

Synthetic Completeness

Asta Halkjær From

March 24, 2023

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 saturated 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 saturated when it contains a witness for every existential formula. To prove completeness using these maximal consistent saturated sets, we prove a truth lemma: every formula in an MCS has a satisfying model. Here, Hintikka 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 Hintikka set gives rise to a satisfying model and that MCSs are Hintikka 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 saturated 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 Hintikka 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 sets based on the calculus, but the other four examples derive them from the semantics in the style of the framework. The hybrid and first-order logic examples rely on saturated 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 Locale	6
1.3 Ordinal Locale	6
1.3.1 Lindenbaum Extension	7
1.3.2 Consistency	7
1.3.3 Maximality	8
1.3.4 Saturation	8
1.4 Locale with Saturation	8
1.5 Locale without Saturation	9
1.6 Truth Lemma	9
2 Derivations	11
2.1 Rearranging Assumptions	11
2.2 MCSs and Deriving Falsity	11
2.3 MCSs and Derivability	11
3 Refutations	13
3.1 Rearranging Refutations	13
3.2 MCSs and Refutability	13
4 Example: Propositional Tableau Calculus	14
4.1 Syntax	14
4.2 Semantics	14
4.3 Calculus	14
4.4 Soundness	14
4.5 Maximal Consistent Sets	15
4.6 Truth Lemma	15
4.7 Completeness	15
5 Example: Propositional Sequent Calculus	17
5.1 Syntax	17
5.2 Semantics	17
5.3 Calculus	17
5.4 Soundness	18
5.5 Maximal Consistent Sets	18

5.6	Truth Lemma	18
5.7	Completeness	18
6	Example: Modal Logic	20
6.1	Syntax	20
6.2	Semantics	20
6.3	Calculus	20
6.4	Soundness	21
6.5	Admissible rules	21
6.6	Maximal Consistent Sets	22
6.7	Truth Lemma	22
6.8	Completeness	24
7	Example: Hybrid Logic	25
7.1	Syntax	25
7.2	Semantics	25
7.3	Calculus	26
7.4	Soundness	26
7.5	Admissible Rules	26
7.6	Maximal Consistent Sets	27
7.7	Nominals	27
7.8	Truth Lemma	28
7.9	Cardinalities	29
7.10	Completeness	30
8	Example: First-Order Logic	32
8.1	Syntax	32
8.2	Semantics	32
8.3	Operations	32
8.4	Calculus	34
8.5	Soundness	34
8.6	Admissible Rules	34
8.7	Maximal Consistent Sets	35
8.8	Truth Lemma	35
8.9	Cardinalities	36
8.10	Completeness	37
	Bibliography	38

Chapter 1

Maximal Consistent Sets

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

1.1 Utility

lemma *Set-Diff-Un*: $\langle X - (Y \cup Z) = X - Y - Z \rangle$
<proof>

lemma *infinite-Diff-fin-Un*: $\langle \text{infinite } (X - Y) \implies \text{finite } Z \implies \text{infinite } (X - (Z \cup Y)) \rangle$
<proof>

lemma *infinite-Diff-subset*: $\langle \text{infinite } (X - A) \implies B \subseteq A \implies \text{infinite } (X - B) \rangle$
<proof>

lemma *finite-bound*:
fixes $X :: \langle 'a :: \text{size} \rangle \text{ set}$
assumes $\langle \text{finite } X \rangle \langle X \neq \{\} \rangle$
shows $\langle \exists x \in X. \forall y \in X. \text{size } y \leq \text{size } x \rangle$
<proof>

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>

lemma *split-finite-sets*:
assumes $\langle \text{finite } A \rangle \langle \text{finite } B \rangle$
assumes $\langle A \subseteq B \cup S \rangle$
shows $\langle \exists B' C. \text{finite } C \wedge (B' \cup C = A) \wedge B' \subseteq B \wedge C \subseteq S \rangle$
<proof>

lemma *split-list*:
assumes $\langle \text{set } A \subseteq \text{set } B \cup S \rangle$
shows $\langle \exists B' C. \text{set } (B' @ C) = \text{set } A \wedge \text{set } B' \subseteq \text{set } B \wedge \text{set } C \subseteq S \rangle$
<proof>

lemma *struct-split*:
assumes $\langle \bigwedge A B. P A \implies \text{set } A \subseteq \text{set } B \implies P B \rangle \langle P A \rangle \langle \text{set } A \subseteq \text{set } B \cup X \rangle$
shows $\langle \exists C. \text{set } C \subseteq X \wedge P (B @ C) \rangle$
<proof>

context *wo-rel* **begin**

lemma *underS-bound*: $\langle a \in \text{underS } n \implies b \in \text{underS } n \implies a \in \text{under } b \vee b \in \text{under } a \rangle$
 $\langle \text{proof} \rangle$

lemma *finite-underS-bound*:
assumes $\langle \text{finite } X \rangle \langle X \subseteq \text{underS } n \rangle \langle X \neq \{\} \rangle$
shows $\langle \exists a \in X. \forall b \in X. b \in \text{under } a \rangle$
 $\langle \text{proof} \rangle$

lemma *finite-bound-under*:
assumes $\langle \text{finite } p \rangle \langle p \subseteq (\bigcup n \in \text{Field } r. f n) \rangle$
shows $\langle \exists m. p \subseteq (\bigcup n \in \text{under } m. f n) \rangle$
 $\langle \text{proof} \rangle$

lemma *underS-trans*: $\langle a \in \text{underS } b \implies b \in \text{underS } c \implies a \in \text{underS } c \rangle$
 $\langle \text{proof} \rangle$

end

lemma *card-of-infinite-smaller-Union*:
assumes $\langle \forall x. |f x| < o |X| \rangle \langle \text{infinite } X \rangle$
shows $\langle |\bigcup x \in X. f x| \leq o |X| \rangle$
 $\langle \text{proof} \rangle$

lemma *card-of-params-marker-lists*:
assumes $\langle \text{infinite } (\text{UNIV} :: 'i \text{ set}) \rangle \langle |\text{UNIV} :: 'm \text{ set}| \leq o |\text{UNIV} :: \text{nat set}| \rangle$
shows $\langle |\text{UNIV} :: ('i + 'm \times \text{nat}) \text{ list set}| \leq o |\text{UNIV} :: 'i \text{ set}| \rangle$
 $\langle \text{proof} \rangle$

1.2 Base Locale

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

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

end

1.3 Ordinal Locale

locale *MCS-Lim-Ord* = *MCS-Base* +
fixes *r* :: $\langle 'a \text{ rel} \rangle$
assumes *WELL*: $\langle \text{Well-order } r \rangle$
and *Cinfinite-r*: $\langle \text{Cinfinite } r \rangle$
fixes *params* :: $\langle 'a \Rightarrow 'i \text{ set} \rangle$
and *witness* :: $\langle 'a \Rightarrow 'a \text{ set} \Rightarrow 'a \text{ set} \rangle$
assumes *finite-params*: $\langle \bigwedge p. \text{finite } (\text{params } p) \rangle$
and *finite-witness-params*: $\langle \bigwedge p S. \text{finite } (\bigcup q \in \text{witness } p S. \text{params } q) \rangle$
and *consistent-witness*: $\langle \bigwedge p S. \text{consistent } (\{p\} \cup S) \rangle$

$\implies \text{infinite } (UNIV - (\bigcup q \in S. \text{params } q))$
 $\implies \text{consistent } (\text{witness } p \ S \cup \{p\} \cup S)$

begin

lemma *wo-rel-r*: $\langle \text{wo-rel } r \rangle$
 $\langle \text{proof} \rangle$

lemma *isLimOrd-r*: $\langle \text{isLimOrd } r \rangle$
 $\langle \text{proof} \rangle$

1.3.1 Lindenbaum Extension

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

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

definition *extendL* :: $\langle ('a \Rightarrow 'a \text{ set}) \Rightarrow 'a \Rightarrow 'a \text{ set} \rangle$ **where**
 $\langle \text{extendL } \text{rec } n \equiv \bigcup m \in \text{underS } r \ n. \text{rec } m \rangle$

definition *extend* :: $\langle 'a \text{ set} \Rightarrow 'a \Rightarrow 'a \text{ set} \rangle$ **where**
 $\langle \text{extend } S \ n \equiv \text{worecZSL } r \ S \ \text{extendS } \ \text{extendL } \ n \rangle$

lemma *adm-woL-extendL*: $\langle \text{adm-woL } r \ \text{extendL} \rangle$
 $\langle \text{proof} \rangle$

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

lemma *extend-subset*: $\langle n \in \text{Field } r \implies S \subseteq \text{extend } S \ n \rangle$
 $\langle \text{proof} \rangle$

lemma *Extend-subset'*: $\langle \text{Field } r \neq \{\} \implies S \subseteq \text{Extend } S \rangle$
 $\langle \text{proof} \rangle$

lemma *extend-underS*: $\langle m \in \text{underS } r \ n \implies \text{extend } S \ m \subseteq \text{extend } S \ n \rangle$
 $\langle \text{proof} \rangle$

lemma *extend-under*: $\langle m \in \text{under } r \ n \implies \text{extend } S \ m \subseteq \text{extend } S \ n \rangle$
 $\langle \text{proof} \rangle$

1.3.2 Consistency

lemma *params-origin*:
assumes $\langle a \in \text{paramss } (\text{extend } S \ n) \rangle$
shows $\langle a \in \text{paramss } S \vee (\exists m \in \text{underS } r \ n. a \in \text{paramss } (\text{witness } m \ (\text{extend } S \ m) \cup \{m\})) \rangle$
 $\langle \text{proof} \rangle$

lemma *consistent-extend*:
assumes $\langle \text{consistent } S \rangle \langle r \leq o \mid UNIV - \text{paramss } S \rangle$
shows $\langle \text{consistent } (\text{extend } S \ n) \rangle$
 $\langle \text{proof} \rangle$

lemma *consistent-Extend*:
assumes $\langle \text{consistent } S \rangle \langle r \leq o \mid UNIV - \text{paramss } S \rangle$

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

lemma *Extend-bound*: $\langle n \in \text{Field } r \implies \text{extend } S \ n \subseteq \text{Extend } S \rangle$
 $\langle \text{proof} \rangle$

1.3.3 Maximality

definition *maximal'* :: $\langle 'a \text{ set} \Rightarrow \text{bool} \rangle$ **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*: $\langle \text{maximal}' (\text{Extend } S) \rangle$
 $\langle \text{proof} \rangle$

1.3.4 Saturation

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

lemma *saturated'-Extend*:
assumes $\langle \text{consistent } (\text{Extend } S) \rangle$
shows $\langle \text{saturated}' (\text{Extend } S) \rangle$
 $\langle \text{proof} \rangle$

end

1.4 Locale with Saturation

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

sublocale *MCS-Saturation* \subseteq *MCS-Lim-Ord* - $\langle |\text{UNIV}| \rangle$
 $\langle \text{proof} \rangle$

context *MCS-Saturation* **begin**

theorem *Extend-subset*: $\langle S \subseteq \text{Extend } S \rangle$
 $\langle \text{proof} \rangle$

lemma *maximal-maximal'*: $\langle \text{maximal } S \longleftrightarrow \text{maximal}' S \rangle$
 $\langle \text{proof} \rangle$

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

definition *saturated* :: $\langle 'a \text{ set} \Rightarrow \text{bool} \rangle$ **where**
 $\langle \text{saturated } S \equiv \forall p \in S. \exists S'. \text{witness } p \ S' \subseteq S \rangle$

lemma *saturated-saturated'*: $\langle \text{saturated } S \longleftrightarrow \text{saturated}' S \rangle$

⟨proof⟩

lemma *saturated-Extend*:

assumes ⟨consistent (Extend S)⟩

shows ⟨saturated (Extend S)⟩

⟨proof⟩

theorem *MCS-Extend*:

assumes ⟨consistent S⟩ ⟨|UNIV :: 'a set| ≤_o |UNIV - paramss S|⟩

shows ⟨consistent (Extend S)⟩ ⟨maximal (Extend S)⟩ ⟨saturated (Extend S)⟩

⟨proof⟩

end

1.5 Locale without Saturation

locale *MCS-No-Saturation* = *MCS-Base* +

assumes ⟨infinite (UNIV :: 'a set)⟩

sublocale *MCS-No-Saturation* ⊆ *MCS-Saturation consistent* ⟨λ-. {} :: 'a set⟩ ⟨λ- -. {}⟩

⟨proof⟩

context *MCS-No-Saturation* **begin**

lemma *always-saturated [simp]*: ⟨saturated H⟩

⟨proof⟩

theorem *MCS-Extend'*:

assumes ⟨consistent S⟩

shows ⟨consistent (Extend S)⟩ ⟨maximal (Extend S)⟩

⟨proof⟩

end

1.6 Truth Lemma

locale *Truth-Base* =

fixes *semics* :: ⟨'model ⇒ ('model ⇒ 'fm ⇒ bool) ⇒ 'fm ⇒ bool⟩

and *semantics* :: ⟨'model ⇒ 'fm ⇒ bool⟩

and *models-from* :: ⟨'a set ⇒ 'model set⟩

and *rel* :: ⟨'a set ⇒ 'model ⇒ 'fm ⇒ bool⟩

assumes *semics-semantics*: ⟨semantics M p ⟷ semics M semantics p⟩

and *Hintikka-model*: ⟨∧H M p. ∀ M ∈ models-from H. ∀ p. semics M (rel H) p ⟷ rel H M p ⟹
M ∈ models-from H ⟹ semantics M p ⟷ rel H M p⟩

locale *Truth-Saturation* = *MCS-Saturation* + *Truth-Base* +

assumes *MCS-Hintikka*: ⟨∧H. consistent H ⟹ maximal H ⟹ saturated H ⟹

∀ M ∈ models-from H. ∀ p. semics M (rel H) p ⟷ rel H M p⟩

begin

theorem *truth-lemma-saturation*:

assumes ⟨consistent H⟩ ⟨maximal H⟩ ⟨saturated H⟩ ⟨M ∈ models-from H⟩

shows ⟨semantics M p ⟷ rel H M p⟩

⟨proof⟩

end

locale *Truth-No-Saturation* = *MCS-No-Saturation* + *Truth-Base* +
assumes *MCS-Hintikka*: $\langle \bigwedge H. \text{consistent } H \implies \text{maximal } H \implies$
 $\forall M \in \text{models-from } H. \forall p. \text{semics } M (\text{rel } H) p \longleftrightarrow \text{rel } H M p \rangle$
begin

theorem *truth-lemma-no-saturation*:
assumes $\langle \text{consistent } H \rangle \langle \text{maximal } H \rangle \langle M \in \text{models-from } H \rangle$
shows $\langle \text{semantics } M p \longleftrightarrow \text{rel } H M p \rangle$
 $\langle \text{proof} \rangle$

end

end

Chapter 2

Derivations

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

2.1 Rearranging Assumptions

locale *Derivations* =

fixes *derive* :: $\langle 'a \text{ list} \Rightarrow 'a \Rightarrow \text{bool} \rangle$

assumes *derive-struct*: $\langle \bigwedge A B p. \text{derive } A p \implies \text{set } A \subseteq \text{set } B \implies \text{derive } B p \rangle$

begin

theorem *derive-split*:

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

shows $\langle \exists C. \text{set } C \subseteq X \wedge \text{derive } (B @ C) p \rangle$

$\langle \text{proof} \rangle$

corollary *derive-split1*:

assumes $\langle \text{set } A \subseteq \{q\} \cup X \rangle \langle \text{derive } A p \rangle$

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

$\langle \text{proof} \rangle$

end

2.2 MCSs and Deriving Falsity

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

fixes *fls*

assumes *consistent-derive-fls*: $\langle \bigwedge S. \text{consistent } S = (\nexists S'. \text{set } S' \subseteq S \wedge \text{derive } S' \text{ fls}) \rangle$

begin

theorem *MCS-derive-fls*:

assumes $\langle \text{consistent } S \rangle \langle \text{maximal } S \rangle$

shows $\langle p \notin S \iff (\exists S'. \text{set } S' \subseteq S \wedge \text{derive } (p \# S') \text{ fls}) \rangle$

$\langle \text{proof} \rangle$

end

2.3 MCSs and Derivability

locale *Derivations-MCS-Cut* = *Derivations-MCS* +

assumes *derive-assm*: $\langle \bigwedge A p. p \in \text{set } A \implies \text{derive } A p \rangle$

and *derive-cut*: $\langle \wedge A B p q. \text{derive } A p \implies \text{derive } (p \# B) q \implies \text{derive } (A @ B) q \rangle$
begin

theorem *MCS-derive*:

assumes $\langle \text{consistent } S \rangle \langle \text{maximal } S \rangle$

shows $\langle p \in S \longleftrightarrow (\exists S'. \text{set } S' \subseteq S \wedge \text{derive } S' p) \rangle$

$\langle \text{proof} \rangle$

end

end

Chapter 3

Refutations

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

3.1 Rearranging Refutations

locale *Refutations* =

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

assumes *refute-struct*: $\langle \bigwedge A B. \text{refute } A \implies \text{set } A \subseteq \text{set } B \implies \text{refute } B \rangle$

begin

theorem *refute-split*:

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

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

$\langle \text{proof} \rangle$

corollary *refute-split1*:

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

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

$\langle \text{proof} \rangle$

end

3.2 MCSs and Refutability

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

assumes *consistent-refute*: $\langle \bigwedge S. \text{consistent } S = (\nexists S'. \text{set } S' \subseteq S \wedge \text{refute } S') \rangle$

begin

theorem *MCS-refute*:

assumes $\langle \text{consistent } S \rangle \langle \text{maximal } S \rangle$

shows $\langle p \notin S \iff (\exists S'. \text{set } S' \subseteq S \wedge \text{refute } (p \# S')) \rangle$

$\langle \text{proof} \rangle$

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⟩ (⟨[-]⟩) where
  ⟨[[I]] (⟨!P⟩) ⟷ I P⟩
| ⟨[[I]] (⟨¬ p⟩) ⟷ ¬ [[I]] p⟩
| ⟨[[I]] (p ⟶ q) ⟷ [[I]] p ⟶ [[I]] q⟩
```

4.3 Calculus

```
inductive Calculus :: ⟨'p fm list ⇒ bool⟩ (⟨⊢T -⟩ [50] 50) where
  Axiom [intro]: ⟨⊢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⟩
| Weaken: ⟨⊢T A ⟹ set A ⊆ set B ⟹ ⊢T B⟩
```

lemma *Weaken2*:

```
  assumes ⟨⊢T p # A⟩ ⟨⊢T q # B⟩
  shows ⟨⊢T p # A @ B ∧ ⊢T q # A @ B⟩
  ⟨proof⟩
```

4.4 Soundness

```
theorem soundness: ⟨⊢T A ⟹ ∃ p ∈ set A. ¬ [[I]] p⟩
  ⟨proof⟩
```

corollary *soundness'*: $\langle \vdash_T [\neg p] \implies \llbracket I \rrbracket p \rangle$
 $\langle \text{proof} \rangle$

corollary $\langle \neg (\vdash_T []) \rangle$
 $\langle \text{proof} \rangle$

4.5 Maximal Consistent Sets

definition *consistent* :: $\langle 'p \text{ fm set} \implies \text{bool} \rangle$ **where**
 $\langle \text{consistent } S \equiv \nexists S'. \text{ set } S' \subseteq S \wedge \vdash_T S' \rangle$

interpretation *MCS-No-Saturation consistent*
 $\langle \text{proof} \rangle$

interpretation *Refutations-MCS Calculus consistent*
 $\langle \text{proof} \rangle$

4.6 Truth Lemma

abbreviation (*input*) *hmodel* :: $\langle 'p \text{ fm set} \implies 'p \text{ model} \rangle$ **where**
 $\langle \text{hmodel } H \equiv \lambda P. \nexists P \in H \rangle$

locale *Hintikka* =
fixes $H :: \langle 'a \text{ fm set} \rangle$
assumes *AxiomH*: $\langle \bigwedge P. \nexists P \in H \implies \neg \nexists P \in H \implies \text{False} \rangle$
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 $\langle (p \in H \longrightarrow \llbracket \text{hmodel } H \rrbracket p) \wedge (\neg p \in H \longrightarrow \neg \llbracket \text{hmodel } H \rrbracket p) \rangle$
 $\langle \text{proof} \rangle$

lemma *MCS-Hintikka*:
assumes $\langle \text{consistent } H \rangle \langle \text{maximal } H \rangle$
shows $\langle \text{Hintikka } H \rangle$
 $\langle \text{proof} \rangle$

lemma *truth-lemma*:
assumes $\langle \text{consistent } H \rangle \langle \text{maximal } H \rangle \langle p \in H \rangle$
shows $\langle \llbracket \text{hmodel } H \rrbracket p \rangle$
 $\langle \text{proof} \rangle$

4.7 Completeness

theorem *strong-completeness*:
assumes $\langle \forall M :: 'p \text{ model}. (\forall q \in X. \llbracket M \rrbracket q) \longrightarrow \llbracket M \rrbracket p \rangle$
shows $\langle \exists A. \text{ set } A \subseteq X \wedge \vdash_T \neg p \# A \rangle$
 $\langle \text{proof} \rangle$

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

theorem *completeness*:

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

shows $\langle \vdash_T [\neg p] \rangle$

$\langle \text{proof} \rangle$

theorem *main*: $\langle \text{valid } p \iff \vdash_T [\neg p] \rangle$

$\langle \text{proof} \rangle$

end

Chapter 5

Example: Propositional Sequent Calculus

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

5.1 Syntax

```
datatype 'p fm
  = Fls (⟨⊥⟩)
  | Pro 'p (⟨‡⟩)
  | Imp 'p fm ⟨'p fm⟩ (infixr ⟨⟶⟩ 55)
```

abbreviation *Neg* (⟨¬ → [70] 70) **where** $\neg p \equiv p \longrightarrow \perp$

5.2 Semantics

type-synonym 'p model = ⟨'p ⇒ bool⟩

```
fun semantics :: ⟨'p model ⇒ 'p fm ⇒ bool⟩ (⟨[[·]]⟩) where
  ⟨[[⊥]] ⊥ ⟷ False⟩
  | ⟨[[I]] (‡P) ⟷ I P⟩
  | ⟨[[I]] (p ⟶ q) ⟷ [[I] p ⟶ [[I] q]⟩
```

5.3 Calculus

inductive *Calculus* :: ⟨'p fm list ⇒ 'p fm list ⇒ bool⟩ (⟨⊢_S → [50, 50] 50) **where**

```
  Axiom [intro]: ⟨p # A ⊢S p # B⟩
  | FlsL [intro]: ⟨⊥ # A ⊢S B⟩
  | FlsR [dest]: ⟨A ⊢S ⊥ # B ⟹ A ⊢S B⟩
  | ImpL [intro]: ⟨A ⊢S p # B ⟹ q # A ⊢S B ⟹ (p ⟶ q) # A ⊢S B⟩
  | ImpR [intro]: ⟨p # A ⊢S q # B ⟹ A ⊢S (p ⟶ q) # B⟩
  | Cut [elim]: ⟨A ⊢S p # B ⟹ p # C ⊢S D ⟹ A @ C ⊢S B @ D⟩
  | WeakenL: ⟨A ⊢S B ⟹ set A ⊆ set A' ⟹ A' ⊢S B⟩
  | WeakenR: ⟨A ⊢S B ⟹ set B ⊆ set B' ⟹ A ⊢S B'⟩
```

lemma *Boole*: ⟨¬ p # A ⊢_S [] ⟹ A ⊢_S [p]⟩
 ⟨proof⟩

5.4 Soundness

theorem *soundness*: $\langle A \vdash_S B \implies \forall q \in \text{set } A. \llbracket I \rrbracket q \implies \exists p \in \text{set } B. \llbracket I \rrbracket p \rangle$
 $\langle \text{proof} \rangle$

corollary *soundness'*: $\langle \llbracket \] \vdash_S [p] \implies \llbracket I \rrbracket p \rangle$
 $\langle \text{proof} \rangle$

corollary $\langle \neg (\llbracket \] \vdash_S \llbracket \]) \rangle$
 $\langle \text{proof} \rangle$

5.5 Maximal Consistent Sets

definition *consistent* :: $\langle 'p \text{ fm set} \Rightarrow \text{bool} \rangle$ **where**
 $\langle \text{consistent } S \equiv \nexists S'. \text{ set } S' \subseteq S \wedge S' \vdash_S [\perp] \rangle$

interpretation *MCS-No-Saturation consistent*
 $\langle \text{proof} \rangle$

interpretation *Derivations-MCS-Cut* $\langle \lambda A p. A \vdash_S [p] \rangle$ *consistent* $\langle \perp \rangle$
 $\langle \text{proof} \rangle$

5.6 Truth Lemma

abbreviation *hmodel* :: $\langle 'p \text{ fm set} \Rightarrow 'p \text{ model} \rangle$ **where**
 $\langle \text{hmodel } H \equiv \lambda P. \ddagger P \in H \rangle$

fun *semics* :: $\langle 'p \text{ model} \Rightarrow ('p \text{ model} \Rightarrow 'p \text{ fm} \Rightarrow \text{bool}) \Rightarrow 'p \text{ fm} \Rightarrow \text{bool} \rangle$ **where**
 $\langle \text{semics } - \text{ } \perp \longleftrightarrow \text{False} \rangle$
 $| \langle \text{semics } I \text{ } (\ddagger P) \longleftrightarrow I P \rangle$
 $| \langle \text{semics } I \text{ rel } (p \longrightarrow q) \longleftrightarrow \text{rel } I p \longrightarrow \text{rel } I q \rangle$

fun *rel* :: $\langle 'p \text{ fm set} \Rightarrow 'p \text{ model} \Rightarrow 'p \text{ fm} \Rightarrow \text{bool} \rangle$ **where**
 $\langle \text{rel } H \text{ } p = (p \in H) \rangle$

theorem *Hintikka-model'*:
assumes $\langle \bigwedge p. \text{semics } (\text{hmodel } H) (\text{rel } H) p \longleftrightarrow p \in H \rangle$
shows $\langle p \in H \longleftrightarrow \llbracket \text{hmodel } H \rrbracket p \rangle$
 $\langle \text{proof} \rangle$

lemma *Hintikka-Extend*:
assumes $\langle \text{consistent } H \rangle \langle \text{maximal } H \rangle$
shows $\langle \text{semics } (\text{hmodel } H) (\text{rel } H) p \longleftrightarrow p \in H \rangle$
 $\langle \text{proof} \rangle$

interpretation *Truth-No-Saturation consistent semics semantics* $\langle \lambda H. \{ \text{hmodel } H \} \rangle$ *rel*
 $\langle \text{proof} \rangle$

5.7 Completeness

theorem *strong-completeness*:
assumes $\langle \forall M :: 'p \text{ model}. (\forall q \in X. \llbracket M \rrbracket q) \longrightarrow \llbracket M \rrbracket p \rangle$
shows $\langle \exists A. \text{ set } A \subseteq X \wedge A \vdash_S [p] \rangle$
 $\langle \text{proof} \rangle$

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

theorem *completeness*:

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

shows $\langle [] \vdash_S [p] \rangle$

$\langle \text{proof} \rangle$

theorem *main*: $\langle \text{valid } p \longleftrightarrow [] \vdash_S [p] \rangle$

$\langle \text{proof} \rangle$

end

Chapter 6

Example: Modal Logic

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

6.1 Syntax

datatype $\langle 'i, 'p \rangle$ *fm*
= *Fls* $\langle \perp \rangle$
| *Pro* $\langle 'p \rangle$ $\langle \dagger \rangle$
| *Imp* $\langle ('i, 'p) \text{ fm} \rangle$ $\langle ('i, 'p) \text{ fm} \rangle$ (**infixr** $\langle \longrightarrow \rangle$ 55)
| *Box* $\langle 'i \rangle$ $\langle ('i, 'p) \text{ fm} \rangle$ $\langle \Box \rangle$

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

6.2 Semantics

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

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

fun *semantics* :: $\langle ('i, 'p, 'w) \text{ ctx} \Rightarrow ('i, 'p) \text{ fm} \Rightarrow \text{bool} \rangle$ $\langle \langle - \models - \rangle [50, 50] 50 \rangle$ **where**
 $\langle - \models \perp \longleftrightarrow \text{False} \rangle$
| $\langle \langle M, w \rangle \models \dagger P \longleftrightarrow \mathcal{V} M w P \rangle$
| $\langle \langle M, w \rangle \models p \longrightarrow q \longleftrightarrow (M, w) \models p \longrightarrow (M, w) \models q \rangle$
| $\langle \langle M, w \rangle \models \Box i p \longleftrightarrow (\forall v \in \mathcal{W} M \cap \mathcal{R} M i w. (M, v) \models p) \rangle$

6.3 Calculus

primrec *eval* :: $\langle ('p \Rightarrow \text{bool}) \Rightarrow (('i, 'p) \text{ fm} \Rightarrow \text{bool}) \Rightarrow ('i, 'p) \text{ fm} \Rightarrow \text{bool} \rangle$ **where**
 $\langle \text{eval } - \text{ } \perp = \text{False} \rangle$
| $\langle \text{eval } g \text{ } (\dagger P) = g P \rangle$
| $\langle \text{eval } g \text{ } h (p \longrightarrow q) = (\text{eval } g \text{ } h p \longrightarrow \text{eval } g \text{ } h q) \rangle$
| $\langle \text{eval } - \text{ } h (\Box i p) = h (\Box i p) \rangle$

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

inductive *Calculus* :: $\langle ('i, 'p) \text{ fm} \Rightarrow \text{bool} \rangle$ $\langle \langle \vdash_{\Box} - \rangle [50] 50 \rangle$ **where**
A1: $\langle \text{tautology } p \implies \vdash_{\Box} p \rangle$
| *A2*: $\langle \vdash_{\Box} \Box i (p \longrightarrow q) \longrightarrow \Box i p \longrightarrow \Box i q \rangle$

| *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* :: $\langle ('i, 'p) \text{ fm list} \implies ('i, 'p) \text{ fm} \implies ('i, 'p) \text{ fm} \rangle$ (**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* ($\langle \vdash_{\square} \rightarrow [50, 50] 50 \rangle$) **where**
 $\langle A \vdash_{\square} p \equiv \vdash_{\square} A \rightsquigarrow p \rangle$

6.4 Soundness

lemma *eval-antics*: $\langle \text{eval } (g w) (\lambda q. (\text{Model } W r g, w) \models q) p = ((\text{Model } W r g, w) \models p) \rangle$
 $\langle \text{proof} \rangle$

lemma *tautology*:

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

theorem *soundness*:

assumes $\langle \bigwedge M w p. A p \implies w \in \mathcal{W} M \implies (M, w) \models p \rangle$
shows $\langle \vdash_{\square} p \implies w \in \mathcal{W} M \implies (M, w) \models p \rangle$
 $\langle \text{proof} \rangle$

6.5 Admissible rules

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

lemma *K-imply-Cons*:

assumes $\langle A \vdash_{\square} q \rangle$
shows $\langle p \# A \vdash_{\square} q \rangle$
 $\langle \text{proof} \rangle$

lemma *K-right-mp*:

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

lemma *deduct1*: $\langle A \vdash_{\square} p \longrightarrow q \implies p \# A \vdash_{\square} q \rangle$
 $\langle \text{proof} \rangle$

lemma *imply-append [iff]*: $\langle (A @ B \rightsquigarrow r) = (A \rightsquigarrow B \rightsquigarrow r) \rangle$
 $\langle \text{proof} \rangle$

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

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

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

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

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

lemma *K-Boole*:
assumes $\langle (\neg p) \# A \vdash_{\square} \perp \rangle$
shows $\langle A \vdash_{\square} p \rangle$
 $\langle \text{proof} \rangle$

lemma *K-distrib-K-imp*:
assumes $\langle \vdash_{\square} \square i (A \rightsquigarrow q) \rangle$
shows $\langle \text{map } (\square i) A \vdash_{\square} \square i q \rangle$
 $\langle \text{proof} \rangle$

interpretation *Derivations Calculus-assms*
 $\langle \text{proof} \rangle$

6.6 Maximal Consistent Sets

definition *consistent* :: $\langle ('i, 'p) \text{ fm set} \Rightarrow \text{bool} \rangle$ **where**
 $\langle \text{consistent } S \equiv \nexists S'. \text{ set } S' \subseteq S \wedge S' \vdash_{\square} \perp \rangle$

interpretation *MCS-No-Saturation consistent*
 $\langle \text{proof} \rangle$

interpretation *Derivations-MCS-Cut Calculus-assms consistent* $\langle \perp \rangle$
 $\langle \text{proof} \rangle$

lemma *exists-finite-inconsistent*:
assumes $\langle \neg \text{consistent } (\{\neg p\} \cup V) \rangle$
obtains W **where** $\langle \{\neg p\} \cup W \subseteq \{\neg p\} \cup V \rangle \langle (\neg p) \notin W \rangle \langle \text{finite } W \rangle \langle \neg \text{consistent } (\{\neg p\} \cup W) \rangle$
 $\langle \text{proof} \rangle$

lemma *MCS-consequent*:
assumes $\langle \text{consistent } V \rangle \langle \text{maximal } V \rangle \langle p \longrightarrow q \in V \rangle \langle p \in V \rangle$
shows $\langle q \in V \rangle$
 $\langle \text{proof} \rangle$

theorem *deriv-in-maximal*:
assumes $\langle \text{consistent } V \rangle \langle \text{maximal } V \rangle \langle \vdash_{\square} p \rangle$
shows $\langle p \in V \rangle$
 $\langle \text{proof} \rangle$

theorem *exactly-one-in-maximal*:
assumes $\langle \text{consistent } V \rangle \langle \text{maximal } V \rangle$
shows $\langle p \in V \longleftrightarrow (\neg p) \notin V \rangle$
 $\langle \text{proof} \rangle$

6.7 Truth Lemma

abbreviation *val* :: $\langle ('i, 'p) \text{ fm set} \Rightarrow 'p \Rightarrow \text{bool} \rangle$ **where**
 $\langle \text{val } V P \equiv \nexists P \in V \rangle$

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

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

abbreviation *mcss* :: $\langle ('i, 'p) \text{ fm set set} \rangle$ **where**
 $\langle \text{mcss} \equiv \{W. \text{consistent } W \wedge \text{maximal } W\} \rangle$

abbreviation *canonical* :: $\langle ('i, 'p, ('i, 'p) \text{ fm set}) \text{ model} \rangle$ **where**
 $\langle \text{canonical} \equiv \text{Model } \text{mcss } \text{reach } \text{val} \rangle$

fun *semics* ::
 $\langle ('i, 'p, 'w) \text{ ctx} \Rightarrow (('i, 'p, 'w) \text{ ctx} \Rightarrow ('i, 'p) \text{ fm} \Rightarrow \text{bool}) \Rightarrow ('i, 'p) \text{ fm} \Rightarrow \text{bool} \rangle$ **where**
 $\langle \text{semics } - \text{ } \perp \longleftrightarrow \text{False} \rangle$
 $| \langle \text{semics } (M, w) \text{ } - (\ddagger P) \longleftrightarrow \mathcal{V} \text{ } M \text{ } w \text{ } P \rangle$
 $| \langle \text{semics } (M, w) \text{ rel } (p \longrightarrow q) \longleftrightarrow \text{rel } (M, w) \text{ } p \longrightarrow \text{rel } (M, w) \text{ } q \rangle$
 $| \langle \text{semics } (M, w) \text{ rel } (\Box i \ p) \longleftrightarrow (\forall v \in \mathcal{W} \text{ } M \cap \mathcal{R} \text{ } M \text{ } i \text{ } w. \text{rel } (M, v) \text{ } p) \rangle$

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

lemma *Hintikka-model'*:

fixes $V :: \langle ('i, 'p) \text{ fm set} \rangle$

assumes $\langle \bigwedge (V :: ('i, 'p) \text{ fm set}) \text{ } p. V \in \text{mcss} \implies \text{semics } (\text{canonical}, V) \text{ } (\text{rel } H) \text{ } p \longleftrightarrow p \in V \rangle$

shows $\langle V \in \text{mcss} \implies (\text{canonical}, V) \models p \longleftrightarrow p \in V \rangle$

$\langle \text{proof} \rangle$

lemma *maximal-extension*:

assumes $\langle \text{consistent } V \rangle$

shows $\langle \exists W. V \subseteq W \wedge \text{consistent } W \wedge \text{maximal } W \rangle$

$\langle \text{proof} \rangle$

lemma *Hintikka-canonical*:

assumes $\langle V \in \text{mcss} \rangle$

shows $\langle \text{semics } (\text{canonical}, V) \text{ } (\text{rel } H) \text{ } p \longleftrightarrow \text{rel } H \text{ } (\text{canonical}, V) \text{ } p \rangle$

$\langle \text{proof} \rangle$

interpretation *Truth-No-Saturation consistent semics semantics*

$\langle \lambda-. \{(\text{canonical}, V) \mid V. V \in \text{mcss} \} \rangle \text{ rel}$

$\langle \text{proof} \rangle$

lemma *Truth-lemma*:

assumes $\langle \text{consistent } V \rangle \langle \text{maximal } V \rangle$

shows $\langle (\text{canonical}, V) \models p \longleftrightarrow p \in V \rangle$

$\langle \text{proof} \rangle$

lemma *canonical-model*:

assumes $\langle \text{consistent } S \rangle \langle p \in S \rangle$

defines $\langle V \equiv \text{Extend } S \rangle$ **and** $\langle M \equiv \text{canonical} \rangle$

shows $\langle (M, V) \models p \rangle \langle \text{consistent } V \rangle \langle \text{maximal } V \rangle$

$\langle \text{proof} \rangle$

6.8 Completeness

theorem *strong-completeness*:

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

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

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

$\langle \text{proof} \rangle$

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

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

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

$\langle \text{proof} \rangle$

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

$\langle \text{proof} \rangle$

end

Chapter 7

Example: Hybrid Logic

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

7.1 Syntax

datatype (*nominals-fm*: 'i, 'p) *fm*
= *Fls* ($\langle \perp \rangle$)
| *Pro* 'p ($\langle \ddagger \rangle$)
| *Nom* 'i ($\langle \cdot \rangle$)
| *Imp* ($\langle 'i, 'p \rangle$ *fm*) ($\langle 'i, 'p \rangle$ *fm*) (**infixr** $\langle \longrightarrow \rangle$ 55)
| *Dia* ($\langle 'i, 'p \rangle$ *fm*) ($\langle \diamond \rangle$)
| *Sat* 'i ($\langle 'i, 'p \rangle$ *fm*) ($\langle @ \rangle$)

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

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

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

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

7.2 Semantics

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

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

fun *semantics* :: $\langle ('i, 'p, 'w) \text{ ctx} \Rightarrow ('i, 'p) \text{ fm} \Rightarrow \text{bool} \rangle$ ($\langle \cdot \models \rightarrow [50, 50] 50 \rangle$) **where**
 $\langle (M, g, w) \models \perp \longleftrightarrow \text{False} \rangle$
 $\langle (M, \cdot, w) \models \ddagger P \longleftrightarrow V M w P \rangle$
 $\langle (\cdot, g, w) \models \cdot i \longleftrightarrow w = g i \rangle$
 $\langle (M, g, w) \models (p \longrightarrow q) \longleftrightarrow (M, g, w) \models p \longrightarrow (M, g, w) \models q \rangle$

| $\langle (M, g, w) \models \diamond p \longleftrightarrow (\exists v \in R M w. (M, g, v) \models p) \rangle$
| $\langle (M, g, -) \models @i p \longleftrightarrow (M, g, g i) \models p \rangle$

lemma semantics-fresh: $\langle i \notin \text{nominals-fm } p \implies ((M, g, w) \models p) = ((M, g(i := v), w) \models p) \rangle$
 $\langle \text{proof} \rangle$

lemma semantics-fresh-lbd:
 $\langle k \notin \text{nominals-lbd } (i, p) \implies ((M, g, w) \models p) = ((M, g(k := v), w) \models p) \rangle$
 $\langle \text{proof} \rangle$

7.3 Calculus

inductive Calculus :: $\langle ('i, 'p) \text{ lbd list} \implies ('i, 'p) \text{ lbd} \implies \text{bool} \rangle \langle \cdot \vdash_{@} \cdot \rightarrow [50, 50] 50 \rangle$ **where**

Assm [intro]: $\langle (i, p) \in \text{set } A \implies A \vdash_{@} (i, p) \rangle$
| *Ref* [intro]: $\langle A \vdash_{@} (i, \cdot i) \rangle$
| *Nom* [intro]: $\langle A \vdash_{@} (i, \cdot k) \implies A \vdash_{@} (i, p) \implies A \vdash_{@} (k, p) \rangle$
| *FlsE* [elim]: $\langle A \vdash_{@} (i, \perp) \implies A \vdash_{@} (k, p) \rangle$
| *ImpI* [intro]: $\langle (i, p) \# A \vdash_{@} (i, q) \implies A \vdash_{@} (i, p \longrightarrow q) \rangle$
| *ImpE* [elim]: $\langle A \vdash_{@} (i, p \longrightarrow q) \implies A \vdash_{@} (i, p) \implies A \vdash_{@} (i, q) \rangle$
| *SatI* [intro]: $\langle A \vdash_{@} (i, p) \implies A \vdash_{@} (k, @i p) \rangle$
| *SatE* [elim]: $\langle A \vdash_{@} (i, @k p) \implies A \vdash_{@} (k, p) \rangle$
| *DiaI* [intro]: $\langle A \vdash_{@} (i, \diamond (\cdot k)) \implies A \vdash_{@} (k, p) \implies A \vdash_{@} (i, \diamond p) \rangle$
| *DiaE* [elim]: $\langle A \vdash_{@} (i, \diamond p) \implies k \notin \text{nominals } (\{(i, p), (j, q)\} \cup \text{set } A) \implies$
 $(k, p) \# (i, \diamond (\cdot k)) \# A \vdash_{@} (j, q) \implies A \vdash_{@} (j, q) \rangle$
| *Clas*: $\langle (i, p \longrightarrow q) \# A \vdash_{@} (i, p) \implies A \vdash_{@} (i, p) \rangle$
| *Weak*: $\langle A \vdash_{@} (i, p) \implies (k, q) \# A \vdash_{@} (i, p) \rangle$

7.4 Soundness

theorem soundness: $\langle A \vdash_{@} (i, p) \implies \text{list-all } (\lambda(i, p). (M, g, g i) \models p) A \implies (M, g, g i) \models p \rangle$
 $\langle \text{proof} \rangle$

corollary soundness': $\langle [] \vdash_{@} (i, p) \implies i \notin \text{nominals-fm } p \implies (M, g, w) \models p \rangle$
 $\langle \text{proof} \rangle$

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

7.5 Admissible Rules

lemma Assm-head: $\langle (p, i) \# A \vdash_{@} (p, i) \rangle$
 $\langle \text{proof} \rangle$

lemma deduct1: $\langle A \vdash_{@} (i, p \longrightarrow q) \implies (i, p) \# A \vdash_{@} (i, q) \rangle$
 $\langle \text{proof} \rangle$

lemma Boole: $\langle (i, \neg p) \# A \vdash_{@} (i, \perp) \implies A \vdash_{@} (i, p) \rangle$
 $\langle \text{proof} \rangle$

lemma Weak': $\langle A \vdash_{@} (i, p) \implies B @ A \vdash_{@} (i, p) \rangle$
 $\langle \text{proof} \rangle$

lemma ImpI':
assumes $\langle (k, q) \# A \vdash_{@} (i, p) \rangle$

shows $\langle A \vdash_{@} (i, (@k q) \longrightarrow p) \rangle$
 $\langle \text{proof} \rangle$

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

interpretation *Derivations Calculus*
 $\langle \text{proof} \rangle$

7.6 Maximal Consistent Sets

definition *consistent* :: $\langle ('i, 'p) \text{ lbd set} \implies \text{bool} \rangle$ **where**
 $\langle \text{consistent } S \equiv \nexists S' a. \text{ set } S' \subseteq S \wedge S' \vdash_{@} (a, \perp) \rangle$

lemma *consistent-add-witness*:
assumes $\langle \text{consistent } S \rangle \langle (i, \diamond p) \in S \rangle \langle k \notin \text{nominals } S \rangle$
shows $\langle \text{consistent } (\{(k, p), (i, \diamond (\cdot k))\} \cup S) \rangle$
 $\langle \text{proof} \rangle$

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

lemma *consistent-witness'*:
assumes $\langle \text{consistent } (\{(i, p)\} \cup S) \rangle \langle \text{infinite } (\text{UNIV} - \text{nominals } S) \rangle$
shows $\langle \text{consistent } (\text{witness } (i, p) S \cup \{(i, p)\} \cup S) \rangle$
 $\langle \text{proof} \rangle$

interpretation *MCS-Saturation consistent nominals-lbd witness*
 $\langle \text{proof} \rangle$

interpretation *Derivations-MCS-Cut Calculus consistent* $\langle (\text{undefined}, \perp) \rangle$
 $\langle \text{proof} \rangle$

lemma *saturated*: $\langle \text{saturated } H \implies (i, \diamond p) \in H \implies \exists k. (i, \diamond (\cdot k)) \in H \wedge (k, p) \in H \rangle$
 $\langle \text{proof} \rangle$

7.7 Nominals

lemma *MCS-Nom-refl*:
assumes $\langle \text{consistent } S \rangle \langle \text{maximal } S \rangle$
shows $\langle (i, \cdot i) \in S \rangle$
 $\langle \text{proof} \rangle$

lemma *MCS-Nom-sym*:
assumes $\langle \text{consistent } S \rangle \langle \text{maximal } S \rangle \langle (i, \cdot k) \in S \rangle$
shows $\langle (k, \cdot i) \in S \rangle$
 $\langle \text{proof} \rangle$

lemma *MCS-Nom-trans*:
assumes $\langle \text{consistent } S \rangle \langle \text{maximal } S \rangle \langle (i, \cdot j) \in S \rangle \langle (j, \cdot k) \in S \rangle$
shows $\langle (i, \cdot k) \in S \rangle$
 $\langle \text{proof} \rangle$

7.8 Truth Lemma

fun *semics* :: $\langle ('i, 'p, 'w) \text{ ctx} \Rightarrow (('i, 'p, 'w) \text{ ctx} \Rightarrow ('i, 'p) \text{ fm} \Rightarrow \text{bool}) \Rightarrow ('i, 'p) \text{ fm} \Rightarrow \text{bool} \rangle$
where

$\langle \text{semics } - - \perp \longleftrightarrow \text{False} \rangle$
 $\langle \text{semics } (M, -, w) - (\ddagger P) \longleftrightarrow V M w P \rangle$
 $\langle \text{semics } (-, g, w) - (\cdot i) \longleftrightarrow w = g i \rangle$
 $\langle \text{semics } (M, g, w) \text{ rel } (p \longrightarrow q) \longleftrightarrow \text{rel } (M, g, w) p \longrightarrow \text{rel } (M, g, w) q \rangle$
 $\langle \text{semics } (M, g, w) \text{ rel } (\diamond p) \longleftrightarrow (\exists v \in R M w. \text{rel } (M, g, v) p) \rangle$
 $\langle \text{semics } (M, g, -) \text{ rel } (@ i p) \longleftrightarrow \text{rel } (M, g, g i) p \rangle$

fun *rel* :: $\langle ('i, 'p) \text{ lbd set} \Rightarrow ('i, 'p, 'i) \text{ ctx} \Rightarrow ('i, 'p) \text{ fm} \Rightarrow \text{bool} \rangle$ **where**
 $\langle \text{rel } H (-, -, i) p = ((i, p) \in H) \rangle$

definition *val* :: $\langle ('i, 'p) \text{ lbd set} \Rightarrow 'i \Rightarrow 'p \Rightarrow \text{bool} \rangle$ **where**
 $\langle \text{val } H i P \equiv (i, \ddagger P) \in H \rangle$

definition *hequiv* :: $\langle ('i, 'p) \text{ lbd set} \Rightarrow 'i \Rightarrow 'i \Rightarrow \text{bool} \rangle$ **where**
 $\langle \text{hequiv } H i k \equiv (i, \cdot k) \in H \rangle$

lemma *hequiv-reflp*:
assumes $\langle \text{consistent } H \rangle \langle \text{maximal } H \rangle$
shows $\langle \text{reflp } (\text{hequiv } H) \rangle$
 $\langle \text{proof} \rangle$

lemma *hequiv-symp*:
assumes $\langle \text{consistent } H \rangle \langle \text{maximal } H \rangle$
shows $\langle \text{symp } (\text{hequiv } H) \rangle$
 $\langle \text{proof} \rangle$

lemma *hequiv-transp*:
assumes $\langle \text{consistent } H \rangle \langle \text{maximal } H \rangle$
shows $\langle \text{transp } (\text{hequiv } H) \rangle$
 $\langle \text{proof} \rangle$

lemma *hequiv-equivp*:
assumes $\langle \text{consistent } H \rangle \langle \text{maximal } H \rangle$
shows $\langle \text{equivp } (\text{hequiv } H) \rangle$
 $\langle \text{proof} \rangle$

definition *assign* :: $\langle ('i, 'p) \text{ lbd set} \Rightarrow 'i \Rightarrow 'i \rangle$ **where**
 $\langle \text{assign } H i \equiv \text{minim } (|UNIV|) \{k. \text{hequiv } H i k\} \rangle$

lemma *hequiv-ne*:
assumes $\langle \text{consistent } H \rangle \langle \text{maximal } H \rangle$
shows $\langle \{k. \text{hequiv } H i k\} \neq \{\} \rangle$
 $\langle \text{proof} \rangle$

lemma *hequiv-assign*:
assumes $\langle \text{consistent } H \rangle \langle \text{maximal } H \rangle$
shows $\langle \text{hequiv } H i (\text{assign } H i) \rangle$
 $\langle \text{proof} \rangle$

lemma *hequiv-Nom*:
assumes $\langle \text{consistent } H \rangle \langle \text{maximal } H \rangle \langle \text{hequiv } H i k \rangle \langle (i, p) \in H \rangle$
shows $\langle (k, p) \in H \rangle$

⟨proof⟩

definition $reach :: \langle ('i, 'p) \text{ lbd set} \Rightarrow 'i \Rightarrow 'i \text{ set} \rangle$ **where**
⟨ $reach\ H\ i \equiv \{assign\ H\ k \mid k. (i, \diamond (\cdot k)) \in H\}$ ⟩

abbreviation $canonical :: \langle ('i \times ('i, 'p) \text{ fm}) \text{ set} \Rightarrow 'i \Rightarrow ('i, 'p, 'i) \text{ ctx} \rangle$ **where**
⟨ $canonical\ H\ i \equiv (Model\ (reach\ H)\ (val\ H),\ assign\ H,\ assign\ H\ i)$ ⟩

lemma *Hintikka-model'*:

assumes ⟨ $\bigwedge i\ p. \text{semics}\ (canonical\ H\ i)\ (rel\ H)\ p \longleftrightarrow rel\ H\ (canonical\ H\ i)\ p$ ⟩
shows ⟨ $(canonical\ H\ i \models p) \longleftrightarrow rel\ H\ (canonical\ H\ i)\ p$ ⟩

⟨proof⟩

lemma *Hintikka-Extend'*:

assumes ⟨ $consistent\ H$ ⟩ ⟨ $maximal\ H$ ⟩ ⟨ $saturated\ H$ ⟩
shows ⟨ $\text{semics}\ (canonical\ H\ i)\ (rel\ H)\ p \longleftrightarrow rel\ H\ (canonical\ H\ i)\ p$ ⟩

⟨proof⟩

interpretation *Truth-Saturation*

$consistent\ nominals\text{-lbd}\ \text{witness}\ \text{semics}\ \text{semantics} \langle \lambda H. \{canonical\ H\ i \mid i. True\} \rangle\ rel$

⟨proof⟩

lemma *Truth-lemma*:

assumes ⟨ $consistent\ H$ ⟩ ⟨ $maximal\ H$ ⟩ ⟨ $saturated\ H$ ⟩
shows ⟨ $(canonical\ H\ i \models p) \longleftrightarrow (i, p) \in H$ ⟩

⟨proof⟩

7.9 Cardinalities

datatype $marker = FlsM \mid ImpM \mid DiaM \mid SatM$

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

abbreviation ⟨ $NOM\ i \equiv Inl\ (Inl\ i)$ ⟩

abbreviation ⟨ $PRO\ x \equiv Inl\ (Inr\ x)$ ⟩

abbreviation ⟨ $FLS \equiv Inr\ (FlsM, 0)$ ⟩

abbreviation ⟨ $IMP\ n \equiv Inr\ (FlsM, n)$ ⟩

abbreviation ⟨ $DIA \equiv Inr\ (DiaM, 0)$ ⟩

abbreviation ⟨ $SAT \equiv Inr\ (SatM, 0)$ ⟩

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

⟨ $encode\ \perp = [FLS]$ ⟩

| ⟨ $encode\ (\ddagger P) = [PRO\ P]$ ⟩

| ⟨ $encode\ (\cdot i) = [NOM\ i]$ ⟩

| ⟨ $encode\ (p \longrightarrow q) = IMP\ (length\ (encode\ p)) \# encode\ p\ @\ encode\ q$ ⟩

| ⟨ $encode\ (\diamond p) = DIA\ \# encode\ p$ ⟩

| ⟨ $encode\ (@\ i\ p) = SAT\ \# NOM\ i\ \# encode\ p$ ⟩

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

⟨proof⟩

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

⟨proof⟩

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

⟨proof⟩

primrec *encode-lbd* :: ⟨('i, 'p) lbd ⇒ ('i, 'p) enc list⟩ **where**
⟨*encode-lbd* (i, p) = NOM i # encode p⟩

lemma *inj-encode-lbd'*: ⟨*encode-lbd* (i, p) = *encode-lbd* (k, q) ⇒ i = k ∧ p = q⟩
⟨proof⟩

lemma *inj-encode-lbd*: ⟨*inj encode-lbd*⟩
⟨proof⟩

lemma *finite-marker*: ⟨*finite* (UNIV :: marker set)⟩
⟨proof⟩

lemma *card-of-lbd*:
 assumes ⟨*infinite* (UNIV :: 'i set)⟩
 shows ⟨|UNIV :: ('i, 'p) lbd set| ≤*o* |UNIV :: 'i set| +*c* |UNIV :: 'p set|⟩
⟨proof⟩

7.10 Completeness

theorem *strong-completeness*:
 fixes *p* :: ⟨('i, 'p) fm⟩
 assumes ⟨∀ *M* :: ('i, 'p) model. ∀ *g w*.
 (∀ (k, q) ∈ *X*. (M, g, g k) ⊨ q) ⟶ (M, g, w) ⊨ *p*⟩
 ⟨*infinite* (UNIV :: 'i set)⟩
 ⟨|UNIV :: 'i set| +*c* |UNIV :: 'p set| ≤*o* |UNIV – nominals *X*|⟩
 shows ⟨∃ *A*. set *A* ⊆ *X* ∧ *A* ⊢_@ (i, p)⟩
⟨proof⟩

abbreviation *valid* :: ⟨('i, 'p) fm ⇒ bool⟩ **where**
⟨*valid* *p* ≡ ∀ (M :: ('i, 'p) model) *g w*. (M, g, w) ⊨ *p*⟩

theorem *completeness*:
 fixes *p* :: ⟨('i, 'p) fm⟩
 assumes ⟨*valid* *p*⟩ ⟨*infinite* (UNIV :: 'i set)⟩ ⟨|UNIV :: 'p set| ≤*o* |UNIV :: 'i set|⟩
 shows ⟨[] ⊢_@ (i, p)⟩
⟨proof⟩

corollary *completeness'*:
 fixes *p* :: ⟨('i, 'i) fm⟩
 assumes ⟨*valid* *p*⟩ ⟨*infinite* (UNIV :: 'i set)⟩
 shows ⟨[] ⊢_@ (i, p)⟩
⟨proof⟩

theorem *main*:
 fixes *p* :: ⟨('i, 'p) fm⟩
 assumes ⟨*i* ∉ *nominals-fm* *p*⟩ ⟨*infinite* (UNIV :: 'i set)⟩ ⟨|UNIV :: 'p set| ≤*o* |UNIV :: 'i set|⟩
 shows ⟨*valid* *p* ⟷ [] ⊢_@ (i, p)⟩
⟨proof⟩

corollary *main'*:
 fixes *p* :: ⟨('i, 'i) fm⟩
 assumes ⟨*i* ∉ *nominals-fm* *p*⟩ ⟨*infinite* (UNIV :: 'i set)⟩
 shows ⟨*valid* *p* ⟷ [] ⊢_@ (i, p)⟩

<proof>

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)
| *Uni* <'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 (<[-]>) **where**
| <[-] ⊥ ↔ *False*>
| <[(*E*, *F*, *G*)] (‡*P ts*) ↔ *G P* (map [(*E*, *F*)] *ts*)>
| <[(*E*, *F*, *G*)] (*p* —> *q*) ↔ [(*E*, *F*, *G*)] *p* —> [(*E*, *F*, *G*)] *q*>
| <[(*E*, *F*, *G*)] (∀*p*) ↔ (∀*x*. [(*x* § *E*, *F*, *G*)] *p*)>

8.3 Operations

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

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

primrec *sub-tm* :: $\langle \text{nat} \Rightarrow 'f \text{ tm} \Rightarrow 'f \text{ tm} \Rightarrow 'f \text{ tm} \rangle$ **where**
 $\langle \text{sub-tm } s (\#n) = s \ n \rangle$
 $\mid \langle \text{sub-tm } s (\dagger f \text{ ts}) = \dagger f (\text{map (sub-tm } s) \text{ ts}) \rangle$

primrec *sub-fm* :: $\langle \text{nat} \Rightarrow 'f \text{ tm} \Rightarrow ('f, 'p) \text{ fm} \Rightarrow ('f, 'p) \text{ fm} \rangle$ **where**
 $\langle \text{sub-fm } - \perp = \perp \rangle$
 $\mid \langle \text{sub-fm } s (\ddagger P \text{ ts}) = \ddagger P (\text{map (sub-tm } s) \text{ ts}) \rangle$
 $\mid \langle \text{sub-fm } s (p \longrightarrow q) = \text{sub-fm } s \ p \longrightarrow \text{sub-fm } s \ q \rangle$
 $\mid \langle \text{sub-fm } s (\forall p) = \forall (\text{sub-fm } (\#0 \circ \lambda n. \text{lift-tm } (s \ n)) \ p) \rangle$

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]*: $\langle f \notin \text{params-tm } t \Longrightarrow \llbracket (E, F(f := x)) \rrbracket t = \llbracket (E, F) \rrbracket t \rangle$
 $\langle \text{proof} \rangle$

lemma *upd-params-fm [simp]*: $\langle f \notin \text{params-fm } p \Longrightarrow \llbracket (E, F(f := x), G) \rrbracket p = \llbracket (E, F, G) \rrbracket p \rangle$
 $\langle \text{proof} \rangle$

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

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

lemma *env [simp]*: $\langle P ((x \circ E) \ n) = (P \ x \circ \lambda n. P (E \ n)) \ n \rangle$
 $\langle \text{proof} \rangle$

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

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

lemma *sub-fm-semantics [simp]*: $\langle \llbracket (E, F, G) \rrbracket (\text{sub-fm } s \ p) = \llbracket (\lambda n. \llbracket (E, F) \rrbracket (s \ n), F, G) \rrbracket p \rangle$
 $\langle \text{proof} \rangle$

lemma *sub-tm-Var*: $\langle \text{sub-tm } \# \ t = t \rangle$
 $\langle \text{proof} \rangle$

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

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

lemma *semantics-tm-id [simp]*: $\langle \llbracket (\#, \dagger) \rrbracket t = t \rangle$

⟨proof⟩

lemma *semantics-tm-id-map* [simp]: ⟨map ((#, †)) ts = ts⟩
⟨proof⟩

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

primrec *size-fm* :: ⟨('f, 'p) fm ⇒ nat⟩ **where**
| *size-fm* ⊥ = 1
| *size-fm* (‡-) = 1
| *size-fm* (p → q) = 1 + *size-fm* p + *size-fm* q
| *size-fm* (∀ p) = 1 + *size-fm* p

lemma *size-sub-fm* [simp]: ⟨*size-fm* (sub-fm s p) = *size-fm* p⟩
⟨proof⟩

8.4 Calculus

inductive *Calculus* :: ⟨('f, 'p) fm list ⇒ ('f, 'p) fm ⇒ bool⟩ (⟨- ⊢_∀ -⟩ [50, 50] 50) **where**
| *Assm* [intro]: ⟨p ∈ set A ⇒ A ⊢_∀ p⟩
| *FlsE* [elim]: ⟨A ⊢_∀ ⊥ ⇒ A ⊢_∀ p⟩
| *ImpI* [intro]: ⟨p # A ⊢_∀ q ⇒ A ⊢_∀ p → q⟩
| *ImpE* [elim]: ⟨A ⊢_∀ p → q ⇒ A ⊢_∀ p ⇒ A ⊢_∀ q⟩
| *UniI* [intro]: ⟨A ⊢_∀ (★a)p ⇒ a ∉ params (p # A) ⇒ A ⊢_∀ ∀ p⟩
| *UniE* [elim]: ⟨A ⊢_∀ ∀ p ⇒ A ⊢_∀ ⟨t⟩p⟩
| *Clas*: ⟨(p → q) # A ⊢_∀ p ⇒ A ⊢_∀ p⟩
| *Weak*: ⟨A ⊢_∀ p ⇒ q # A ⊢_∀ p⟩

8.5 Soundness

theorem *soundness*: ⟨A ⊢_∀ p ⇒ list-all [(E, F, G)] A ⇒ [(E, F, G)] p⟩
⟨proof⟩

corollary *soundness'*: ⟨[] ⊢_∀ p ⇒ [M] p⟩
⟨proof⟩

corollary ⟨¬ ([] ⊢_∀ ⊥)⟩
⟨proof⟩

8.6 Admissible Rules

lemma *Assm-head*: ⟨p # A ⊢_∀ p⟩
⟨proof⟩

lemma *deduct1*: ⟨A ⊢_∀ p → q ⇒ p # A ⊢_∀ q⟩
⟨proof⟩

lemma *Boole*: ⟨(¬ p) # A ⊢_∀ ⊥ ⇒ A ⊢_∀ p⟩
⟨proof⟩

lemma *Weak'*: ⟨A ⊢_∀ p ⇒ B @ A ⊢_∀ p⟩
⟨proof⟩

lemma *Weaken*: ⟨A ⊢_∀ p ⇒ set A ⊆ set B ⇒ B ⊢_∀ p⟩
⟨proof⟩

interpretation *Derivations Calculus*
 ⟨proof⟩

8.7 Maximal Consistent Sets

definition *consistent* :: ⟨('f, 'p) fm set ⇒ bool⟩ **where**
 ⟨consistent S ≡ $\nexists S'. \text{set } S' \subseteq S \wedge S' \vdash_{\forall} \perp$ ⟩

fun *witness* :: ⟨('f, 'p) fm ⇒ ('f, 'p) fm set ⇒ ('f, 'p) fm set⟩ **where**
 ⟨witness (¬ (∀ p)) S = {¬ (★(SOME a. a ∉ params' ({p} ∪ S)))p}⟩
 | ⟨witness - - = {}⟩

lemma *consistent-add-instance*:
assumes ⟨consistent S⟩ ⟨∀ p ∈ S⟩
shows ⟨consistent ({t} ∪ S)⟩
 ⟨proof⟩

lemma *consistent-add-witness*:
assumes ⟨consistent S⟩ ⟨¬ (∀ p) ∈ S⟩ ⟨a ∉ params' S⟩
shows ⟨consistent ({¬ (★a)p} ∪ S)⟩
 ⟨proof⟩

lemma *consistent-witness'*:
assumes ⟨consistent ({p} ∪ S)⟩ ⟨infinite (UNIV - params' S)⟩
shows ⟨consistent (witness p S ∪ {p} ∪ S)⟩
 ⟨proof⟩

interpretation *MCS-Saturation consistent params-fm witness*
 ⟨proof⟩

interpretation *Derivations-MCS-Cut Calculus consistent* ⟨⊥⟩
 ⟨proof⟩

8.8 Truth Lemma

abbreviation *hmodel* :: ⟨('f, 'p) fm set ⇒ ('f tm, 'f, 'p) model⟩ **where**
 ⟨hmodel H ≡ (♯, †, λP ts. †P ts ∈ H)⟩

fun *semics* ::
 ⟨('a, 'f, 'p) model ⇒ (('a, 'f, 'p) model ⇒ ('f, 'p) fm ⇒ bool) ⇒ ('f, 'p) fm ⇒ bool⟩ **where**
 ⟨semics - - ⊥ ↔ False⟩
 | ⟨semics (E, F, G) - (†P ts) ↔ G P (map ((E, F)) ts)⟩
 | ⟨semics (E, F, G) rel (p → q) ↔ rel (E, F, G) p → rel (E, F, G) q⟩
 | ⟨semics (E, F, G) rel (∀ p) ↔ (∀ x. rel (x ∘ E, F, G) p)⟩

fun *rel* :: ⟨('f, 'p) fm set ⇒ ('f tm, 'f, 'p) model ⇒ ('f, 'p) fm ⇒ bool⟩ **where**
 ⟨rel H (E, -, -) p = (sub-fm E p ∈ H)⟩

theorem *Hintikka-model'*:
assumes ⟨∧p. semics (hmodel H) (rel H) p ↔ p ∈ H⟩
shows ⟨p ∈ H ↔ ⟦hmodel H⟧ p⟩
 ⟨proof⟩

lemma *Hintikka-Extend*:

assumes $\langle \text{consistent } H \rangle \langle \text{maximal } H \rangle \langle \text{saturated } H \rangle$
shows $\langle \text{semics } (hmodel\ H) \ (rel\ H) \ p \longleftrightarrow p \in H \rangle$
 $\langle \text{proof} \rangle$

interpretation *Truth-Saturation*

consistent params-fm witness semics semantics-fm $\langle \lambda H. \{hmodel\ H\} \rangle \ rel$
 $\langle \text{proof} \rangle$

8.9 Cardinalities

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

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

abbreviation $\langle FUNS\ f \equiv Inl\ (Inl\ f) \rangle$

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

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

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

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

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

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

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

abbreviation $\langle UNI \equiv Inr\ (UniM, 0) \rangle$

primrec

encode-tm :: $\langle 'f\ tm \Rightarrow ('f, 'p)\ enc\ list \rangle$ **and**
encode-tms :: $\langle 'f\ tm\ list \Rightarrow ('f, 'p)\ enc\ list \rangle$ **where**
 $\langle \text{encode-tm } (\#n) = [VAR\ n] \rangle$
 $| \langle \text{encode-tm } (\dagger f\ ts) = FUN\ (length\ ts) \# FUNS\ f \# \text{encode-tms } ts \rangle$
 $| \langle \text{encode-tms } [] = [] \rangle$
 $| \langle \text{encode-tms } (t \# ts) = TM\ (length\ (\text{encode-tm } t)) \# \text{encode-tm } t \ @ \ \text{encode-tms } ts \rangle$

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

lemma *inj-encode-tm'*:

$\langle (\text{encode-tm } t :: ('f, 'p)\ enc\ list) = \text{encode-tm } s \Longrightarrow t = s \rangle$
 $\langle (\text{encode-tms } ts :: ('f, 'p)\ enc\ list) = \text{encode-tms } ss \Longrightarrow ts = ss \rangle$
 $\langle \text{proof} \rangle$

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

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

$\langle \text{encode-fm } \perp = [FLS] \rangle$
 $| \langle \text{encode-fm } (\dagger P\ ts) = PRE\ (length\ ts) \# PRES\ P \# \text{encode-tms } ts \rangle$
 $| \langle \text{encode-fm } (p \longrightarrow q) = IMP\ (length\ (\text{encode-fm } p)) \# \text{encode-fm } p \ @ \ \text{encode-fm } q \rangle$
 $| \langle \text{encode-fm } (\forall p) = UNI \# \text{encode-fm } p \rangle$

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

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

⟨proof⟩

lemma *inj-encode-fm*: ⟨inj encode-fm⟩

⟨proof⟩

lemma *finite-marker*: ⟨finite (UNIV :: marker set)⟩

⟨proof⟩

lemma *card-of-fm*:

assumes ⟨infinite (UNIV :: 'f set)⟩

shows ⟨|UNIV :: ('f, 'p) fm set| ≤ o |UNIV :: 'f set| + c |UNIV :: 'p set|⟩

⟨proof⟩

8.10 Completeness

theorem *strong-completeness*:

assumes ⟨∀ M :: ('f tm, 'f, 'p) model. (∀ q ∈ X. ⟦M⟧ q ⟶ ⟦M⟧ p)⟩

⟨infinite (UNIV :: 'f set)⟩

⟨|UNIV :: 'f set| + c |UNIV :: 'p set| ≤ o |UNIV - params' X|⟩

shows ⟨∃ A. set A ⊆ X ∧ A ⊢_∀ p⟩

⟨proof⟩

abbreviation *valid* :: ⟨('f, 'p) fm ⇒ bool⟩ **where**

⟨valid p ≡ ∀ M :: ('f tm, -, -) model. ⟦M⟧ p⟩

theorem *completeness*:

fixes p :: ⟨('f, 'p) fm⟩

assumes ⟨valid p⟩ ⟨infinite (UNIV :: 'f set)⟩ ⟨|UNIV :: 'p set| ≤ o |UNIV :: 'f set|⟩

shows ⟨[] ⊢_∀ p⟩

⟨proof⟩

corollary *completeness'*:

fixes p :: ⟨('f, 'f) fm⟩

assumes ⟨valid p⟩ ⟨infinite (UNIV :: 'f set)⟩

shows ⟨[] ⊢_∀ p⟩

⟨proof⟩

theorem *main*:

fixes p :: ⟨('f, 'p) fm⟩

assumes ⟨infinite (UNIV :: 'f set)⟩ ⟨|UNIV :: 'p set| ≤ o |UNIV :: 'f set|⟩

shows ⟨valid p ⟷ [] ⊢_∀ p⟩

⟨proof⟩

corollary *main'*:

fixes p :: ⟨('f, 'f) fm⟩

assumes ⟨infinite (UNIV :: 'f set)⟩

shows ⟨valid p ⟷ [] ⊢_∀ p⟩

⟨proof⟩

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.