle (f, 'p) \text{ fm set} \Rightarrow (f \text{ tm}, 'f, 'p) \text{ model} \Rightarrow (f, 'p) \text{ fm} \Rightarrow \text{bool} \rangle \langle \mathcal{R}_\exists \rangle$ **where**
 $\langle \mathcal{R}_\exists(S) (E, -, -) p \longleftrightarrow \text{sub-fm } E p \in S \rangle$

theorem *saturated-model*:

assumes $\langle \bigwedge p. \forall M \in \{\mathcal{M}_\exists(S)\}. M \llbracket \mathcal{R}_\exists(S) \rrbracket_\exists p \longleftrightarrow \mathcal{R}_\exists(S) M p \rangle \langle M \in \{\mathcal{M}_\exists(S)\} \rangle$
shows $\langle \mathcal{R}_\exists(S) M p \longleftrightarrow M \models_\exists p \rangle$
proof (*induct p rule: wf-induct[where r= $\langle \text{measure size-fm} \rangle$]*)
case 1
then show ?case ..
next
case (2 x)
then show ?case
using *assms(1)[of x] assms(2) by (cases x) simp-all*

qed

theorem *saturated-MCS*:

fixes $p :: \langle (f :: \text{inf-univ}, 'p) \text{ fm} \rangle$
assumes $\langle \text{MCS } S \rangle$
shows $\langle \mathcal{R}_\exists(S) (\mathcal{M}_\exists(S)) p \longleftrightarrow \mathcal{M}_\exists(S) \llbracket \mathcal{R}_\exists(S) \rrbracket_\exists p \rangle$
using *assms by (cases p) (auto cong: map-cong)*

interpretation *Truth-Witness semics semantics-fm* $\langle \lambda S. \{\mathcal{M}_\exists(S)\} \text{ rel consistent witness} \rangle$
 $\langle \text{params-fm} :: (f :: \text{inf-univ}, 'p) \text{ fm} \Rightarrow - \rangle$

proof

fix p **and** $M :: \langle (f \text{ tm}, 'f, 'p) \text{ model} \rangle$
show $\langle M \models_\exists p \longleftrightarrow M \llbracket (\models_\exists) \rrbracket_\exists p \rangle$
by (*cases M, induct p*) auto

qed (*use saturated-model saturated-MCS in blast*)+

8.9 Cardinalities

datatype *marker* = *VarM* | *FunM* | *TmM* | *FlsM* | *PreM* | *ImpM* | *ExiM*

type-synonym $\langle 'f, 'p \rangle \text{ enc} = \langle ('f + 'p) + \text{marker} \times \text{nat} \rangle$

abbreviation $\langle \text{FUNS } f \equiv \text{Inl } (\text{Inl } f) \rangle$

abbreviation $\langle \text{PRES } p \equiv \text{Inl } (\text{Inr } p) \rangle$

abbreviation $\langle \text{VAR } n \equiv \text{Inr } (\text{VarM}, n) \rangle$

abbreviation $\langle \text{FUN } n \equiv \text{Inr } (\text{FunM}, n) \rangle$

abbreviation $\langle \text{TM } n \equiv \text{Inr } (\text{TmM}, n) \rangle$

abbreviation $\langle \text{PRE } n \equiv \text{Inr } (\text{PreM}, n) \rangle$

abbreviation $\langle \text{FLS} \equiv \text{Inr } (\text{FlsM}, 0) \rangle$

abbreviation $\langle \text{IMP } n \equiv \text{Inr } (\text{FlsM}, n) \rangle$

abbreviation $\langle \text{EXI} \equiv \text{Inr } (\text{ExiM}, 0) \rangle$

primrec

$\text{encode-tm} :: \langle 'f \text{ tm} \Rightarrow ('f, 'p) \text{ enc list} \rangle$ **and**

$\text{encode-tms} :: \langle 'f \text{ tm list} \Rightarrow ('f, 'p) \text{ enc list} \rangle$ **where**

$\langle \text{encode-tm } (\#n) = [\text{VAR } n] \rangle$

| $\langle \text{encode-tm } (\cdot f \text{ ts}) = \text{FUN } (\text{length } \text{ts}) \# \text{FUNS } f \# \text{encode-tms } \text{ts} \rangle$

| $\langle \text{encode-tms } [] = [] \rangle$

| $\langle \text{encode-tms } (t \# \text{ts}) = \text{TM } (\text{length } (\text{encode-tm } t)) \# \text{encode-tm } t @ \text{encode-tms } \text{ts} \rangle$

lemma encode-tm-ne [simp]: $\langle \text{encode-tm } t \neq [] \rangle$

by $(\text{induct } t) \text{ auto}$

lemma $\text{inj-encode-tm}'$:

$\langle (\text{encode-tm } t :: ('f, 'p) \text{ enc list}) = \text{encode-tm } s \implies t = s \rangle$

$\langle (\text{encode-tms } \text{ts} :: ('f, 'p) \text{ enc list}) = \text{encode-tms } \text{ss} \implies \text{ts} = \text{ss} \rangle$

proof $(\text{induct } t \text{ and } \text{ts arbitrary: } s \text{ and } \text{ss rule: encode-tm.induct encode-tms.induct})$

case $(\text{Var } n)$

then show $?case$

by $(\text{cases } s) \text{ auto}$

next

case $(\text{Fun } f \text{ fts})$

then show $?case$

by $(\text{cases } s) \text{ auto}$

next

case Nil-tm

then show $?case$

by $(\text{cases } \text{ss}) \text{ auto}$

next

case $(\text{Cons-tm } t \text{ ts})$

then show $?case$

by $(\text{cases } \text{ss}) \text{ auto}$

qed

lemma inj-encode-tm : $\langle \text{inj } \text{encode-tm} \rangle$

unfolding inj-def **using** $\text{inj-encode-tm}'$ **by** blast

primrec $\text{encode-fm} :: \langle ('f, 'p) \text{ fm} \Rightarrow ('f, 'p) \text{ enc list} \rangle$ **where**

$\langle \text{encode-fm } \perp = [\text{FLS}] \rangle$

| $\langle \text{encode-fm } (\cdot P \text{ ts}) = \text{PRE } (\text{length } \text{ts}) \# \text{PRES } P \# \text{encode-tms } \text{ts} \rangle$

| $\langle \text{encode-fm } (p \longrightarrow q) = \text{IMP } (\text{length } (\text{encode-fm } p)) \# \text{encode-fm } p @ \text{encode-fm } q \rangle$

| $\langle \text{encode-fm } (\exists p) = \text{EXI} \# \text{encode-fm } p \rangle$

lemma encode-fm-ne [simp]: $\langle \text{encode-fm } p \neq [] \rangle$

```

by (induct p) auto

lemma inj-encode-fm': ⟨encode-fm p = encode-fm q ⟹ p = q⟩
proof (induct p arbitrary: q)
  case Fls
  then show ?case
    by (cases q) auto
next
  case (Pre P ts)
  then show ?case
    by (cases q) (auto simp: inj-encode-tm')
next
  case (Imp p1 p2)
  then show ?case
    by (cases q) auto
next
  case (Exi p)
  then show ?case
    by (cases q) auto
qed

lemma inj-encode-fm: ⟨inj encode-fm⟩
  unfolding inj-def using inj-encode-fm' by blast

lemma finite-marker: ⟨finite (UNIV :: marker set)⟩
proof -
  have ⟨p ∈ {VarM, FunM, TmM, FlsM, PreM, ImpM, ExiM}⟩ for p
    by (cases p) auto
  then show ?thesis
    by (meson ex-new-if-finite finite.emptyI finite.insert)
qed

lemma card-of-fm:
  ⟨|UNIV :: ('f :: inf-univ, 'p) fm set| ≤o |UNIV :: 'f set| +c |UNIV :: 'p set|⟩
proof -
  have ⟨|UNIV :: marker set| ≤o |UNIV :: nat set|⟩
    using finite-marker by (simp add: ordLess-imp-ordLeq)
  moreover have ⟨infinite (UNIV :: ('f + 'p) set)⟩
    by (simp add: inf-univ-class.infinite-UNIV)
  ultimately have ⟨|UNIV :: ('f, 'p) enc list set| ≤o |UNIV :: ('f + 'p) set|⟩
    using card-of-params-marker-lists by blast
  moreover have ⟨|UNIV :: ('f, 'p) fm set| ≤o |UNIV :: ('f, 'p) enc list set|⟩
    using card-of-ordLeq inj-encode-fm by blast
  ultimately have ⟨|UNIV :: ('f, 'p) fm set| ≤o |UNIV :: ('f + 'p) set|⟩
    using ordLeq-transitive by blast
  then show ?thesis
    unfolding csum-def by simp
qed

```

8.10 Completeness

theorem strong-completeness:

```

assumes ⟨∀ M :: ('f tm, 'f :: inf-univ, 'p) model. (∀ q ∈ X. M ⊨∃ q) ⟶ M ⊨∃ p⟩
  ⟨|UNIV :: 'f set| +c |UNIV :: 'p set| ≤o |UNIV - params X|⟩
shows ⟨∃ A. set A ⊆ X ∧ A ⊢∃ p⟩

```

proof (*rule ccontr*)

assume $\langle \nexists A. \text{set } A \subseteq X \wedge A \vdash_{\exists} p \rangle$
then have *: $\langle \forall A. \text{set } A \subseteq \{\neg p\} \cup X \longrightarrow \neg A \vdash_{\exists} \perp \rangle$
using *Boole FlsE* **by** (*metis derive-split1 insert-is-Un subset-insert*)

let $?X = \langle \{\neg p\} \cup X \rangle$
let $?S = \langle \text{Extend } ?X \rangle$

have $\langle \text{consistent } ?X \rangle$

unfolding *consistent-def* **using** * **by** *blast*
moreover have $\langle \text{infinite } (UNIV - \text{params } X) \rangle$
using *assms(2) inf-univ-class.infinite-UNIV*
by (*metis Cinfiniteness-csum Cnotzero-UNIV Field-card-of cinfiniteness-def cinfiniteness-mono*)
then have $\langle |UNIV :: 'f \text{ set}| + c \mid |UNIV :: 'p \text{ set}| \leq o \mid UNIV - \text{params } X - \text{params-fm } (\neg p) \rangle$
using *assms(2) finite-params-fm card-of-infinite-diff-finite*
by (*metis ordIso-iff-ordLeq ordLeq-transitive*)
then have $\langle |UNIV :: 'f \text{ set}| + c \mid |UNIV :: 'p \text{ set}| \leq o \mid UNIV - (\text{params } X \cup \text{params-fm } (\neg p)) \rangle$
by (*metis Set-Diff-Un*)
then have $\langle |UNIV :: 'f \text{ set}| + c \mid |UNIV :: 'p \text{ set}| \leq o \mid UNIV - \text{params } ?X \rangle$
by (*metis UN-insert insert-is-Un sup-commute*)
then have $\langle |UNIV :: ('f, 'p) \text{ fm set}| \leq o \mid UNIV - \text{params } ?X \rangle$
using *assms card-of-fm ordLeq-transitive* **by** *blast*
ultimately have $\langle \text{MCS } ?S \rangle$
using *MCS-Extend* **by** *fast*
then have $\langle p \in ?S \longleftrightarrow \mathcal{M}_{\exists} (?S) \models_{\exists} p \rangle$ **for** p
using *truth-lemma* **by** *fastforce*
then have $\langle p \in ?X \longrightarrow \mathcal{M}_{\exists} (?S) \models_{\exists} p \rangle$ **for** p
using *Extend-subset* **by** *blast*
then have $\langle \mathcal{M}_{\exists} (?S) \models_{\exists} \neg p \rangle \langle \forall q \in X. \mathcal{M}_{\exists} (?S) \models_{\exists} q \rangle$
by *blast+*
moreover from *this* **have** $\langle \mathcal{M}_{\exists} (?S) \models_{\exists} p \rangle$
using *assms(1)* **by** *blast*
ultimately show *False*
by *simp*

qed

abbreviation *valid* :: $\langle ('f, 'p) \text{ fm} \Rightarrow \text{bool} \rangle$ **where**
 $\langle \text{valid } p \equiv \forall M :: ('f \text{ tm}, -, -) \text{ model. } M \models_{\exists} p \rangle$

theorem *completeness*:

fixes $p :: \langle ('f :: \text{inf-univ}, 'p) \text{ fm} \rangle$
assumes $\langle \text{valid } p \rangle \langle |UNIV :: 'p \text{ set}| \leq o \mid |UNIV :: 'f \text{ set}| \rangle$
shows $\langle \square \vdash_{\exists} p \rangle$

proof –

have $\langle |UNIV :: 'f \text{ set}| + c \mid |UNIV :: 'p \text{ set}| \leq o \mid |UNIV :: 'f \text{ set}| \rangle$
using *assms(2)*
by (*simp add: inf-univ-class.infinite-UNIV cinfiniteness-def csum-absorb1 ordIso-imp-ordLeq*)
then show *?thesis*
using *assms strong-completeness[where X={}] infinite-UNIV-listI* **by** *auto*

qed

corollary *completeness'*:

fixes $p :: \langle ('f :: \text{inf-univ}, 'f) \text{ fm} \rangle$
assumes $\langle \text{valid } p \rangle$
shows $\langle \square \vdash_{\exists} p \rangle$
using *assms completeness[of p]* **by** *simp*

theorem *main*:
 fixes $p :: \langle 'f :: \text{inf-univ}, 'p \rangle \text{fm}$
 assumes $\langle |UNIV :: 'p \text{ set}| \leq o \mid UNIV :: 'f \text{ set} \rangle$
 shows $\langle \text{valid } p \longleftrightarrow \Box \vdash_{\exists} p \rangle$
 using *assms completeness soundness'* **by** *blast*

corollary *main'*:
 fixes $p :: \langle 'f :: \text{inf-univ}, 'f \rangle \text{fm}$
 shows $\langle \text{valid } p \longleftrightarrow \Box \vdash_{\exists} p \rangle$
 using *completeness' soundness'* **by** *blast*

end

Bibliography

- [1] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001.
- [2] J. C. Blanchette, A. Popescu, and D. Traytel. Cardinals in Isabelle/HOL. In G. Klein and R. Gamboa, editors, *Interactive Theorem Proving - 5th International Conference, ITP 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings*, volume 8558 of *Lecture Notes in Computer Science*, pages 111–127. Springer, 2014.
- [3] T. Braüner. *Hybrid Logic and its Proof-Theory*. Springer Dordrecht, first edition, 2011.
- [4] C. C. Chang and H. J. Keisler. *Model theory, Third Edition*, volume 73 of *Studies in logic and the foundations of mathematics*. North-Holland, 1992.
- [5] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.
- [6] R. M. Smullyan. *First-order logic*. Dover Publications, 1995.