Synthetic Completeness

Asta Halkjær From

January 10, 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

Chapter 1

Maximal Consistent Sets

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

1.1 Utility

lemma *Set-Diff-Un*: ‹$X - (Y \cup Z) = X - Y - Z$›
⟨*proof*⟩

lemma *infinite-Diff-fin-Un*: ‹*infinite* $(X - Y) \Longrightarrow$ *finite* $Z \Longrightarrow$ *infinite* $(X - (Z \cup Y))$›
⟨*proof*⟩

lemma *infinite-Diff-subset*: ‹*infinite* $(X - A) \Longrightarrow B \subseteq A \Longrightarrow$ *infinite* $(X - B)$›
⟨*proof*⟩

lemma *finite-bound*:
fixes X :: ‹$('a$ :: *size*$)$ *set*›
assumes ‹*finite* X› ‹$X \neq \{\}$›
shows ‹$\exists\, x \in X.\ \forall\, y \in X.$ *size* $y \leq$ *size* x›
⟨*proof*⟩

lemma *infinite-UNIV-size*:
fixes f :: ‹$('a$ :: *size*$) \Rightarrow 'a$›
assumes ‹$\bigwedge x.$ *size* $x <$ *size* $(f\, x)$›
shows ‹*infinite* $($*UNIV* :: $'a$ *set*$)$›
⟨*proof*⟩

lemma *split-finite-sets*:
assumes ‹*finite* A› ‹*finite* B›
assumes ‹$A \subseteq B \cup S$›
shows ‹$\exists\, B'\ C.$ *finite* $C \wedge (B' \cup C = A) \wedge B' \subseteq B \wedge C \subseteq S$›
⟨*proof*⟩

lemma *split-list*:
assumes ‹*set* $A \subseteq$ *set* $B \cup S$›
shows ‹$\exists\, B'\ C.$ *set* $(B' @ C) =$ *set* $A \wedge$ *set* $B' \subseteq$ *set* $B \wedge$ *set* $C \subseteq S$›

⟨*proof*⟩

lemma *struct-split*:
assumes ‹⋀*A B*. *P A* ⟹ *set A* ⊆ *set B* ⟹ *P B*› ‹*P A*› ‹*set A* ⊆ *set B* ∪ *X*›
shows ‹∃ *C*. *set C* ⊆ *X* ∧ *P* (*B* @ *C*)›
⟨*proof*⟩

context *wo-rel* **begin**

lemma *underS-bound*: ‹*a* ∈ *underS n* ⟹ *b* ∈ *underS n* ⟹ *a* ∈ *under b* ∨ *b* ∈ *under a*›
⟨*proof*⟩

lemma *finite-underS-bound*:
assumes ‹*finite X*› ‹*X* ⊆ *underS n*› ‹*X* ≠ {}›
shows ‹∃ *a* ∈ *X*. ∀ *b* ∈ *X*. *b* ∈ *under a*›
⟨*proof*⟩

lemma *finite-bound-under*:
assumes ‹*finite p*› ‹*p* ⊆ (⋃ *n* ∈ *Field r*. *f n*)›
shows ‹∃ *m*. *p* ⊆ (⋃ *n* ∈ *under m*. *f n*)›
⟨*proof*⟩

lemma *underS-trans*: ‹*a* ∈ *underS b* ⟹ *b* ∈ *underS c* ⟹ *a* ∈ *underS c*›
⟨*proof*⟩

end

lemma *card-of-infinite-smaller-Union*:
assumes ‹∀ *x*. |*f x*| <*o* |*X*|› ‹*infinite X*›
shows ‹|⋃ *x* ∈ *X*. *f x*| ≤*o* |*X*|›
⟨*proof*⟩

lemma *card-of-info-marker-lists*:
assumes ‹*infinite* (*UNIV* :: *'i set*)› ‹|*UNIV* :: *'m set*| ≤*o* |*UNIV* :: *nat set*|›
shows ‹|*UNIV* :: (*'i* + *'m* × *nat*) *list set*| ≤*o* |*UNIV* :: *'i set*|›
⟨*proof*⟩

1.2 Base Locale

locale *MCS-Base* =
fixes *consistent* :: ‹*'a set* ⇒ *bool*›
assumes *consistent-hereditary*: ‹⋀*S S'*. *consistent S* ⟹ *S'* ⊆ *S* ⟹ *consistent S'*›
and *inconsistent-finite*: ‹⋀*S*. ¬ *consistent S* ⟹ ∃ *S'* ⊆ *S*. *finite S'* ∧ ¬ *consistent S'*›
begin

definition *maximal* :: ‹*'a set* ⇒ *bool*› **where**
‹*maximal S* ≡ ∀ *p*. *consistent* ({*p*} ∪ *S*) ⟶ *p* ∈ *S*›

end

1.3 Ordinal Locale

locale *MCS-Lim-Ord* = *MCS-Base* +
  fixes r :: ‹$'a$ *rel*›
  assumes *WELL*: ‹*Well-order* r›
    and *Cinfinite-r*: ‹*Cinfinite* r›
  fixes *info* :: ‹$'a \Rightarrow 'i$ *set*›
    and *witness* :: ‹$'a \Rightarrow 'a$ *set* $\Rightarrow 'a$ *set*›
  assumes *finite-info*: ‹$\bigwedge p$. *finite* (*info* p)›
    and *finite-witness-info*: ‹$\bigwedge p\ S$. *finite* ($\bigcup$(*info* ‘ *witness* $p\ S$))›
    and *consistent-witness*: ‹$\bigwedge p\ S$. *consistent* ($\{p\} \cup S$)
      $\Longrightarrow$ *infinite* (*UNIV* $-$ $\bigcup$(*info* ‘ S))
      $\Longrightarrow$ *consistent* (*witness* $p\ S \cup \{p\} \cup S$)›
begin

lemma *wo-rel-r*: ‹*wo-rel* r›
  ⟨*proof*⟩

lemma *isLimOrd-r*: ‹*isLimOrd* r›
  ⟨*proof*⟩

1.3.1 Lindenbaum Extension

abbreviation *infos* :: ‹$'a$ *set* $\Rightarrow 'i$ *set*› **where**
  ‹*infos* $S \equiv \bigcup$(*info* ‘ S)›

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

definition *extendL* :: ‹($'a \Rightarrow 'a$ *set*) $\Rightarrow 'a \Rightarrow 'a$ *set*› **where**
  ‹*extendL rec* $n \equiv \bigcup m \in$ *underS* $r\ n$. *rec* m›

definition *extend* :: ‹$'a$ *set* $\Rightarrow 'a \Rightarrow 'a$ *set*› **where**
  ‹*extend* $S\ n \equiv$ *worecZSL* $r\ S$ *extendS extendL* n›

lemma *adm-woL-extendL*: ‹*adm-woL* r *extendL*›
  ⟨*proof*⟩

definition *Extend* :: ‹$'a$ *set* $\Rightarrow 'a$ *set*› **where**
  ‹*Extend* $S \equiv \bigcup n \in$ *Field* r. *extend* $S\ n$›

lemma *extend-subset*: ‹$n \in$ *Field* $r \Longrightarrow S \subseteq$ *extend* $S\ n$›
⟨*proof*⟩

lemma *Extend-subset'*: ‹*Field* $r \neq \{\} \Longrightarrow S \subseteq$ *Extend* S›
  ⟨*proof*⟩

lemma *extend-underS*: ‹$m \in$ *underS* $r\ n \Longrightarrow$ *extend* $S\ m \subseteq$ *extend* $S\ n$›
⟨*proof*⟩

lemma *extend-under*: ‹$m \in under\ r\ n \Longrightarrow extend\ S\ m \subseteq extend\ S\ n$›
⟨*proof*⟩

1.3.2 Consistency

lemma *info-origin*:
assumes ‹$a \in infos\ (extend\ S\ n)$›
shows ‹$a \in infos\ S \vee (\exists\, m \in underS\ r\ n.\ a \in infos\ (witness\ m\ (extend\ S\ m) \cup \{m\}))$›
⟨*proof*⟩

lemma *consistent-extend*:
assumes ‹*consistent S*› ‹$r \leq o\ |UNIV - infos\ S|$›
shows ‹*consistent* (*extend S n*)›
⟨*proof*⟩

lemma *consistent-Extend*:
assumes ‹*consistent S*› ‹$r \leq o\ |UNIV - infos\ S|$›
shows ‹*consistent* (*Extend S*)›
⟨*proof*⟩

lemma *Extend-bound*: ‹$n \in Field\ r \Longrightarrow extend\ S\ n \subseteq Extend\ S$›
⟨*proof*⟩

1.3.3 Maximality

definition *maximal'* :: ‹'*a set* ⇒ *bool*› **where**
‹$maximal'\ S \equiv \forall\, p \in Field\ r.\ consistent\ (\{p\} \cup S) \longrightarrow p \in S$›

lemma *maximal'-Extend*: ‹*maximal'* (*Extend S*)›
⟨*proof*⟩

1.3.4 Saturation

definition *saturated'* :: ‹'*a set* ⇒ *bool*› **where**
‹$saturated'\ S \equiv \forall\, p \in S.\ p \in Field\ r \longrightarrow (\exists\, S'.\ witness\ p\ S' \subseteq S)$›

lemma *saturated'-Extend*:
assumes ‹*consistent* (*Extend S*)›
shows ‹*saturated'* (*Extend S*)›
⟨*proof*⟩

end

1.4 Locale with Saturation

locale *MCS-Saturation* = *MCS-Base* +
assumes *infinite-UNIV*: ‹*infinite* (*UNIV* :: '*a set*)›
fixes *info* :: ‹'*a* ⇒ '*i set*›
and *witness* :: ‹'*a* ⇒ '*a set* ⇒ '*a set*›

assumes ‹$\bigwedge p$. *finite* (*info p*)›
and ‹$\bigwedge p\ S$. *finite* ($\bigcup$(*info* ‘ *witness p S*))›
and ‹$\bigwedge p\ S$. *consistent* ($\{p\} \cup S$) $\Longrightarrow$ *infinite* (*UNIV* − $\bigcup$(*info* ‘ *S*)) $\Longrightarrow$ *consistent* (*witness p S* ∪ $\{p\}$ ∪ *S*)›

sublocale *MCS-Saturation* ⊆ *MCS-Lim-Ord* - ‹|*UNIV*|›
⟨*proof*⟩

context *MCS-Saturation* **begin**

theorem *Extend-subset*: ‹*S* ⊆ *Extend S*›
⟨*proof*⟩

lemma *maximal-maximal'*: ‹*maximal S* ⟷ *maximal' S*›
⟨*proof*⟩

lemma *maximal-Extend*: ‹*maximal* (*Extend S*)›
⟨*proof*⟩

definition *saturated* :: ‹'*a set* ⇒ *bool*› **where**
‹*saturated S* ≡ ∀ *p* ∈ *S*. ∃ *S'*. *witness p S'* ⊆ *S*›

lemma *saturated-saturated'*: ‹*saturated S* ⟷ *saturated' S*›
⟨*proof*⟩

lemma *saturated-Extend*:
assumes ‹*consistent* (*Extend S*)›
shows ‹*saturated* (*Extend S*)›
⟨*proof*⟩

theorem *MCS-Extend*:
assumes ‹*consistent S*› ‹|*UNIV* :: '*a set*| ≤*o* |*UNIV* − *infos 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 *interp* :: ‹'*model* ⇒ ('*model* ⇒ '*fm* ⇒ *bool*) ⇒ '*fm* ⇒ *bool*›
and *semantics* :: ‹'*model* ⇒ '*fm* ⇒ *bool*›
and *models-from* :: ‹'*a set* ⇒ '*model set*›
and *P* :: ‹'*a set* ⇒ '*model* ⇒ '*fm* ⇒ *bool*›
assumes *interp-semantics*: ‹*semantics M p* ⟷ *interp M semantics p*›
and *Hintikka-model*: ‹⋀*H M p*. ∀ *M* ∈ *models-from H*. ∀ *p*. *interp M* (*P H*) *p* ⟷ *P H M p* ⟹
M ∈ *models-from H* ⟹ *semantics M p* ⟷ *P 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*. *interp M* (*P H*) *p* ⟷ *P H M p*›
begin

theorem *truth-lemma-saturation*:
assumes ‹*consistent H*› ‹*maximal H*› ‹*saturated H*› ‹*M* ∈ *models-from H*›
shows ‹*semantics M p* ⟷ *P H M p*›
⟨*proof*⟩

end

locale *Truth-No-Saturation* = *MCS-No-Saturation* + *Truth-Base* +
assumes *MCS-Hintikka*: ‹⋀*H*. *consistent H* ⟹ *maximal H* ⟹
∀ *M* ∈ *models-from H*. ∀ *p*. *interp M* (*P H*) *p* ⟷ *P H M p*›
begin

theorem *truth-lemma-no-saturation*:
assumes ‹*consistent H*› ‹*maximal H*› ‹*M* ∈ *models-from H*›
shows ‹*semantics M p* ⟷ *P H M p*›
⟨*proof*⟩

end

end

Chapter 2

Derivations

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

2.1 Rearranging Assumptions

locale *Derivations* =
  fixes *derive* :: ‹$'a$ *list* ⇒ $'a$ ⇒ *bool*›
  assumes *derive-struct*: ‹$\bigwedge A\ B\ p$. *derive* $A\ p \Longrightarrow$ *set* $A \subseteq$ *set* $B \Longrightarrow$ *derive* $B\ p$›
begin

theorem *derive-split*:
  assumes ‹*set* $A \subseteq$ *set* $B \cup X$› ‹*derive* $A\ p$›
  shows ‹$\exists\, C$. *set* $C \subseteq X \wedge$ *derive* $(B\ @\ C)\ p$›
  ⟨*proof*⟩

corollary *derive-split1*:
  assumes ‹*set* $A \subseteq \{q\} \cup X$› ‹*derive* $A\ p$›
  shows ‹$\exists\, C$. *set* $C \subseteq X \wedge$ *derive* $(q\ \#\ C)\ p$›
  ⟨*proof*⟩

end

2.2 MCSs and Deriving Falsity

locale *Derivations-MCS* = *Derivations* + *MCS-Base* +
  fixes *fls*
  assumes *consistent-derive-fls*: ‹$\bigwedge S$. *consistent* $S = (\nexists\, S'$. *set* $S' \subseteq S \wedge$ *derive* S' *fls*)›
begin

theorem *MCS-derive-fls*:
  assumes ‹*consistent* S› ‹*maximal* S›
  shows ‹$p \notin S \longleftrightarrow (\exists\, S'$. *set* $S' \subseteq S \wedge$ *derive* $(p\ \#\ S')$ *fls*)›
⟨*proof*⟩

end

2.3 MCSs and Derivability

locale *Derivations-MCS-Cut* = *Derivations-MCS* +
  assumes *derive-assm*: ‹$\bigwedge A\ p.\ p \in set\ A \Longrightarrow derive\ A\ p$›
    and *derive-cut*: ‹$\bigwedge A\ B\ p\ q.\ derive\ A\ p \Longrightarrow derive\ (p\ \#\ B)\ q \Longrightarrow derive\ (A\ @\ B)\ q$›
begin

theorem *MCS-derive*:
  assumes ‹*consistent S*› ‹*maximal S*›
  shows ‹$p \in S \longleftrightarrow (\exists\, S'.\ set\ S' \subseteq S \wedge derive\ S'\ p)$›
⟨*proof*⟩

end

end

Chapter 3

Refutations

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

3.1 Rearranging Refutations

locale *Refutations* =
  fixes *refute* :: ‹'*a list* ⇒ *bool*›
  assumes *refute-struct*: ‹⋀*A B*. *refute A* ⟹ *set A* ⊆ *set B* ⟹ *refute B*›
begin

theorem *refute-split*:
  assumes ‹*set A* ⊆ *set B* ∪ *X*› ‹*refute A*›
  shows ‹∃ *C*. *set C* ⊆ *X* ∧ *refute* (*B* @ *C*)›
  ⟨*proof*⟩

corollary *refute-split1*:
  assumes ‹*set A* ⊆ {*q*} ∪ *X*› ‹*refute A*›
  shows ‹∃ *C*. *set C* ⊆ *X* ∧ *refute* (*q* # *C*)›
  ⟨*proof*⟩

end

3.2 MCSs and Refutability

locale *Refutations-MCS* = *Refutations* + *MCS-Base* +
  assumes *consistent-refute*: ‹⋀*S*. *consistent S* = (∄ *S'*. *set S'* ⊆ *S* ∧ *refute S'*)›
begin

theorem *MCS-refute*:
  assumes ‹*consistent S*› ‹*maximal S*›
  shows ‹*p* ∉ *S* ⟷ (∃ *S'*. *set S'* ⊆ *S* ∧ *refute* (*p* # *S'*))›
⟨*proof*⟩

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*: ‹$\vdash_T A \Longrightarrow \exists\, p \in set\ A.\ \neg\ [\![I]\!]\ p$›
⟨*proof*⟩

corollary *soundness′*: ‹$\vdash_T [\neg\ p] \Longrightarrow [\![I]\!]\ p$›
⟨*proof*⟩

corollary ‹$\neg\ (\vdash_T [])$›
⟨*proof*⟩

4.5 Maximal Consistent Sets

definition *consistent* :: ‹$'p\ fm\ set \Rightarrow bool$› **where**
‹$consistent\ S \equiv \nexists\, S'.\ set\ S' \subseteq S \wedge \vdash_T S'$›

interpretation *MCS-No-Saturation consistent*
⟨*proof*⟩

interpretation *Refutations-MCS Calculus consistent*
⟨*proof*⟩

4.6 Truth Lemma

abbreviation (*input*) *hmodel* :: ‹$'p\ fm\ set \Rightarrow\ 'p\ model$› **where**
‹$hmodel\ H \equiv \lambda P.\ \ddagger P \in H$›

locale *Hintikka* =
fixes H :: ‹$'a\ fm\ set$›
assumes *AxiomH*: ‹$\bigwedge P.\ \ddagger P \in H \Longrightarrow \neg\ \ddagger P \in H \Longrightarrow False$›
and *NegIH*: ‹$\bigwedge p.\ \neg\ \neg\ p \in H \Longrightarrow p \in H$›
and *ImpPH*: ‹$\bigwedge p\ q.\ p \longrightarrow q \in H \Longrightarrow \neg\ p \in H \vee q \in H$›
and *ImpNH*: ‹$\bigwedge p\ q.\ \neg\ (p \longrightarrow q) \in H \Longrightarrow p \in H \wedge \neg\ q \in H$›

lemma *Hintikka-model*:
assumes ‹*Hintikka H*›
shows ‹$(p \in H \longrightarrow [\![hmodel\ H]\!]\ p) \wedge (\neg\ p \in H \longrightarrow \neg\ [\![hmodel\ H]\!]\ p)$›
⟨*proof*⟩

lemma *MCS-Hintikka*:
assumes ‹*consistent H*› ‹*maximal H*›
shows ‹*Hintikka H*›
⟨*proof*⟩

lemma *truth-lemma*:
assumes ‹*consistent H*› ‹*maximal H*› ‹$p \in H$›

shows ‹$[\![hmodel\ H]\!]\ p$›
⟨*proof*⟩

4.7 Completeness

theorem *strong-completeness*:
assumes ‹$\forall\, M :: 'p\ model.\ (\forall\, q \in X.\ [\![M]\!]\ q) \longrightarrow [\![M]\!]\ p$›
shows ‹$\exists\, A.\ set\ A \subseteq X \wedge \vdash_T \neg\ p\ \#\ A$›
⟨*proof*⟩

abbreviation *valid* :: ‹$'p\ fm \Rightarrow bool$› **where**
‹$valid\ p \equiv \forall\, M.\ [\![M]\!]\ p$›

theorem *completeness*:
assumes ‹*valid p*›
shows ‹$\vdash_T [\neg\ p]$›
⟨*proof*⟩

theorem *main*: ‹$valid\ p \longleftrightarrow \vdash_T [\neg\ p]$›
⟨*proof*⟩

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** ‹¬ *p* ≡ *p* ⟶ ⊥›

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*: ‹$A \vdash_S B \Longrightarrow set\ B \subseteq set\ B' \Longrightarrow A \vdash_S B'$›

lemma *Boole*: ‹$\neg\ p\ \#\ A \vdash_S [] \Longrightarrow A \vdash_S [p]$›
⟨*proof*⟩

5.4 Soundness

theorem *soundness*: ‹$A \vdash_S B \Longrightarrow \forall\, q \in set\ A.\ [\![I]\!]\ q \Longrightarrow \exists\, p \in set\ B.\ [\![I]\!]\ p$›
⟨*proof*⟩

corollary *soundness'*: ‹$[] \vdash_S [p] \Longrightarrow [\![I]\!]\ p$›
⟨*proof*⟩

corollary ‹$\neg\ ([] \vdash_S [])$›
⟨*proof*⟩

5.5 Maximal Consistent Sets

definition *consistent* :: ‹$'p\ fm\ set \Rightarrow bool$› **where**
‹$consistent\ S \equiv \nexists S'.\ set\ S' \subseteq S \wedge S' \vdash_S [\bot]$›

interpretation *MCS-No-Saturation consistent*
⟨*proof*⟩

interpretation *Derivations-MCS-Cut* ‹$\lambda A\ p.\ A \vdash_S [p]$› *consistent* ‹$\bot$›
⟨*proof*⟩

5.6 Truth Lemma

abbreviation *hmodel* :: ‹$'p\ fm\ set \Rightarrow 'p\ model$› **where**
‹$hmodel\ H \equiv \lambda P.\ \ddagger P \in H$›

fun *interp* :: ‹$'p\ model \Rightarrow ('p\ model \Rightarrow 'p\ fm \Rightarrow bool) \Rightarrow 'p\ fm \Rightarrow bool$› **where**
‹$interp\ -\ -\ \bot = False$›
| ‹$interp\ I\ -\ (\ddagger P) = I\ P$›
| ‹$interp\ I\ X\ (p \longrightarrow q) = (X\ I\ p \longrightarrow X\ I\ q)$›

fun *rel* :: ‹$'p\ fm\ set \Rightarrow 'p\ model \Rightarrow 'p\ fm \Rightarrow bool$› **where**
‹$rel\ H\ -\ p = (p \in H)$›

theorem *Hintikka-model'*:
assumes ‹$\bigwedge p.\ interp\ (hmodel\ H)\ (rel\ H)\ p \longleftrightarrow p \in H$›
shows ‹$p \in H \longleftrightarrow [\![hmodel\ H]\!]\ p$›
⟨*proof*⟩

lemma *Hintikka-Extend*:
assumes ‹$consistent\ H$› ‹$maximal\ H$›
shows ‹$interp\ (hmodel\ H)\ (rel\ H)\ p \longleftrightarrow p \in H$›

⟨*proof*⟩

interpretation *Truth-No-Saturation consistent interp semantics* ‹$\lambda H.\ \{hmodel\ H\}$› *rel*
⟨*proof*⟩

5.7 Completeness

theorem *strong-completeness*:
  assumes ‹$\forall\, M :: 'p\ model.\ (\forall\, q \in X.\ [\![M]\!]\ q) \longrightarrow [\![M]\!]\ p$›
  shows ‹$\exists\, A.\ set\ A \subseteq X \wedge A \vdash_S [p]$›
⟨*proof*⟩

abbreviation *valid* :: ‹$'p\ fm \Rightarrow bool$› **where**
  ‹$valid\ p \equiv \forall\, M.\ [\![M]\!]\ p$›

theorem *completeness*:
  assumes ‹*valid p*›
  shows ‹$[] \vdash_S [p]$›
  ⟨*proof*⟩

theorem *main*: ‹$valid\ p \longleftrightarrow [] \vdash_S [p]$›
  ⟨*proof*⟩

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**
  ‹*Neg p* ≡ *p* ⟶ ⊥›

6.2 Semantics

record $('i, 'p, 'w)$ *kripke* =
  $\mathcal{W}$ :: ‹$'w$ *set*›
  $\mathcal{K}$ :: ‹$'i$ ⇒ $'w$ ⇒ $'w$ *set*›
  π :: ‹$'w$ ⇒ $'p$ ⇒ *bool*›

type-synonym $('i, 'p, 'w)$ *ctx* = ‹$('i, 'p, 'w)$ *kripke* × $'w$›

fun *semantics* :: ‹$('i, 'p, 'w)$ *ctx* ⇒ $('i, 'p)$ *fm* ⇒ *bool*› (‹- ⊨ -› [*50*, *50*] *50*) **where**
  ‹- ⊨ ⊥ ⟷ *False*›
| ‹(M, w) ⊨ *Pro x* ⟷ π *M w x*›
| ‹(M, w) ⊨ *p* ⟶ *q* ⟷ (M, w) ⊨ *p* ⟶ (M, w) ⊨ *q*›
| ‹(M, w) ⊨ □ *i p* ⟷ (∀ *v* ∈ $\mathcal{W}$ *M* ∩ $\mathcal{K}$ *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* - (*Pro x*) = *g x*›

| ‹*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 *SystemK* :: ‹('*i*, '*p*) *fm* ⇒ *bool*› (‹⊢□ -› [*50*] *50*) **where**
A1: ‹*tautology p* ⟹ ⊢□ *p*›
| *A2*: ‹⊢□ □ *i* (*p* ⟶ *q*) ⟶ □ *i p* ⟶ □ *i q*›
| *R1*: ‹⊢□ *p* ⟹ ⊢□ *p* ⟶ *q* ⟹ ⊢□ *q*›
| *R2*: ‹⊢□ *p* ⟹ ⊢□ □ *i p*›

primrec *imply* :: ‹('*i*, '*p*) *fm list* ⇒ ('*i*, '*p*) *fm* ⇒ ('*i*, '*p*) *fm*› (**infixr** ‹⇝› *56*) **where**
‹([] ⇝ *q*) = *q*›
| ‹(*p* # *ps* ⇝ *q*) = (*p* ⟶ *ps* ⇝ *q*)›

abbreviation *K-assms* (‹- ⊢□ -› [*50*, *50*] *50*) **where**
‹*G* ⊢□ *p* ≡ ⊢□ *G* ⇝ *p*›

6.4 Soundness

lemma *eval-semantics*:
‹*eval* (*pi w*) (λ*q*. (⦇𝒲 = *W*, 𝒦 = *r*, π = *pi*⦈, *w*) ⊨ *q*) *p* = ((⦇𝒲 = *W*, 𝒦 = *r*, π = *pi*⦈, *w*) ⊨ *p*)›
⟨*proof*⟩

lemma *tautology*:
assumes ‹*tautology p*›
shows ‹(*M*, *w*) ⊨ *p*›
⟨*proof*⟩

theorem *soundness*:
assumes ‹⋀*M w p*. *A p* ⟹ *w* ∈ 𝒲 *M* ⟹ (*M*, *w*) ⊨ *p*›
shows ‹⊢□ *p* ⟹ *w* ∈ 𝒲 *M* ⟹ (*M*, *w*) ⊨ *p*›
⟨*proof*⟩

6.5 Derived rules

lemma *K-imply-head*: ‹*p* # *ps* ⊢□ *p*›
⟨*proof*⟩

lemma *K-imply-Cons*:
assumes ‹*ps* ⊢□ *q*›
shows ‹*p* # *ps* ⊢□ *q*›
⟨*proof*⟩

lemma *K-right-mp*:
assumes ‹*ps* ⊢□ *p*› ‹*ps* ⊢□ *p* ⟶ *q*›
shows ‹*ps* ⊢□ *q*›
⟨*proof*⟩

lemma *deduct1*: ‹$ps \vdash_\Box p \longrightarrow q \Longrightarrow p \# ps \vdash_\Box q$›
⟨*proof*⟩

lemma *imply-append* [*iff*]: ‹$(ps @ qs \rightsquigarrow r) = (ps \rightsquigarrow qs \rightsquigarrow r)$›
⟨*proof*⟩

lemma *imply-swap-append*: ‹$ps @ qs \vdash_\Box r \Longrightarrow qs @ ps \vdash_\Box r$›
⟨*proof*⟩

lemma *K-ImpI*: ‹$p \# ps \vdash_\Box q \Longrightarrow ps \vdash_\Box p \longrightarrow q$›
⟨*proof*⟩

lemma *imply-mem* [*simp*]: ‹$p \in set\ ps \Longrightarrow ps \vdash_\Box p$›
⟨*proof*⟩

lemma *add-imply* [*simp*]: ‹$\vdash_\Box q \Longrightarrow ps \vdash_\Box q$›
⟨*proof*⟩

lemma *K-imply-weaken*: ‹$ps \vdash_\Box q \Longrightarrow set\ ps \subseteq set\ ps' \Longrightarrow ps' \vdash_\Box q$›
⟨*proof*⟩

lemma *K-Boole*:
assumes ‹$(\neg p) \# G \vdash_\Box \bot$›
shows ‹$G \vdash_\Box p$›
⟨*proof*⟩

lemma *K-distrib-K-imp*:
assumes ‹$\vdash_\Box \Box i\ (G \rightsquigarrow q)$›
shows ‹$map\ (\Box i)\ G \vdash_\Box \Box i\ q$›
⟨*proof*⟩

interpretation *Derivations K-assms*
⟨*proof*⟩

6.6 Maximal Consistent Sets

definition *consistent* :: ‹$('i, 'p)\ fm\ set \Rightarrow bool$› **where**
‹$consistent\ S \equiv \nexists S'.\ set\ S' \subseteq S \wedge S' \vdash_\Box \bot$›

interpretation *MCS-No-Saturation consistent*
⟨*proof*⟩

interpretation *Derivations-MCS-Cut K-assms consistent* ‹$\bot$›
⟨*proof*⟩

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

⟨*proof*⟩

lemma *MCS-consequent*:
assumes ‹*consistent V*› ‹*maximal V*› ‹$(p \longrightarrow q) \in V$› ‹$p \in V$›
shows ‹$q \in V$›
⟨*proof*⟩

theorem *deriv-in-maximal*:
assumes ‹*consistent V*› ‹*maximal V*› ‹$\vdash_{\Box} p$›
shows ‹$p \in V$›
⟨*proof*⟩

theorem *exactly-one-in-maximal*:
assumes ‹*consistent V*› ‹*maximal V*›
shows ‹$p \in V \longleftrightarrow (\neg\ p) \notin V$›
⟨*proof*⟩

6.7 Truth Lemma

abbreviation *pi* :: ‹('*i*, '*p*) *fm set* ⇒ '*p* ⇒ *bool*› **where**
‹*pi V x* ≡ *Pro x* ∈ *V*›

abbreviation *known* :: ‹('*i*, '*p*) *fm set* ⇒ '*i* ⇒ ('*i*, '*p*) *fm set*› **where**
‹*known V i* ≡ {*p*. □ *i p* ∈ *V*}›

abbreviation *reach* :: ‹'*i* ⇒ ('*i*, '*p*) *fm set* ⇒ ('*i*, '*p*) *fm set set*› **where**
‹*reach i V* ≡ {*W*. *known V i* ⊆ *W*}›

abbreviation *mcss* :: ‹('*i*, '*p*) *fm set set*› **where**
‹*mcss* ≡ {*W*. *consistent W* ∧ *maximal W*}›

abbreviation *canonical* :: ‹('*i*, '*p*, ('*i*, '*p*) *fm set*) *kripke*› **where**
‹*canonical* ≡ (|$\mathcal{W}$ = *mcss*, $\mathcal{K}$ = *reach*, π = *pi*|)›

fun *interp* ::
‹('*i*, '*p*, '*w*) *ctx* ⇒ (('*i*, '*p*, '*w*) *ctx* ⇒ ('*i*, '*p*) *fm* ⇒ *bool*) ⇒ ('*i*, '*p*) *fm* ⇒ *bool*› **where**
‹*interp* - - ⊥ = *False*›
| ‹*interp* (*M*, *w*) - (*Pro x*) = π *M w x*›
| ‹*interp* (*M*, *w*) *X* (*p* ⟶ *q*) = (*X* (*M*, *w*) *p* ⟶ *X* (*M*, *w*) *q*)›
| ‹*interp* (*M*, *w*) *X* (□ *i p*) = (∀ *v* ∈ $\mathcal{W}$ *M* ∩ $\mathcal{K}$ *M i w*. *X* (*M*, *v*) *p*)›

fun *rel* :: ‹('*i*, '*p*) *fm set* ⇒ ('*i*, '*p*, ('*i*, '*p*) *fm set*) *ctx* ⇒ ('*i*, '*p*) *fm* ⇒ *bool*› **where**
‹*rel* - (-, *w*) *p* = (*p* ∈ *w*)›

lemma *Hintikka-model'*:
fixes *V* :: ‹('*i*, '*p*) *fm set*›
assumes ‹⋀(*V* :: ('*i*, '*p*) *fm set*) *p*. *V* ∈ *mcss* ⟹ *interp* (*canonical*, *V*) (*rel H*) *p* ⟷ *p* ∈ *V*›
shows ‹*V* ∈ *mcss* ⟹ (*canonical*, *V*) ⊨ *p* ⟷ *p* ∈ *V*›
⟨*proof*⟩

lemma *maximal-extension*:
fixes $V :: \langle('i, 'p)\ fm\ set\rangle$
assumes $\langle consistent\ V\rangle$
shows $\langle\exists\ W.\ V \subseteq W \wedge consistent\ W \wedge maximal\ W\rangle$
⟨*proof*⟩

lemma *Hintikka-canonical*:
fixes $V :: \langle('i, 'p)\ fm\ set\rangle$
assumes $\langle V \in mcss\rangle$
shows $\langle interp\ (canonical,\ V)\ (rel\ H)\ p = rel\ H\ (canonical,\ V)\ p\rangle$
⟨*proof*⟩

interpretation *Truth-No-Saturation consistent interp semantics*
$\langle\lambda\text{-}.\ \{(canonical,\ V)\ |\ V.\ V \in mcss\}\rangle$ *rel*
⟨*proof*⟩

lemma *Truth-lemma*:
fixes $p :: \langle('i, 'p)\ fm\rangle$
assumes $\langle consistent\ V\rangle$ $\langle maximal\ V\rangle$
shows $\langle(canonical,\ V) \models p \longleftrightarrow p \in V\rangle$
⟨*proof*⟩

lemma *canonical-model*:
assumes $\langle consistent\ S\rangle$ $\langle p \in S\rangle$
defines $\langle V \equiv Extend\ S\rangle$ **and** $\langle M \equiv canonical\rangle$
shows $\langle(M,\ V) \models p\rangle$ $\langle consistent\ V\rangle$ $\langle maximal\ V\rangle$
⟨*proof*⟩

6.8 Completeness

abbreviation *valid* :: $\langle('i, 'p)\ fm\ set \Rightarrow ('i, 'p)\ fm \Rightarrow bool\rangle$
($\langle\text{-} \Vdash \text{-}\rangle$ [50, 50] 50) **where**
$\langle G \Vdash p \equiv \forall\ M :: ('i, 'p, ('i, 'p)\ fm\ set)\ kripke.$
$(\forall\ w \in \mathcal{W}\ M.\ (\forall\ q \in G.\ (M,\ w) \models q) \longrightarrow (M,\ w) \models p)\rangle$

theorem *strong-completeness*:
assumes $\langle G \Vdash p\rangle$
shows $\langle\exists\ qs.\ set\ qs \subseteq G \wedge qs \vdash_{\Box} p\rangle$
⟨*proof*⟩

corollary *completeness*: $\langle\{\} \Vdash p \Longrightarrow \vdash_{\Box} p\rangle$
⟨*proof*⟩

theorem *main*: $\langle\{\} \Vdash p \longleftrightarrow \vdash_{\Box} p\rangle$
⟨*proof*⟩

end

Chapter 7

Example: Hybrid Logic

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

7.1 Syntax

datatype (*nominals-fm*: *'i*, *'p*) *fm*
= *Fls* (‹⊥›)
| *Pro 'p*
| *Nom 'i*
| *Imp* ‹(*'i*, *'p*) *fm*› ‹(*'i*, *'p*) *fm*› (**infixr** ‹⟶› *55*)
| *Dia* ‹(*'i*, *'p*) *fm*› (‹◇›)
| *Sat 'i* ‹(*'i*, *'p*) *fm*› (‹@›)

abbreviation *Neg* (‹¬ -› [*70*] *70*) **where** ‹¬ *p* ≡ *p* ⟶ ⊥›

type-synonym (*'i*, *'p*) *lbd* = ‹*'i* × (*'i*, *'p*) *fm*›

primrec *nominals-lbd* :: ‹(*'i*, *'p*) *lbd* ⇒ *'i set*› **where**
‹*nominals-lbd* (*i*, *p*) = {*i*} ∪ *nominals-fm p*›

abbreviation *nominals* :: ‹(*'i*, *'p*) *lbd set* ⇒ *'i set*› **where**
‹*nominals S* ≡ ⋃ *ip* ∈ *S*. *nominals-lbd ip*›

lemma *finite-nominals-fm*: ‹*finite* (*nominals-fm p*)›
⟨*proof*⟩

lemma *finite-nominals-lbd*: ‹*finite* (*nominals-lbd p*)›
⟨*proof*⟩

7.2 Semantics

datatype (*'w*, *'p*) *model* =
Model (*R*: ‹*'w* ⇒ *'w set*›) (*V*: ‹*'w* ⇒ *'p* ⇒ *bool*›)

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

fun *semantics* :: $\langle('i, 'p, 'w)\ ctx \Rightarrow ('i, 'p)\ fm \Rightarrow bool\rangle$ $(\langle\text{-} \models \text{-}\rangle\ [50, 50]\ 50)$ **where**
$\langle(M, g, w) \models \bot \longleftrightarrow False\rangle$
| $\langle(M, \text{-}, w) \models Pro\ x \longleftrightarrow V\ M\ w\ x\rangle$
| $\langle(\text{-}, g, w) \models Nom\ i \longleftrightarrow w = g\ i\rangle$
| $\langle(M, g, w) \models (p \longrightarrow q) \longleftrightarrow ((M, g, w) \models p \longrightarrow (M, g, w) \models q)\rangle$
| $\langle(M, g, w) \models \Diamond\ p \longleftrightarrow (\exists v \in R\ M\ w.\ (M, g, v) \models p)\rangle$
| $\langle(M, g, \text{-}) \models @i\ p \longleftrightarrow ((M, g, g\ i) \models p)\rangle$

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

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

7.3 Calculus

abbreviation *list-member-syntax* :: $\langle'a \Rightarrow 'a\ list \Rightarrow bool\rangle$ $(\langle\text{-}\ [\in]\ \text{-}\rangle\ [51, 51]\ 50)$ **where**
$\langle x\ [\in]\ A \equiv x \in set\ A\rangle$

inductive *Calculus* :: $\langle('i, 'p)\ lbd\ list \Rightarrow ('i, 'p)\ lbd \Rightarrow bool\rangle$ $(\langle\text{-} \vdash_@ \text{-}\rangle\ [50, 50]\ 50)$ **where**
Assm [*intro*]: $\langle(i, p)\ [\in]\ A \Longrightarrow A \vdash_@ (i, p)\rangle$
| *Ref* [*intro*]: $\langle A \vdash_@ (i, Nom\ i)\rangle$
| *Nom* [*intro*]: $\langle A \vdash_@ (i, Nom\ k) \Longrightarrow A \vdash_@ (i, p) \Longrightarrow A \vdash_@ (k, p)\rangle$
| *FlsE* [*elim*]: $\langle A \vdash_@ (i, \bot) \Longrightarrow A \vdash_@ (k, p)\rangle$
| *ImpI* [*intro*]: $\langle(i, p) \#\ A \vdash_@ (i, q) \Longrightarrow A \vdash_@ (i, p \longrightarrow q)\rangle$
| *ImpE* [*elim*]: $\langle A \vdash_@ (i, p \longrightarrow q) \Longrightarrow A \vdash_@ (i, p) \Longrightarrow A \vdash_@ (i, q)\rangle$
| *SatI* [*intro*]: $\langle A \vdash_@ (i, p) \Longrightarrow A \vdash_@ (k, @i\ p)\rangle$
| *SatE* [*elim*]: $\langle A \vdash_@ (i, @k\ p) \Longrightarrow A \vdash_@ (k, p)\rangle$
| *DiaI* [*intro*]: $\langle A \vdash_@ (i, \Diamond\ (Nom\ k)) \Longrightarrow A \vdash_@ (k, p) \Longrightarrow A \vdash_@ (i, \Diamond\ p)\rangle$
| *DiaE* [*elim*]: $\langle A \vdash_@ (i, \Diamond\ p) \Longrightarrow k \notin nominals\ (\{(i, p), (j, q)\} \cup set\ A) \Longrightarrow$
$(k, p) \#\ (i, \Diamond\ (Nom\ k)) \#\ A \vdash_@ (j, q) \Longrightarrow A \vdash_@ (j, q)\rangle$
| *Clas*: $\langle(i, p \longrightarrow q) \#\ A \vdash_@ (i, p) \Longrightarrow A \vdash_@ (i, p)\rangle$
| *Weak*: $\langle A \vdash_@ (i, p) \Longrightarrow (k, q) \#\ A \vdash_@ (i, p)\rangle$

7.4 Soundness

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

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

corollary $\langle\neg\ ([] \vdash_@ (i, \bot))\rangle$
⟨*proof*⟩

7.5 Derived Rules

lemma *Assm-head*: ‹$(p,\ i)\ \#\ A \vdash_@ (p,\ i)$›
⟨*proof*⟩

lemma *deduct1*: ‹$A \vdash_@ (i,\ p \longrightarrow q) \Longrightarrow (i,\ p)\ \#\ A \vdash_@ (i,\ q)$›
⟨*proof*⟩

lemma *Boole*: ‹$(i,\ \neg\ p)\ \#\ A \vdash_@ (i,\ \bot) \Longrightarrow A \vdash_@ (i,\ p)$›
⟨*proof*⟩

lemma *Weak'*: ‹$A \vdash_@ (i,\ p) \Longrightarrow B\ @\ A \vdash_@ (i,\ p)$›
⟨*proof*⟩

lemma *ImpI'*:
assumes ‹$(k,\ q)\ \#\ A \vdash_@ (i,\ p)$›
shows ‹$A \vdash_@ (i,\ (@k\ q) \longrightarrow p)$›
⟨*proof*⟩

lemma *Weaken*: ‹$A \vdash_@ (i,\ p) \Longrightarrow set\ A \subseteq set\ B \Longrightarrow B \vdash_@ (i,\ p)$›
⟨*proof*⟩

interpretation *Derivations Calculus*
⟨*proof*⟩

7.6 Maximal Consistent Sets

definition *consistent* :: ‹$('i,\ 'p)\ lbd\ set \Rightarrow bool$› **where**
‹$consistent\ S \equiv \nexists S'\ a.\ set\ S' \subseteq S \wedge S' \vdash_@ (a,\ \bot)$›

lemma *consistent-add-witness*:
assumes ‹$consistent\ S$› ‹$(i,\ \Diamond\ p) \in S$› ‹$k \notin nominals\ S$›
shows ‹$consistent\ (\{(k,\ p),\ (i,\ \Diamond\ (Nom\ k))\} \cup S)$›
⟨*proof*⟩

fun *witness* :: ‹$('i,\ 'p)\ lbd \Rightarrow ('i,\ 'p)\ lbd\ set \Rightarrow ('i,\ 'p)\ lbd\ set$› **where**
‹$witness\ (i,\ \Diamond\ p)\ S = (let\ k = (SOME\ k.\ k \notin nominals\ (\{(i,\ p)\} \cup S))\ in\ \{(k,\ p),\ (i,\ \Diamond\ (Nom\ k))\})$›
| ‹$witness\ (\text{-},\ \text{-})\ \text{-} = \{\}$›

lemma *consistent-witness'*:
assumes ‹$consistent\ (\{(i,\ p)\} \cup S)$› ‹$infinite\ (UNIV - nominals\ S)$›
shows ‹$consistent\ (witness\ (i,\ p)\ S \cup \{(i,\ p)\} \cup S)$›
⟨*proof*⟩

interpretation *MCS-Saturation consistent nominals-lbd witness*
⟨*proof*⟩

interpretation *Derivations-MCS-Cut Calculus consistent* ‹$(undefined,\ \bot)$›
⟨*proof*⟩

lemma *saturated*: ‹*saturated H* $\Longrightarrow$ $(i, \Diamond p) \in H \Longrightarrow \exists\, k.\ (i, \Diamond\ (Nom\ k)) \in H \wedge (k, p) \in H$›
⟨*proof*⟩

7.7 Nominals

lemma *MCS-Nom-refl*:
assumes ‹*consistent S*› ‹*maximal S*›
shows ‹$(i, Nom\ i) \in S$›
⟨*proof*⟩

lemma *MCS-Nom-sym*:
assumes ‹*consistent S*› ‹*maximal S*› ‹$(i, Nom\ j) \in S$›
shows ‹$(j, Nom\ i) \in S$›
⟨*proof*⟩

lemma *MCS-Nom-trans*:
assumes ‹*consistent S*› ‹*maximal S*› ‹$(i, Nom\ j) \in S$› ‹$(j, Nom\ k) \in S$›
shows ‹$(i, Nom\ k) \in S$›
⟨*proof*⟩

7.8 Truth Lemma

fun *interp* :: ‹$('i, 'p, 'w)\ ctx \Rightarrow (('i, 'p, 'w)\ ctx \Rightarrow ('i, 'p)\ fm \Rightarrow bool) \Rightarrow ('i, 'p)\ fm \Rightarrow bool$›
where
‹$interp\ (M, g, w)\ X\ \bot \longleftrightarrow False$›
| ‹$interp\ (M, -, w)\ X\ (Pro\ x) \longleftrightarrow V\ M\ w\ x$›
| ‹$interp\ (-, g, w)\ X\ (Nom\ i) \longleftrightarrow w = g\ i$›
| ‹$interp\ (M, g, w)\ X\ (p \longrightarrow q) \longleftrightarrow X\ (M, g, w)\ p \longrightarrow X\ (M, g, w)\ q$›
| ‹$interp\ (M, g, w)\ X\ (\Diamond\ p) \longleftrightarrow (\exists\, v \in R\ M\ w.\ X\ (M, g, v)\ p)$›
| ‹$interp\ (M, g, -)\ X\ (@\ i\ p) \longleftrightarrow X\ (M, g, g\ i)\ p$›

fun *rel* :: ‹$('i, 'p)\ lbd\ set \Rightarrow ('i, 'p, 'i)\ ctx \Rightarrow ('i, 'p)\ fm \Rightarrow bool$› **where**
‹$rel\ H\ (-, -, i)\ p = ((i, p) \in H)$›

definition *val* :: ‹$('i, 'p)\ lbd\ set \Rightarrow 'i \Rightarrow 'p \Rightarrow bool$› **where**
‹$val\ H\ i\ x \equiv (i, Pro\ x) \in H$›

definition *hequiv* :: ‹$('i, 'p)\ lbd\ set \Rightarrow 'i \Rightarrow 'i \Rightarrow bool$› **where**
‹$hequiv\ H\ i\ j \equiv (i, Nom\ j) \in H$›

lemma *hequiv-reflp*:
assumes ‹*consistent H*› ‹*maximal H*›
shows ‹$reflp\ (hequiv\ H)$›
⟨*proof*⟩

lemma *hequiv-symp*:
assumes ‹*consistent H*› ‹*maximal H*›

shows ‹*symp* (*hequiv H*)›
⟨*proof*⟩

lemma *hequiv-transp*:
assumes ‹*consistent H*› ‹*maximal H*›
shows ‹*transp* (*hequiv H*)›
⟨*proof*⟩

lemma *hequiv-equivp*:
assumes ‹*consistent H*› ‹*maximal H*›
shows ‹*equivp* (*hequiv H*)›
⟨*proof*⟩

definition *assign* :: ‹('*i*, '*p*) *lbd set* ⇒ '*i* ⇒ '*i*› **where**
‹*assign H i* ≡ *minim* (|*UNIV*|) {*j*. *hequiv H i j*}›

lemma *hequiv-ne*:
assumes ‹*consistent H*› ‹*maximal H*›
shows ‹{*j*. *hequiv H i j*} ≠ {}›
⟨*proof*⟩

lemma *hequiv-assign*:
assumes ‹*consistent H*› ‹*maximal H*›
shows ‹*hequiv H i* (*assign H i*)›
⟨*proof*⟩

lemma *hequiv-Nom*:
assumes ‹*consistent H*› ‹*maximal H*› ‹*hequiv H i j*› ‹(*i*, *p*) ∈ *H*›
shows ‹(*j*, *p*) ∈ *H*›
⟨*proof*⟩

definition *reach* :: ‹('*i*, '*p*) *lbd set* ⇒ '*i* ⇒ '*i set*› **where**
‹*reach H i* ≡ {*assign H j* |*j*. (*i*, ◇ (*Nom j*)) ∈ *H*}›

abbreviation *canonical* :: ‹('*i* × ('*i*, '*p*) *fm*) *set* ⇒ '*i* ⇒ ('*i*, '*p*, '*i*) *ctx*› **where**
‹*canonical H i* ≡ (*Model* (*reach H*) (*val H*), *assign H*, *assign H i*)›

lemma *Hintikka-model'*:
assumes ‹⋀*i p*. *interp* (*canonical H i*) (*rel H*) *p* = *rel H* (*canonical H i*) *p*›
shows ‹(*canonical H i* ⊨ *p*) = *rel H* (*canonical H i*) *p*›
⟨*proof*⟩

lemma *Hintikka-Extend'*:
assumes ‹*consistent H*› ‹*maximal H*› ‹*saturated H*›
shows ‹*interp* (*canonical H i*) (*rel H*) *p* = *rel H* (*canonical H i*) *p*›
⟨*proof*⟩

interpretation *Truth-Saturation*
‹*consistent* :: ('*i*, '*p*) *lbd set* ⇒ *bool*›

nominals-lbd witness interp semantics ‹λH. {canonical H i |i. True}› rel
⟨*proof*⟩

lemma *Truth-lemma*:
assumes ‹*consistent H*› ‹*maximal H*› ‹*saturated H*›
shows ‹(*canonical H i* ⊨ *p*) ⟷ (*i*, *p*) ∈ *H*›
⟨*proof*⟩

7.9 Cardinalities

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

type-synonym ('*i*, '*p*) *enc* = ‹('*i* + '*p*) + *marker* × *nat*›

abbreviation ‹*NOM i* ≡ *Inl* (*Inl i*)›
abbreviation ‹*PRO x* ≡ *Inl* (*Inr x*)›
abbreviation ‹*FLS* ≡ *Inr* (*FlsM*, *0*)›
abbreviation ‹*IMP n* ≡ *Inr* (*FlsM*, *n*)›
abbreviation ‹*DIA* ≡ *Inr* (*DiaM*, *0*)›
abbreviation ‹*SAT* ≡ *Inr* (*SatM*, *0*)›

primrec *encode* :: ‹('*i*, '*p*) *fm* ⇒ ('*i*, '*p*) *enc list*› **where**
‹*encode* ⊥ = [*FLS*]›
| ‹*encode* (*Pro x*) = [*PRO x*]›
| ‹*encode* (*Nom i*) = [*NOM i*]›
| ‹*encode* (*p* ⟶ *q*) = *IMP* (*length* (*encode p*)) # *encode p* @ *encode q*›
| ‹*encode* (◇ *p*) = *DIA* # *encode p*›
| ‹*encode* (@ *i p*) = *SAT* # *NOM i* # *encode p*›

lemma *encode-ne* [*simp*]: ‹*encode p* ≠ []›
⟨*proof*⟩

lemma *inj-encode'*: ‹*encode p* = *encode q* ⟹ *p* = *q*›
⟨*proof*⟩

lemma *inj-encode*: ‹*inj encode*›
⟨*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*. (∀ (*k*, *q*) ∈ *X*. (*M*, *g*, *g k*) ⊨ *q*) ⟶ (*M*, *g*, *g i*) ⊨ *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 i*. (*M*, *g*, *g i*) ⊨ *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** ‹∘⇩9› *0*) **where**
  ‹(*t* ∘⇩9 *s*) *0* = *t*›
| ‹(*t* ∘⇩9 *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**
‹*lift-tm* (#n) = #(n+1)›
| ‹*lift-tm* (†f ts) = †f (*map lift-tm* ts)›

primrec *sub-tm* :: ‹(*nat* ⇒ 'f tm) ⇒ 'f tm ⇒ 'f tm› **where**
‹*sub-tm* s (#n) = s n›
| ‹*sub-tm* s (†f ts) = †f (*map* (*sub-tm* s) ts)›

primrec *sub-fm* :: ‹(*nat* ⇒ 'f tm) ⇒ ('f, 'p) fm ⇒ ('f, 'p) fm› **where**
‹*sub-fm* s ⊥ = ⊥›
| ‹*sub-fm* s (‡P ts) = ‡P (*map* (*sub-tm* s) ts)›
| ‹*sub-fm* s (p ⟶ q) = *sub-fm* s p ⟶ *sub-fm* s q›
| ‹*sub-fm* s (∀ p) = ∀ (*sub-fm* (#0 ⨟ λn. *lift-tm* (s n)) p)›

abbreviation *inst-single* :: ‹'f tm ⇒ ('f, 'p) fm ⇒ ('f, 'p) fm› (‹⟨-⟩›) **where**
‹⟨t⟩ ≡ *sub-fm* (t ⨟ #)›

abbreviation ‹*params'* S ≡ ⋃p ∈ S. *params-fm* p›

abbreviation ‹*params* l ≡ *params'* (*set* l)›

lemma *upd-params-tm* [*simp*]: ‹f ∉ *params-tm* t ⟹ ⦇(E, F(f := x))⦈ t = ⦇(E, F)⦈ t›
⟨*proof*⟩

lemma *upd-params-fm* [*simp*]: ‹f ∉ *params-fm* p ⟹ ⟦(E, F(f := x), G)⟧ p = ⟦(E, F, G)⟧ p›
⟨*proof*⟩

lemma *finite-params-tm* [*simp*]: ‹*finite* (*params-tm* t)›
⟨*proof*⟩

lemma *finite-params-fm* [*simp*]: ‹*finite* (*params-fm* p)›
⟨*proof*⟩

lemma *env* [*simp*]: ‹P ((x ⨟ E) n) = (P x ⨟ λn. P (E n)) n›
⟨*proof*⟩

lemma *lift-lemma*: ‹⦇(x ⨟ E, F)⦈ (*lift-tm* t) = ⦇(E, F)⦈ t›
⟨*proof*⟩

lemma *sub-tm-semantics*: ‹⦇(E, F)⦈ (*sub-tm* s t) = ⦇(λn. ⦇(E, F)⦈ (s n), F)⦈ t›
⟨*proof*⟩

lemma *sub-fm-semantics* [*simp*]: ‹⟦(E, F, G)⟧ (*sub-fm* s p) = ⟦(λn. ⦇(E, F)⦈ (s n), F, G)⟧ p›
⟨*proof*⟩

lemma *sub-tm-Var*: ‹*sub-tm* # *t* = *t*›
⟨*proof*⟩

lemma *reduce-Var* [*simp*]: ‹(# *0* ⨟ λ*n*. # (*Suc n*)) = #›
⟨*proof*⟩

lemma *sub-fm-Var* [*simp*]:
fixes *p* :: ‹('*f*, '*p*) *fm*›
shows ‹*sub-fm* # *p* = *p*›
⟨*proof*⟩

lemma *semantics-tm-id* [*simp*]: ‹⦇(#, †)⦈ *t* = *t*›
⟨*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 Derived 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* ≡ ∄*S′*. *set S′* ⊆ *S* ∧ *S′* ⊢∀ ⊥›

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*⟩*p*} ∪ *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* ≡ (#, †, λ*P ts*. ‡*P ts* ∈ *H*)›

fun *interp* ::
‹('*a*, '*f*, '*p*) *model* ⇒ (('*a*, '*f*, '*p*) *model* ⇒ ('*f*, '*p*) *fm* ⇒ *bool*) ⇒ ('*f*, '*p*) *fm* ⇒ *bool*› **where**
‹*interp* - - ⊥ = *False*›
| ‹*interp* (*E*, *F*, *G*) *X* (‡*P ts*) = *G P* (*map* (⦇(*E*, *F*)⦈) *ts*)›
| ‹*interp* (*E*, *F*, *G*) *X* (*p* ⟶ *q*) = (*X* (*E*, *F*, *G*) *p* ⟶ *X* (*E*, *F*, *G*) *q*)›
| ‹*interp* (*E*, *F*, *G*) *X* (∀ *p*) = (∀ *x*. *X* (*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*. *interp* (*hmodel H*) (*rel H*) *p* ⟷ *p* ∈ *H*›
shows ‹*p* ∈ *H* ⟷ ⟦*hmodel H*⟧ *p*›
⟨*proof*⟩

lemma *Hintikka-Extend*:
assumes ‹*consistent H*› ‹*maximal H*› ‹*saturated H*›
shows ‹*interp* (*hmodel H*) (*rel H*) *p* ⟷ *p* ∈ *H*›
⟨*proof*⟩

interpretation *Truth-Saturation*
consistent params-fm witness interp semantics-fm ‹λ*H*. {*hmodel H*}› *rel*
⟨*proof*⟩

8.9 Cardinalities

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

type-synonym ('*f*, '*p*) *enc* = ‹('*f* + '*p*) + *marker* × *nat*›

abbreviation ‹*FUNS f* ≡ *Inl* (*Inl f*)›
abbreviation ‹*PRES p* ≡ *Inl* (*Inr p*)›

abbreviation ‹*VAR n* ≡ *Inr* (*VarM*, *n*)›
abbreviation ‹*FUN n* ≡ *Inr* (*FunM*, *n*)›
abbreviation ‹*TM n* ≡ *Inr* (*TmM*, *n*)›

abbreviation ‹*PRE n* ≡ *Inr* (*PreM*, *n*)›

abbreviation ‹*FLS* ≡ *Inr* (*FlsM*, *0*)›
abbreviation ‹*IMP n* ≡ *Inr* (*FlsM*, *n*)›
abbreviation ‹*UNI* ≡ *Inr* (*UniM*, *0*)›

primrec
  encode-tm :: ‹'*f tm* ⇒ ('*f*, '*p*) *enc list*› **and**
  encode-tms :: ‹'*f tm list* ⇒ ('*f*, '*p*) *enc list*› **where**
  ‹*encode-tm* (#*n*) = [*VAR n*]›
| ‹*encode-tm* (†*f ts*) = *FUN* (*length ts*) # *FUNS f* # *encode-tms ts*›
| ‹*encode-tms* [] = []›
| ‹*encode-tms* (*t* # *ts*) = *TM* (*length* (*encode-tm t*)) # *encode-tm t* @ *encode-tms ts*›

lemma *encode-tm-ne* [*simp*]: ‹*encode-tm t* ≠ []›
  ⟨*proof*⟩

lemma *inj-encode-tm'*:
  ‹(*encode-tm t* :: ('*f*, '*p*) *enc list*) = *encode-tm s* ⟹ *t* = *s*›
  ‹(*encode-tms ts* :: ('*f*, '*p*) *enc list*) = *encode-tms ss* ⟹ *ts* = *ss*›
⟨*proof*⟩

lemma *inj-encode-tm*: ‹*inj encode-tm*›
  ⟨*proof*⟩

primrec *encode-fm* :: ‹('*f*, '*p*) *fm* ⇒ ('*f*, '*p*) *enc list*› **where**
  ‹*encode-fm* ⊥ = [*FLS*]›
| ‹*encode-fm* (‡*P ts*) = *PRE* (*length ts*) # *PRES P* # *encode-tms ts*›
| ‹*encode-fm* (*p* ⟶ *q*) = *IMP* (*length* (*encode-fm p*)) # *encode-fm p* @ *encode-fm q*›
| ‹*encode-fm* (∀ *p*) = *UNI* # *encode-fm p*›

lemma *encode-fm-ne* [*simp*]: ‹*encode-fm p* ≠ []›
  ⟨*proof*⟩

lemma *inj-encode-fm'*: ‹*encode-fm p* = *encode-fm q* ⟹ *p* = *q*›
⟨*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 ‹$\forall M :: ('f\ tm,\ 'f,\ 'p)\ model.\ (\forall q \in X.\ [\![M]\!]\ q) \longrightarrow [\![M]\!]\ p$›
‹$infinite\ (UNIV :: 'f\ set)$›
‹$|UNIV :: 'f\ set| +c |UNIV :: 'p\ set| \leq o\ |UNIV - params'\ X|$›
shows ‹$\exists A.\ set\ A \subseteq X \wedge A \vdash_\forall p$›
⟨*proof*⟩

abbreviation *valid* :: ‹$('f,\ 'p)\ fm \Rightarrow bool$› **where**
‹$valid\ p \equiv \forall M :: ('f\ tm,\ -,\ -)\ model.\ [\![M]\!]\ p$›

theorem *completeness*:
fixes p :: ‹$('f,\ 'p)\ fm$›
assumes ‹$valid\ p$› ‹$infinite\ (UNIV :: 'f\ set)$› ‹$|UNIV :: 'p\ set| \leq o\ |UNIV :: 'f\ set|$›
shows ‹$[] \vdash_\forall p$›
⟨*proof*⟩

corollary *completeness'*:
fixes p :: ‹$('f,\ 'f)\ fm$›
assumes ‹$valid\ p$› ‹$infinite\ (UNIV :: 'f\ set)$›
shows ‹$[] \vdash_\forall p$›
⟨*proof*⟩

theorem *main*:
fixes p :: ‹$('f,\ 'p)\ fm$›
assumes ‹$infinite\ (UNIV :: 'f\ set)$› ‹$|UNIV :: 'p\ set| \leq o\ |UNIV :: 'f\ set|$›
shows ‹$valid\ p \longleftrightarrow [] \vdash_\forall p$›
⟨*proof*⟩

corollary *main'*:
fixes p :: ‹$('f,\ 'f)\ fm$›
assumes ‹$infinite\ (UNIV :: 'f\ set)$›
shows ‹$valid\ p \longleftrightarrow [] \vdash_\forall p$›
⟨*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.