Synthetic Completeness

Asta Halkjær From

March 17, 2025

Abstract

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

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

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

Contents

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*⟩

context *wo-rel* **begin**

lemma *underS-bound*: ‹$a \in$ *underS* $c \Longrightarrow b \in$ *underS* $c \Longrightarrow a \in$ *under* $b \vee b \in$ *under* a›
⟨*proof*⟩

lemma *finite-underS-bound*:
assumes ‹*finite* X› ‹$X \subseteq$ *underS* c› ‹$X \neq \{\}$›
shows ‹$\exists\, a \in X.\ \forall\, b \in X.\ b \in$ *under* a›
⟨*proof*⟩

lemma *finite-bound-under*:
assumes ‹*finite* p› ‹$p \subseteq (\bigcup a \in$ *Field* $r.\ f\ a)$›
shows ‹$\exists\, b.\ p \subseteq (\bigcup a \in$ *under* $b.\ f\ a)$›
⟨*proof*⟩

lemma *underS-trans*: ‹$a \in underS\ b \Longrightarrow b \in underS\ c \Longrightarrow a \in underS\ c$›
⟨*proof*⟩

end

lemma *card-of-infinite-smaller-Union*:
assumes ‹$\forall x.\ |f\ x| <o |X|$› ‹*infinite X*›
shows ‹$|\bigcup x \in X.\ f\ x| \leq o\ |X|$›
⟨*proof*⟩

lemma *card-of-params-marker-lists*:
assumes ‹*infinite* (*UNIV* :: *'i set*)› ‹$|UNIV :: 'm\ set| \leq o\ |UNIV :: nat\ set|$›
shows ‹$|UNIV :: ('i + 'm \times nat)\ list\ set| \leq o\ |UNIV :: 'i\ set|$›
⟨*proof*⟩

1.2 Base Locales

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

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

end

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

definition *witnessed* :: ‹*'a set* ⇒ *bool*› **where**
‹*witnessed S* $\equiv \forall p \in S. \exists S'.$ *witness p S'* $\subseteq S$›

abbreviation *MCS* :: ‹*'a set* ⇒ *bool*› **where**
‹*MCS S* $\equiv$ *consistent S* $\wedge$ *maximal S* $\wedge$ *witnessed S*›

end

locale *MCS-No-Witness* = *MCS-Base consistent* **for** *consistent* :: ‹*'a set* ⇒ *bool*›

sublocale *MCS-No-Witness* $\subseteq$ *MCS-Witness consistent* ‹λ- -. {}› ‹λ-. {}›
⟨*proof*⟩

1.3 Ordinal Locale

locale *MCS-Lim-Ord* = *MCS-Witness consistent witness params*

for *consistent* :: ‹*'a set ⇒ bool*›
and *witness* :: ‹*'a ⇒ 'a set ⇒ 'a set*›
and *params* :: ‹*'a ⇒ 'i set*› +
fixes *r* :: ‹*'a rel*›
assumes *Cinfinite-r*: ‹*Cinfinite r*›
begin

lemma *WELL*: ‹*Well-order r*›
⟨*proof*⟩

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

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

lemma *nonempty-Field-r*: ‹*Field r ≠ {}*›
⟨*proof*⟩

1.3.1 Lindenbaum Extension

abbreviation *paramss* :: ‹*'a set ⇒ 'i set*› **where**
‹*paramss S ≡ ⋃ p ∈ S. params p*›

definition *extendS* :: ‹*'a ⇒ 'a set ⇒ 'a set*› **where**
‹*extendS a prev ≡ if consistent ({a} ∪ prev) then {a} ∪ prev ∪ witness a prev else prev*›

definition *extendL* :: ‹*('a ⇒ 'a set) ⇒ 'a ⇒ 'a set*› **where**
‹*extendL rec a ≡ ⋃ b ∈ underS r a. rec b*›

definition *extend* :: ‹*'a set ⇒ 'a ⇒ 'a set*› **where**
‹*extend S a ≡ worecZSL r S extendS extendL a*›

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

definition *Extend* :: ‹*'a set ⇒ 'a set*› **where**
‹*Extend S ≡ ⋃ a ∈ Field r. extend S a*›

lemma *extend-subset*: ‹*a ∈ Field r ⟹ S ⊆ extend S a*›
⟨*proof*⟩

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

lemma *extend-underS*: ‹*b ∈ underS r a ⟹ extend S b ⊆ extend S a*›
⟨*proof*⟩

lemma *extend-under*: ‹*b ∈ under r a ⟹ extend S b ⊆ extend S a*›
⟨*proof*⟩

1.3.2 Consistency

lemma *params-origin*:
assumes ‹*x ∈ paramss (extend S a)*›
shows ‹*x ∈ paramss S ∨ (∃ b ∈ underS r a. x ∈ paramss ({b} ∪ witness b (extend S b)))*›

⟨*proof*⟩

lemma *consistent-extend*:
assumes ‹*consistent S*› ‹*r* ≤*o* |*UNIV* − *paramss S*|›
shows ‹*consistent* (*extend S a*)›
⟨*proof*⟩

lemma *consistent-Extend*:
assumes ‹*consistent S*› ‹*r* ≤*o* |*UNIV* − *paramss S*|›
shows ‹*consistent* (*Extend S*)›
⟨*proof*⟩

lemma *Extend-bound*: ‹*a* ∈ *Field r* ⟹ *extend S a* ⊆ *Extend S*›
⟨*proof*⟩

1.3.3 Maximality

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

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

1.3.4 Witnessing

definition *witnessed'* :: ‹'*a set* ⇒ *bool*› **where**
‹*witnessed' S* ≡ ∀ *p* ∈ *Field r*. *p* ∈ *S* ⟶ (∃ *S'*. *witness p S'* ⊆ *S*)›

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

end

1.4 Locales for Universe Well-Order

locale *MCS-Witness-UNIV* = *MCS-Witness consistent witness params*
for *consistent* :: ‹'*a set* ⇒ *bool*›
and *witness* :: ‹'*a* ⇒ '*a set* ⇒ '*a set*›
and *params* :: ‹'*a* ⇒ '*i set*› +
assumes *infinite-UNIV*: ‹*infinite* (*UNIV* :: '*a set*)›

sublocale *MCS-Witness-UNIV* ⊆ *MCS-Lim-Ord consistent witness params* ‹|*UNIV*|›
⟨*proof*⟩

context *MCS-Witness-UNIV* **begin**

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

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

lemma *witnessed-witnessed'*: ‹*witnessed S* ⟷ *witnessed' S*›

⟨*proof*⟩

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

theorem *MCS-Extend*:
assumes ‹*consistent S*› ‹|*UNIV* :: *'a set*| ≤*o* |*UNIV* − *paramss S*|›
shows ‹*MCS* (*Extend S*)›
⟨*proof*⟩

end

locale *MCS-No-Witness-UNIV* = *MCS-No-Witness consistent*
for *consistent* :: ‹*'a set* ⇒ *bool*› +
assumes *infinite-UNIV'* [*simp*]: ‹*infinite* (*UNIV* :: *'a set*)›

sublocale *MCS-No-Witness-UNIV* ⊆ *MCS-Witness-UNIV consistent* ‹λ- -. {}› ‹λ-. {}›
⟨*proof*⟩

context *MCS-No-Witness-UNIV*
begin

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

end

1.5 Truth Lemma

locale *Truth-Base* =
fixes *semics* :: ‹*'model* ⇒ (*'model* ⇒ *'fm* ⇒ *bool*) ⇒ *'fm* ⇒ *bool*› (‹(- ⟦-⟧ -)› [*55*, *0*, *55*] *55*)
and *semantics* :: ‹*'model* ⇒ *'fm* ⇒ *bool*› (**infix** ‹⊨› *50*)
and $\mathcal{M}$ :: ‹*'a set* ⇒ *'model set*›
and $\mathcal{R}$ :: ‹*'a set* ⇒ *'model* ⇒ *'fm* ⇒ *bool*›
assumes *semics-semantics*: ‹$M \models p \longleftrightarrow M\ [\![(\models)]\!]\ p$›
begin

abbreviation *saturated* :: ‹*'a set* ⇒ *bool*› **where**
‹*saturated S* ≡ ∀ *p*. ∀ $M \in \mathcal{M}(S)$. $M\ [\![\mathcal{R}(S)]\!]\ p \longleftrightarrow \mathcal{R}(S)\ M\ p$›

end

locale *Truth-Witness* = *Truth-Base semics semantics* $\mathcal{M}$ $\mathcal{R}$ + *MCS-Witness consistent witness params*
for *semics* :: ‹*'model* ⇒ (*'model* ⇒ *'fm* ⇒ *bool*) ⇒ *'fm* ⇒ *bool*› (‹(- ⟦-⟧ -)› [*55*, *0*, *55*] *55*)
and *semantics* :: ‹*'model* ⇒ *'fm* ⇒ *bool*› (**infix** ‹⊨› *50*)
and $\mathcal{M}$ :: ‹*'a set* ⇒ *'model set*›
and $\mathcal{R}$ :: ‹*'a set* ⇒ *'model* ⇒ *'fm* ⇒ *bool*›
and *consistent* :: ‹*'a set* ⇒ *bool*›
and *witness* :: ‹*'a* ⇒ *'a set* ⇒ *'a set*›
and *params* :: ‹*'a* ⇒ *'i set*› +
assumes *saturated-semantics*: ‹$\bigwedge S\ M\ p.$ *saturated S* ⟹ $M \in \mathcal{M}(S)$ ⟹ $M \models p \longleftrightarrow \mathcal{R}(S)\ M\ p$›

and *MCS-saturated*: ‹$\bigwedge S.\ MCS\ S \Longrightarrow saturated\ S$›
begin

theorem *truth-lemma*:
assumes ‹$MCS\ S$› ‹$M \in \mathcal{M}(S)$›
shows ‹$M \models p \longleftrightarrow \mathcal{R}(S)\ M\ p$›
⟨*proof*⟩

end

locale *Truth-No-Witness* = *Truth-Witness semics semantics* $\mathcal{M}$ $\mathcal{R}$ *consistent* ‹λ- -. {}› ‹λ-. {}›
for *semics* :: ‹$'model \Rightarrow ('model \Rightarrow 'fm \Rightarrow bool) \Rightarrow 'fm \Rightarrow bool$›
and *semantics* :: ‹$'model \Rightarrow 'fm \Rightarrow bool$›
and $\mathcal{M}$ :: ‹$'a\ set \Rightarrow 'model\ set$›
and $\mathcal{R}$ :: ‹$'a\ set \Rightarrow 'model \Rightarrow 'fm \Rightarrow bool$›
and *consistent* :: ‹$'a\ set \Rightarrow bool$›

end

Chapter 2

Derivations

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

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

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

2.1 Derivations

locale *Derivations* =
  fixes *derive* :: ‹*'fm list* ⇒ *'fm* ⇒ *bool*› (**infix** ‹⊢› *50*)
  assumes *derive-assm* [*simp*]: ‹$\bigwedge A\ p.\ p \in$ *set A* $\Longrightarrow A \vdash p$›
    and *derive-set*: ‹$\bigwedge A\ B\ r.\ A \vdash r \Longrightarrow$ *set A* = *set B* $\Longrightarrow B \vdash r$›
begin

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

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

end

2.2 MCSs and Explosion

locale *Derivations-MCS* = *MCS-Base consistent* + *Derivations derive*
  for *consistent* :: ‹*'fm set* ⇒ *bool*›
    and *derive* :: ‹*'fm list* ⇒ *'fm* ⇒ *bool*› (**infix** ‹⊢› *50*) +
  assumes *consistent-underivable*: ‹$\bigwedge S.$ *consistent* $S \longleftrightarrow (\forall A.$ *set* $A \subseteq S \longrightarrow (\exists q. \neg A \vdash q))$›
begin

theorem *MCS-explode*:
assumes ‹*consistent S*› ‹*maximal S*›
shows ‹$p \notin S \longleftrightarrow (\exists A.\ set\ A \subseteq S \wedge (\forall q.\ p \#\ A \vdash q))$›
⟨*proof*⟩

end

2.3 MCSs and Derivability

locale *Derivations-Cut-MCS* = *Derivations-MCS consistent derive*
for *consistent* :: ‹*'fm set* ⇒ *bool*›
and *derive* :: ‹*'fm list* ⇒ *'fm* ⇒ *bool*› (**infix** ‹⊢› *50*) +
assumes *derive-cut*: ‹$\bigwedge A\ B\ p\ q.\ A \vdash p \Longrightarrow p \#\ B \vdash q \Longrightarrow A\ @\ B \vdash q$›
begin

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

end

2.4 Proof Rules

locale *Derivations-Bot* = *Derivations-Cut-MCS consistent derive*
for *consistent* :: ‹*'fm set* ⇒ *bool*›
and *derive* :: ‹*'fm list* ⇒ *'fm* ⇒ *bool*› (**infix** ‹⊢› *50*) +
fixes *bot* :: *'fm* (‹⊥›)
assumes *botE*: ‹$\bigwedge A\ p.\ A \vdash \bot \Longrightarrow A \vdash p$›
begin

corollary *MCS-botE* [*elim*]:
assumes ‹*consistent S*› ‹*maximal S*›
and ‹$\bot \in S$›
shows ‹$p \in S$›
⟨*proof*⟩

corollary *MCS-bot* [*simp*]:
assumes ‹*consistent S*› ‹*maximal S*›
shows ‹$\bot \notin S$›
⟨*proof*⟩

end

locale *Derivations-Top* = *Derivations-Cut-MCS* +
fixes *top* (‹⊤›)
assumes *topI*: ‹$\bigwedge A.\ A \vdash \top$›
begin

corollary *MCS-topI* [*simp*]:
assumes ‹*consistent S*› ‹*maximal S*›
shows ‹$\top \in S$›
⟨*proof*⟩

end

locale *Derivations-Not = Derivations-Bot consistent derive bot*
  for *consistent* :: ‹*'fm set ⇒ bool*›
    and *derive* :: ‹*'fm list ⇒ 'fm ⇒ bool*› (**infix** ‹⊢› *50*)
    and *bot* :: *'fm* (‹⊥›) +
  fixes *not* :: ‹*'fm ⇒ 'fm*› (‹¬›)
  assumes
    notI: ‹⋀*A p*. *p # A ⊢ ⊥ ⟹ A ⊢ ¬ p*› **and**
    notE: ‹⋀*A p*. *A ⊢ p ⟹ A ⊢ ¬ p ⟹ A ⊢ ⊥*›
begin

corollary *MCS-not-xor*:
  assumes ‹*consistent S*› ‹*maximal S*›
  shows ‹*p ∈ S ⟷ ¬ p ∉ S*›
⟨*proof*⟩

corollary *MCS-not-both*:
  assumes ‹*consistent S*› ‹*maximal S*›
  shows ‹*p ∉ S ∨ ¬ p ∉ S*›
  ⟨*proof*⟩

corollary *MCS-not-neither*:
  assumes ‹*consistent S*› ‹*maximal S*›
  shows ‹*p ∈ S ∨ ¬ p ∈ S*›
  ⟨*proof*⟩

end

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

corollary *MCS-conI* [*intro*]:
  assumes ‹*consistent S*› ‹*maximal S*›
    and ‹*p ∈ S*› ‹*q ∈ S*›
  shows ‹*(p ∧ q) ∈ S*›
  ⟨*proof*⟩

corollary *MCS-conE* [*dest*]:
  assumes ‹*consistent S*› ‹*maximal S*›
    and ‹*(p ∧ q) ∈ S*›
  shows ‹*p ∈ S ∧ q ∈ S*›
⟨*proof*⟩

corollary *MCS-con*:
  assumes ‹*consistent S*› ‹*maximal S*›
  shows ‹*(p ∧ q) ∈ S ⟷ p ∈ S ∧ q ∈ S*›
  ⟨*proof*⟩

end

locale *Derivations-Dis = Derivations-Cut-MCS consistent derive*
  for *consistent* :: ‹*'fm set ⇒ bool*›
    and *derive* :: ‹*'fm list ⇒ 'fm ⇒ bool*› (**infix** ‹⊢› *50*) +
  fixes *dis* :: ‹*'fm ⇒ 'fm ⇒ 'fm*› (‹- ∨ -›)
  assumes
    disI1: ‹$\bigwedge A\ p\ q.\ A \vdash p \Longrightarrow A \vdash (p \vee q)$› **and**
    disI2: ‹$\bigwedge A\ p\ q.\ A \vdash q \Longrightarrow A \vdash (p \vee q)$› **and**
    disE: ‹$\bigwedge A\ p\ q\ r.\ A \vdash (p \vee q) \Longrightarrow p \# A \vdash r \Longrightarrow q \# A \vdash r \Longrightarrow A \vdash r$›
begin

corollary *MCS-disI1* [*intro*]:
  assumes ‹*consistent S*› ‹*maximal S*›
    and ‹$p \in S$›
  shows ‹$(p \vee q) \in S$›
  ⟨*proof*⟩

corollary *MCS-disI2* [*intro*]:
  assumes ‹*consistent S*› ‹*maximal S*›
    and ‹$q \in S$›
  shows ‹$(p \vee q) \in S$›
  ⟨*proof*⟩

corollary *MCS-disE* [*elim*]:
  assumes ‹*consistent S*› ‹*maximal S*›
    and ‹$(p \vee q) \in S$›
  shows ‹$p \in S \vee q \in S$›
⟨*proof*⟩

corollary *MCS-dis*:
  assumes ‹*consistent S*› ‹*maximal S*›
  shows ‹$(p \vee q) \in S \longleftrightarrow p \in S \vee q \in S$›
  ⟨*proof*⟩

end

locale *Derivations-Imp = Derivations-Cut-MCS consistent derive*
  for *consistent* :: ‹*'fm set ⇒ bool*›
    and *derive* :: ‹*'fm list ⇒ 'fm ⇒ bool*› (**infix** ‹⊢› *50*) +
  fixes *imp* :: ‹*'fm ⇒ 'fm ⇒ 'fm*› (‹- → -›)
  assumes
    impI: ‹$\bigwedge A\ p\ q.\ p \# A \vdash q \Longrightarrow A \vdash (p \to q)$› **and**
    impE: ‹$\bigwedge A\ p\ q.\ A \vdash p \Longrightarrow A \vdash (p \to q) \Longrightarrow A \vdash q$›
begin

corollary *MCS-impI* [*intro*]:
  assumes ‹*consistent S*› ‹*maximal S*›
    and ‹$p \in S \longrightarrow q \in S$›
  shows ‹$(p \to q) \in S$›
    ⟨*proof*⟩

corollary *MCS-impE* [*dest*]:
  assumes ‹*consistent S*› ‹*maximal S*›
    and ‹$(p \to q) \in S$› ‹$p \in S$›
  shows ‹$q \in S$›
  ⟨*proof*⟩

corollary *MCS-imp*:
assumes ‹*consistent S*› ‹*maximal S*›
shows ‹$(p \longrightarrow q) \in S \longleftrightarrow (p \in S \longrightarrow q \in S)$›
⟨*proof*⟩

end

locale *Derivations-Exi = MCS-Witness consistent witness params + Derivations-Cut-MCS consistent derive*
for *consistent* :: ‹*'fm set* ⇒ *bool*›
and *witness params*
and *derive* :: ‹*'fm list* ⇒ *'fm* ⇒ *bool*› (**infix** ‹⊢› *50*) +
fixes *exi* :: ‹*'fm* ⇒ *'fm*› (‹∃›)
and *inst* :: ‹*'t* ⇒ *'fm* ⇒ *'fm*› (‹⟨-⟩›)
assumes
exi-witness: ‹$\bigwedge S\ S'\ p.\ MCS\ S \Longrightarrow witness\ (\exists\, p)\ S' \subseteq S \Longrightarrow \exists\, t.\ \langle t\rangle p \in S$› **and**
exiI: ‹$\bigwedge A\ p\ t.\ A \vdash \langle t\rangle p \Longrightarrow A \vdash \exists\, p$›
begin

corollary *MCS-exiI* [*intro*]:
assumes ‹*consistent S*› ‹*maximal S*›
and ‹$\langle t\rangle p \in S$›
shows ‹$\exists\, p \in S$›
⟨*proof*⟩

corollary *MCS-exiE* [*dest*]:
assumes ‹*consistent S*› ‹*maximal S*› ‹*witnessed S*›
and ‹$\exists\, p \in S$›
shows ‹$\exists\, t.\ \langle t\rangle p \in S$›
⟨*proof*⟩

corollary *MCS-exi*:
assumes ‹*consistent S*› ‹*maximal S*› ‹*witnessed S*›
shows ‹$\exists\, p \in S \longleftrightarrow (\exists\, t.\ \langle t\rangle p \in S)$›
⟨*proof*⟩

end

locale *Derivations-Uni = MCS-Witness consistent witness params + Derivations-Not consistent derive bot not*
for *consistent* :: ‹*'fm set* ⇒ *bool*›
and *witness params*
and *derive* :: ‹*'fm list* ⇒ *'fm* ⇒ *bool*› (**infix** ‹⊢› *50*)
and *bot* :: *'fm* (‹⊥›)
and *not* :: ‹*'fm* ⇒ *'fm*› (‹¬›) +
fixes *uni* :: ‹*'fm* ⇒ *'fm*› (‹∀›)
and *inst* :: ‹*'t* ⇒ *'fm* ⇒ *'fm*› (‹⟨-⟩›)
assumes
uni-witness: ‹$\bigwedge S\ S'\ p.\ MCS\ S \Longrightarrow witness\ (\neg\ (\forall\, p))\ S' \subseteq S \Longrightarrow \exists\, t.\ \neg\ (\langle t\rangle p) \in S$› **and**
uniE: ‹$\bigwedge A\ p\ t.\ A \vdash \forall\, p \Longrightarrow A \vdash \langle t\rangle p$›
begin

corollary *MCS-uniE* [*dest*]:
assumes ‹*consistent S*› ‹*maximal S*›
and ‹$\forall\, p \in S$›

shows ‹$\langle t \rangle p \in S$›
⟨*proof*⟩

corollary *MCS-uniI* [*intro*]:
assumes ‹*consistent S*› ‹*maximal S*› ‹*witnessed S*›
and ‹$\forall\, t.\ \langle t \rangle p \in S$›
shows ‹$\forall\, p \in S$›
⟨*proof*⟩

corollary *MCS-uni*:
assumes ‹*consistent S*› ‹*maximal S*› ‹*witnessed S*›
shows ‹$\forall\, p \in S \longleftrightarrow (\forall\, t.\ \langle t \rangle p \in S)$›
⟨*proof*⟩

end

end

Chapter 3

Refutations

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

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

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

3.1 Rearranging Refutations

locale *Refutations* =
  fixes *refute* :: ‹'*fm list* ⇒ *bool*›
  assumes *refute-set*: ‹$\bigwedge A\ B.$ *refute A* ⟹ *set A* = *set B* ⟹ *refute B*›
begin

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

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

end

3.2 MCSs and Refutability

locale *Refutations-MCS* = *MCS-Base* + *Refutations* +
  assumes *consistent-refute*: ‹$\bigwedge S.$ *consistent S* $\longleftrightarrow$ $(\forall\, A.$ *set A* $\subseteq$ *S* $\longrightarrow \neg$ *refute A*)›
begin

theorem *MCS-refute*:
  assumes ‹*consistent S*› ‹*maximal S*›

shows ‹$p \notin S \longleftrightarrow (\exists A.\ set\ A \subseteq S \wedge refute\ (p\ \#\ A))$›
⟨*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*› (**infix** ‹⊨$_T$› *50*) **where**
‹*I* ⊨$_T$ •*P* ⟷ *I P*›
| ‹*I* ⊨$_T$ ¬ *p* ⟷ ¬ *I* ⊨$_T$ *p*›
| ‹*I* ⊨$_T$ *p* ⟶ *q* ⟷ *I* ⊨$_T$ *p* ⟶ *I* ⊨$_T$ *q*›

4.3 Calculus

inductive *Calculus* :: ‹*'p fm list* ⇒ *bool*› (‹⊢$_T$ -› [*50*] *50*) **where**
Axiom [*simp*]: ‹⊢$_T$ •*P* # ¬ •*P* # *A*›
| *NegI* [*intro*]: ‹⊢$_T$ *p* # *A* ⟹ ⊢$_T$ ¬ ¬ *p* # *A*›
| *ImpP* [*intro*]: ‹⊢$_T$ ¬ *p* # *A* ⟹ ⊢$_T$ *q* # *A* ⟹ ⊢$_T$ (*p* ⟶ *q*) # *A*›
| *ImpN* [*intro*]: ‹⊢$_T$ *p* # ¬ *q* # *A* ⟹ ⊢$_T$ ¬ (*p* ⟶ *q*) # *A*›
| *Weak*: ‹⊢$_T$ *A* ⟹ *set A* ⊆ *set B* ⟹ ⊢$_T$ *B*›

lemma *Weak2*:
assumes ‹⊢$_T$ *p* # *A*› ‹⊢$_T$ *q* # *B*›
shows ‹⊢$_T$ *p* # *A* @ *B* ∧ ⊢$_T$ *q* # *A* @ *B*›
⟨*proof*⟩

4.4 Soundness

theorem *soundness*: ‹⊢$_T$ *A* ⟹ ∃ *p* ∈ *set A*. ¬ *I* ⊨$_T$ *p*›
⟨*proof*⟩

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

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

4.5 Maximal Consistent Sets

definition *consistent* :: ‹*'p fm set* ⇒ *bool*› **where**
‹*consistent* $S \equiv \forall A.$ *set* $A \subseteq S \longrightarrow \neg \vdash_T A$›

interpretation *MCS-No-Witness-UNIV consistent*
⟨*proof*⟩

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

4.6 Truth Lemma

abbreviation (*input*) *canonical* :: ‹*'p fm set* ⇒ *'p model*› (‹$\mathcal{M}_T$›) **where**
‹$\mathcal{M}_T(S) \equiv \lambda P.\ \cdot P \in S$›

locale *Hintikka* =
fixes H :: ‹*'a fm set*›
assumes *AxiomH*: ‹$\bigwedge P.\ \cdot P \in H \Longrightarrow \neg\ \cdot P \in H \Longrightarrow$ *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 ‹$(p \in H \longrightarrow \mathcal{M}_T(H) \models_T p) \wedge (\neg\ p \in H \longrightarrow \neg\ \mathcal{M}_T(H) \models_T p)$›
⟨*proof*⟩

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

lemma *truth-lemma*:
assumes ‹*MCS* H› ‹$p \in H$›
shows ‹$\mathcal{M}_T(H) \models_T p$›
⟨*proof*⟩

4.7 Completeness

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

abbreviation *valid* :: ‹*'p fm* ⇒ *bool*› **where**
‹*valid* $p \equiv \forall M.\ M \models_T p$›

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*› (**infix** ‹⊨$_S$› *50*) **where**
‹- ⊨$_S$ ⊥ ⟷ *False*›
| ‹*I* ⊨$_S$ •*P* ⟷ *I P*›
| ‹*I* ⊨$_S$ *p* ⟶ *q* ⟷ *I* ⊨$_S$ *p* ⟶ *I* ⊨$_S$ *q*›

5.3 Calculus

inductive *Calculus* :: ‹*'p fm list* ⇒ *'p fm list* ⇒ *bool*› (**infix** ‹⊢$_S$› *50*) **where**
Axiom [*simp*]: ‹*p* # *A* ⊢$_S$ *p* # *B*›
| *FlsL* [*simp*]: ‹⊥ # *A* ⊢$_S$ *B*›
| *FlsR* [*elim*]: ‹*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*: ‹*A* ⊢$_S$ [*p*] ⟹ *p* # *A* ⊢$_S$ *B* ⟹ *A* ⊢$_S$ *B*›
| *WeakL*: ‹*A* ⊢$_S$ *B* ⟹ *set A* ⊆ *set A'* ⟹ *A'* ⊢$_S$ *B*›
| *WeakR*: ‹*A* ⊢$_S$ *B* ⟹ *set B* ⊆ *set B'* ⟹ *A* ⊢$_S$ *B'*›

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

5.4 Soundness

theorem *soundness*: ‹$A \vdash_S B \Longrightarrow \forall\, q \in \mathit{set}\ A.\ I \models_S q \Longrightarrow \exists\, p \in \mathit{set}\ B.\ I \models_S p$›
⟨*proof*⟩

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

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

5.5 Maximal Consistent Sets

definition *consistent* :: ‹$'p\ \mathit{fm}\ \mathit{set} \Rightarrow \mathit{bool}$› **where**
‹$\mathit{consistent}\ S \equiv \forall\, A.\ \mathit{set}\ A \subseteq S \longrightarrow \neg\ A \vdash_S [\bot]$›

interpretation *MCS-No-Witness-UNIV consistent*
⟨*proof*⟩

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

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

interpretation *Derivations-Imp consistent* ‹$\lambda A\ p.\ A \vdash_S [p]$› ‹$\lambda p\ q.\ p \longrightarrow q$›
⟨*proof*⟩

5.6 Truth Lemma

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

fun *semics* :: ‹$'p\ \mathit{model} \Rightarrow ('p\ \mathit{model} \Rightarrow 'p\ \mathit{fm} \Rightarrow \mathit{bool}) \Rightarrow 'p\ \mathit{fm} \Rightarrow \mathit{bool}$›
(‹- $[\![\text{-}]\!]_S$ -› [55, 0, 55] 55) **where**
‹- $[\![\text{-}]\!]_S\ \bot \longleftrightarrow \mathit{False}$›
| ‹$I\ [\![\text{-}]\!]_S\ \cdot P \longleftrightarrow I\ P$›
| ‹$I\ [\![\mathcal{R}]\!]_S\ p \longrightarrow q \longleftrightarrow \mathcal{R}\ I\ p \longrightarrow \mathcal{R}\ I\ q$›

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

theorem *saturated-model*:
assumes ‹$\bigwedge p.\ \forall\, M \in \{\mathcal{M}_S(S)\}.\ M\ [\![\mathcal{R}_S(S)]\!]_S\ p = \mathcal{R}_S(S)\ M\ p$› ‹$M \in \{\mathcal{M}_S(S)\}$›
shows ‹$\mathcal{R}_S(S)\ M\ p \longleftrightarrow M \models_S p$›
⟨*proof*⟩

theorem *saturated-MCS*:
assumes ‹$\mathit{MCS}\ S$› ‹$M \in \{\mathcal{M}_S(S)\}$›
shows ‹$M\ [\![\mathcal{R}_S(S)]\!]_S\ p \longleftrightarrow \mathcal{R}_S(S)\ M\ p$›
⟨*proof*⟩

interpretation *Truth-No-Witness semics semantics* ‹$\lambda S.\ \{\mathcal{M}_S(S)\}$› *rel consistent*
⟨*proof*⟩

5.7 Completeness

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

abbreviation *valid* :: ‹$'p\ fm \Rightarrow bool$› **where**
  ‹$valid\ p \equiv \forall M.\ M \models_S p$›

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* (‹$\bot$›)
  | *Pro* $'p$ (‹•›)
  | *Imp* ‹$('i, 'p)$ *fm*› ‹$('i, 'p)$ *fm*› (**infixr** ‹$\longrightarrow$› *55*)
  | *Box* $'i$ ‹$('i, 'p)$ *fm*› (‹$\Box$›)

abbreviation *Neg* (‹$\neg$ -› [*70*] *70*) **where**
  ‹$\neg\ p \equiv p \longrightarrow \bot$›

6.2 Semantics

datatype $('i, 'p, 'w)$ *model* =
  Model (*W*: ‹$'w$ *set*›) (*R*: ‹$'i \Rightarrow 'w \Rightarrow 'w$ *set*›) (*V*: ‹$'w \Rightarrow 'p \Rightarrow$ *bool*›)

type-synonym $('i, 'p, 'w)$ *ctx* = ‹$('i, 'p, 'w)$ *model* $\times$ $'w$›

fun *semantics* :: ‹$('i, 'p, 'w)$ *ctx* $\Rightarrow$ $('i, 'p)$ *fm* $\Rightarrow$ *bool*› (**infix** ‹$\models_\Box$› *50*) **where**
  ‹- $\models_\Box \bot \longleftrightarrow$ *False*›
| ‹$(M, w) \models_\Box \bullet P \longleftrightarrow V\ M\ w\ P$›
| ‹$(M, w) \models_\Box p \longrightarrow q \longleftrightarrow (M, w) \models_\Box p \longrightarrow (M, w) \models_\Box q$›
| ‹$(M, w) \models_\Box \Box\ i\ p \longleftrightarrow (\forall v \in W\ M \cap R\ M\ i\ w.\ (M, v) \models_\Box p)$›

6.3 Calculus

primrec *eval* :: ‹$('p \Rightarrow$ *bool*$) \Rightarrow (('i, 'p)$ *fm* $\Rightarrow$ *bool*$) \Rightarrow ('i, 'p)$ *fm* $\Rightarrow$ *bool*› **where**
  ‹*eval* - - $\bot$ = *False*›
| ‹*eval* *g* - $(\bullet P) = g\ P$›
| ‹*eval* *g* *h* $(p \longrightarrow q) =$ (*eval* *g* *h* *p* $\longrightarrow$ *eval* *g* *h* *q*)›
| ‹*eval* - *h* $(\Box\ i\ p) = h\ (\Box\ i\ p)$›

abbreviation ‹*tautology* *p* $\equiv \forall g\ h.$ *eval* *g* *h* *p*›

inductive *Calculus* :: ‹$('i, 'p)$ *fm* $\Rightarrow$ *bool*› (‹$\vdash_\Box$ -› [*50*] *50*) **where**
  A1: ‹*tautology* *p* $\Longrightarrow \vdash_\Box p$›
| *A2*: ‹$\vdash_\Box \Box\ i\ (p \longrightarrow q) \longrightarrow \Box\ i\ p \longrightarrow \Box\ i\ q$›

| *R1*: ‹$\vdash_\Box p \Longrightarrow \vdash_\Box p \longrightarrow q \Longrightarrow \vdash_\Box q$›
| *R2*: ‹$\vdash_\Box p \Longrightarrow \vdash_\Box \Box\, i\, p$›

primrec *imply* :: ‹$('i, 'p)\ fm\ list \Rightarrow ('i, 'p)\ fm \Rightarrow ('i, 'p)\ fm$› (**infixr** ‹$\rightsquigarrow$› *56*) **where**
‹$([] \rightsquigarrow p) = p$›
| ‹$(q \# A \rightsquigarrow p) = (q \longrightarrow A \rightsquigarrow p)$›

abbreviation *Calculus-assms* (**infix** ‹$\vdash_\Box$› *50*) **where**
‹$A \vdash_\Box p \equiv \vdash_\Box A \rightsquigarrow p$›

6.4 Soundness

lemma *eval-semantics*: ‹$eval\ (g\ w)\ (\lambda q.\ (Model\ Ws\ r\ g,\ w) \models_\Box q)\ p = ((Model\ Ws\ r\ g,\ w) \models_\Box p)$›
⟨*proof*⟩

lemma *tautology*:
assumes ‹$tautology\ p$›
shows ‹$(M,\ w) \models_\Box p$›
⟨*proof*⟩

theorem *soundness*: ‹$\vdash_\Box p \Longrightarrow w \in W\ M \Longrightarrow (M,\ w) \models_\Box p$›
⟨*proof*⟩

6.5 Admissible rules

lemma *K-imply-head*: ‹$p \# A \vdash_\Box p$›
⟨*proof*⟩

lemma *K-imply-Cons*:
assumes ‹$A \vdash_\Box q$›
shows ‹$p \# A \vdash_\Box q$›
⟨*proof*⟩

lemma *K-right-mp*:
assumes ‹$A \vdash_\Box p$› ‹$A \vdash_\Box p \longrightarrow q$›
shows ‹$A \vdash_\Box q$›
⟨*proof*⟩

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

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

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

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

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

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

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

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

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

6.6 Maximal Consistent Sets

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

interpretation *MCS-No-Witness-UNIV consistent*
⟨*proof*⟩

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

interpretation *Derivations-Bot consistent Calculus-assms* ‹$\bot$›
⟨*proof*⟩

interpretation *Derivations-Imp consistent Calculus-assms* ‹$\lambda p\ q.\ p \longrightarrow q$›
⟨*proof*⟩

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

6.7 Truth Lemma

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

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

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

fun *semics* ::
‹$('i,\ 'p,\ 'w)\ ctx \Rightarrow (('i,\ 'p,\ 'w)\ ctx \Rightarrow ('i,\ 'p)\ fm \Rightarrow bool) \Rightarrow ('i,\ 'p)\ fm \Rightarrow bool$›
(‹(- ⟦-⟧$_\Box$ -)› [55, 0, 55] 55) **where**
‹- ⟦-⟧$_\Box$ $\bot \longleftrightarrow$ *False*›
| ‹$(M,\ w)$ ⟦-⟧$_\Box$ $\cdot P \longleftrightarrow V\ M\ w\ P$›
| ‹$(M,\ w)$ ⟦$\mathcal{R}$⟧$_\Box$ $p \longrightarrow q \longleftrightarrow \mathcal{R}\ (M,\ w)\ p \longrightarrow \mathcal{R}\ (M,\ w)\ q$›
| ‹$(M,\ w)$ ⟦$\mathcal{R}$⟧$_\Box$ $\Box\ i\ p \longleftrightarrow (\forall v \in W\ M \cap R\ M\ i\ w.\ \mathcal{R}\ (M,\ v)\ p)$›

fun *rel* :: ‹('*i*, '*p*) *fm set* ⇒ ('*i*, '*p*, ('*i*, '*p*) *fm set*) *ctx* ⇒ ('*i*, '*p*) *fm* ⇒ *bool*› (‹$\mathcal{R}_\Box$›) **where**
‹$\mathcal{R}_\Box$(-) (-, *w*) *p* ⟷ *p* ∈ *w*›

theorem *saturated-model*:
fixes *S* :: ‹('*i*, '*p*) *fm set*›
assumes ‹$\bigwedge$(*S* :: ('*i*, '*p*) *fm set*) *p*. *MCS S* ⟹ $\mathcal{M}_\Box(S)$ $[\![\mathcal{R}_\Box(S')]\!]_\Box$ *p* ⟷ *p* ∈ *S*›
shows ‹*MCS S* ⟹ $\mathcal{M}_\Box(S)$ $\models_\Box$ *p* ⟷ *p* ∈ *S*›
⟨*proof*⟩

theorem *saturated-MCS*:
assumes ‹*MCS S*›
shows ‹$\mathcal{M}_\Box(S)$ $[\![\mathcal{R}_\Box(S')]\!]_\Box$ *p* ⟷ $\mathcal{R}_\Box(S')$ ($\mathcal{M}_\Box(S)$) *p*›
⟨*proof*⟩

interpretation *Truth-No-Witness semics semantics* ‹λ-. {$\mathcal{M}_\Box(S)$ |*S*. *MCS S*}› *rel consistent*
⟨*proof*⟩

lemma *Truth-lemma*:
assumes ‹*MCS S*›
shows ‹$\mathcal{M}_\Box(S)$ $\models_\Box$ *p* ⟷ *p* ∈ *S*›
⟨*proof*⟩

6.8 Completeness

theorem *strong-completeness*:
assumes ‹∀ *M* :: ('*i*, '*p*, ('*i*, '*p*) *fm set*) *model*. ∀ *w* ∈ *W M*.
(∀ *q* ∈ *X*. (*M*, *w*) $\models_\Box$ *q*) ⟶ (*M*, *w*) $\models_\Box$ *p*›
shows ‹∃ *A*. *set A* ⊆ *X* ∧ *A* $\vdash_\Box$ *p*›
⟨*proof*⟩

abbreviation *valid* :: ‹('*i*, '*p*) *fm* ⇒ *bool*› **where**
‹*valid p* ≡ ∀ (*M* :: ('*i*, '*p*, ('*i*, '*p*) *fm set*) *model*). ∀ *w* ∈ *W M*. (*M*, *w*) $\models_\Box$ *p*›

corollary *completeness*: ‹*valid p* ⟹ $\vdash_\Box$ *p*›
⟨*proof*⟩

theorem *main*: ‹*valid p* ⟷ $\vdash_\Box$ *p*›
⟨*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*› (‹@›)
| *All* ‹(*'i*, *'p*) *fm*› (‹**A**›)

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

abbreviation *Con* (**infixr** ‹∧› *35*) **where** ‹*p* ∧ *q* ≡ ¬ (*p* ⟶ ¬ *q*)›

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

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

7.2 Semantics

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

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

fun *semantics* :: ‹(*'i*, *'p*, *'w*) *ctx* ⇒ (*'i*, *'p*) *fm* ⇒ *bool*› (**infix** ‹⊨₍@₎› *50*) **where**
‹(*M*, *g*, *w*) ⊨₍@₎ ⊥ ⟷ *False*›

| ‹(M, -, w) ⊨@ •P ⟷ V M w P›
| ‹(-, g, w) ⊨@ •i ⟷ w = g i›
| ‹(M, g, w) ⊨@ p ⟶ q ⟷ (M, g, w) ⊨@ p ⟶ (M, g, w) ⊨@ q›
| ‹(M, g, w) ⊨@ ◇ p ⟷ (∃ v ∈ W M ∩ R M w. (M, g, v) ⊨@ p)›
| ‹(M, g, -) ⊨@ @i p ⟷ (M, g, g i) ⊨@ p›
| ‹(M, g, -) ⊨@ A p ⟷ (∀ v ∈ W M. (M, g, v) ⊨@ p)›

lemma *semantics-fresh*: ‹i ∉ nominals-fm p ⟹ (M, g, w) ⊨@ p ⟷ (M, g(i := v), w) ⊨@ p›
⟨*proof*⟩

lemma *semantics-fresh-lbd*:
‹k ∉ nominals-lbd (i, p) ⟹ (M, g, w) ⊨@ p ⟷ (M, g(k := v), w) ⊨@ p›
⟨*proof*⟩

7.3 Calculus

inductive *Calculus* :: ‹('i, 'p) lbd list ⇒ ('i, 'p) lbd ⇒ bool› (**infix** ‹⊢@› 50) **where**
Assm [*simp*]: ‹(i, p) ∈ set A ⟹ A ⊢@ (i, p)›
| *Ref* [*simp*]: ‹A ⊢@ (i, •i)›
| *Nom* [*dest*]: ‹A ⊢@ (i, •k) ⟹ A ⊢@ (i, p) ⟹ A ⊢@ (k, p)›
| *FlsE* [*elim*]: ‹A ⊢@ (i, ⊥) ⟹ A ⊢@ (k, p)›
| *ImpI* [*intro*]: ‹(i, p) # A ⊢@ (i, q) ⟹ A ⊢@ (i, p ⟶ q)›
| *ImpE* [*dest*]: ‹A ⊢@ (i, p ⟶ q) ⟹ A ⊢@ (i, p) ⟹ A ⊢@ (i, q)›
| *SatI* [*intro*]: ‹A ⊢@ (i, p) ⟹ A ⊢@ (k, @i p)›
| *SatE* [*dest*]: ‹A ⊢@ (i, @k p) ⟹ A ⊢@ (k, p)›
| *DiaI* [*intro*]: ‹A ⊢@ (i, ◇ (•k)) ⟹ A ⊢@ (k, p) ⟹ A ⊢@ (i, ◇ p)›
| *DiaE* [*elim*]: ‹A ⊢@ (i, ◇ p) ⟹ k ∉ nominals ({(i, p), (j, q)} ∪ set A) ⟹
(k, p) # (i, ◇ (•k)) # A ⊢@ (j, q) ⟹ A ⊢@ (j, q)›
| *AllI* [*intro*]: ‹A ⊢@ (k, p) ⟹ k ∉ nominals ({(i, p)} ∪ set A) ⟹ A ⊢@ (i, A p)›
| *AllE* [*dest*]: ‹A ⊢@ (i, A p) ⟹ A ⊢@ (k, p)›
| *Clas*: ‹(i, p ⟶ q) # A ⊢@ (i, p) ⟹ A ⊢@ (i, p)›
| *Cut*: ‹A ⊢@ (k, q) ⟹ (k, q) # B ⊢@ (i, p) ⟹ A @ B ⊢@ (i, p)›

7.4 Soundness

theorem *soundness*: ‹A ⊢@ (i, p) ⟹ list-all (λ(i, p). (M, g, g i) ⊨@ p) A ⟹ range g ⊆ W M ⟹
(M, g, g i) ⊨@ p›
⟨*proof*⟩

corollary *soundness'*:
assumes ‹[] ⊢@ (i, p)› ‹i ∉ nominals-fm p›
and ‹range g ⊆ W M› ‹w ∈ W M›
shows ‹(M, g, w) ⊨@ p›
⟨*proof*⟩

corollary ‹¬ ([] ⊢@ (i, ⊥))›
⟨*proof*⟩

7.5 Admissible Rules

lemma *Assm-head* [*simp*]: ‹(p, i) # A ⊢@ (p, i)›
⟨*proof*⟩

lemma *SatE'*:

assumes ‹$(k,\ q)\ \#\ A \vdash_@ (i,\ p)$›
shows ‹$(j,\ @k\ q)\ \#\ A \vdash_@ (i,\ p)$›
⟨*proof*⟩

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

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

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

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

interpretation *Derivations Calculus*
⟨*proof*⟩

7.6 Maximal Consistent Sets

definition *consistent* :: ‹$('i,\ 'p)$ *lbd set* ⇒ *bool*› **where**
‹*consistent* $S \equiv \forall A\ a.\ set\ A \subseteq S \longrightarrow \neg\ A \vdash_@ (a,\ \bot)$›

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

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

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

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-Witness-UNIV consistent witness nominals-lbd*
⟨*proof*⟩

lemma *witnessed-diamond*: ‹*witnessed* $S \Longrightarrow (i, \Diamond\ p) \in S \Longrightarrow \exists\, k.\ (i, \Diamond\ (\cdot k)) \in S \wedge (k, p) \in S$›
⟨*proof*⟩

lemma *witnessed-global*: ‹*witnessed* $S \Longrightarrow (i, \neg\ \mathbf{A}\ p) \in S \Longrightarrow \exists\, k.\ (k, \neg\ p) \in S$›
⟨*proof*⟩

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

interpretation *Derivations-Bot consistent Calculus* ‹$(i, \bot)$›
⟨*proof*⟩

interpretation *Derivations-Not consistent Calculus* ‹$(i, \bot)$› ‹$\lambda(i, p).\ (i, \neg\ p)$›
⟨*proof*⟩

lemma *MCS-impE′*: ‹*consistent* $S \Longrightarrow$ *maximal* $S \Longrightarrow (i, p \longrightarrow q) \in S \Longrightarrow (i, p) \in S \longrightarrow (i, q) \in S$›
⟨*proof*⟩

interpretation *Derivations-Uni consistent witness nominals-lbd Calculus* ‹$(i, \bot)$› ‹$\lambda(i, p).\ (i, \neg\ p)$›
‹$\lambda(i, p).\ (i, \mathbf{A}\ p)$› ‹$\lambda k\ (i, p).\ (k, p)$›
⟨*proof*⟩

lemma *conE1* [*elim*]: ‹$A \vdash_@ (i, p \wedge q) \Longrightarrow A \vdash_@ (i, p)$›
⟨*proof*⟩

lemma *conE2* [*elim*]: ‹$A \vdash_@ (i, p \wedge q) \Longrightarrow A \vdash_@ (i, q)$›
⟨*proof*⟩

lemma *conI* [*intro*]: ‹$A \vdash_@ (i, p) \Longrightarrow A \vdash_@ (i, q) \Longrightarrow A \vdash_@ (i, p \wedge q)$›
⟨*proof*⟩

lemma *MCS-con*:
assumes ‹*MCS* S›
shows ‹$(i, p \wedge q) \in S \longleftrightarrow (i, p) \in S \wedge (i, q) \in S$›
⟨*proof*⟩

interpretation *Derivations-Exi consistent witness nominals-lbd Calculus*
‹$\lambda(i, p).\ (i, \Diamond\ p)$› ‹$\lambda k\ (i, p).\ (i, @k\ p \wedge \Diamond(\cdot k))$›
⟨*proof*⟩

corollary *MCS-uni′*:
assumes ‹*MCS* S› ‹*witnessed* S›
shows ‹$(i, \mathbf{A}\ p) \in S \longleftrightarrow (\forall\, k.\ (k, p) \in S)$›
⟨*proof*⟩

corollary *MCS-exi′*:
assumes ‹*MCS* S› ‹*witnessed* S›
shows ‹$(i, \Diamond\ p) \in S \longleftrightarrow (\exists\, k.\ (i, @k\ p \wedge \Diamond\ (\cdot k)) \in S)$›
⟨*proof*⟩

7.7 Nominals

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

⟨*proof*⟩

lemma *MCS-Nom-sym*:
assumes ‹*consistent S*› ‹*maximal S*› ‹(*i*, ·*k*) ∈ *S*›
shows ‹(*k*, ·*i*) ∈ *S*›
⟨*proof*⟩

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

7.8 Truth Lemma

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

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

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

lemma *equiv-nom-reflp*:
assumes ‹*consistent S*› ‹*maximal S*›
shows ‹*reflp* (*equiv-nom S*)›
⟨*proof*⟩

lemma *equiv-nom-symp*:
assumes ‹*consistent S*› ‹*maximal S*›
shows ‹*symp* (*equiv-nom S*)›
⟨*proof*⟩

lemma *equiv-nom-transp*:
assumes ‹*consistent S*› ‹*maximal S*›
shows ‹*transp* (*equiv-nom S*)›
⟨*proof*⟩

lemma *equiv-nom-equivp*:
assumes ‹*consistent S*› ‹*maximal S*›
shows ‹*equivp* (*equiv-nom S*)›
⟨*proof*⟩

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

lemma *equiv-nom-ne*:
assumes ‹*consistent S*› ‹*maximal S*›

shows ‹$\{k.\ equiv\text{-}nom\ S\ i\ k\} \neq \{\}$›
⟨*proof*⟩

lemma *equiv-nom-assign*:
assumes ‹*consistent S*› ‹*maximal S*›
shows ‹*equiv-nom S i* $([i]_S)$›
⟨*proof*⟩

lemma *equiv-nom-Nom*:
assumes ‹*consistent S*› ‹*maximal S*› ‹*equiv-nom S i k*› ‹$(i,\ p) \in S$›
shows ‹$(k,\ p) \in S$›
⟨*proof*⟩

definition *reach* :: ‹('*i*, '*p*) *lbd set* ⇒ '*i* ⇒ '*i set*› **where**
‹*reach S i* ≡ $\{[k]_S\ |k.\ (i, \Diamond\ (\cdot k)) \in S\}$›

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

theorem *saturated-model*:
assumes ‹$\bigwedge p\ i.\ \mathcal{M}_@(S,\ i)\ [\![\mathcal{R}_@(S)]\!]_@(p) \longleftrightarrow \mathcal{R}_@(S)\ (\mathcal{M}_@(S,\ i))\ p$› ‹$M \in \{\mathcal{M}_@(S,\ i)\ |i.\ True\}$›
shows ‹$\mathcal{R}_@(S)\ (\mathcal{M}_@(S,\ i))\ p \longleftrightarrow \mathcal{M}_@(S,\ i) \models_@ p$›
⟨*proof*⟩

lemma *reach-assign*: ‹*reach S* $([i]_S) \subseteq \{[k]_S \mid k.\ True\}$›
⟨*proof*⟩

theorem *saturated-MCS*:
assumes ‹*MCS S*›
shows ‹$\mathcal{M}_@(S,\ i)\ [\![\mathcal{R}_@(S)]\!]_@(p) \longleftrightarrow \mathcal{R}_@(S)\ (\mathcal{M}_@(S,\ i))\ p$›
⟨*proof*⟩

interpretation *Truth-Witness semics semantics* ‹$\lambda S.\ \{\mathcal{M}_@(S,\ i)\ |i.\ True\}$› *rel consistent witness nominals-lbd*
⟨*proof*⟩

lemma *Truth-lemma*:
assumes ‹*MCS S*›
shows ‹$\mathcal{M}_@(S,\ i) \models_@ p \longleftrightarrow (i,\ p) \in S$›
⟨*proof*⟩

7.9 Cardinalities

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

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*)›
abbreviation ‹*GLO* ≡ *Inr* (*AllM*, *0*)›

primrec *encode* :: ‹('*i*, '*p*) *fm* ⇒ ('*i*, '*p*) *enc list*› **where**
  ‹*encode* ⊥ = [*FLS*]›
| ‹*encode* (·*P*) = [*PRO P*]›
| ‹*encode* (·*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*›
| ‹*encode* (**A** *p*) = *GLO* # *encode p*›

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

lemma *inj-encode'*: ‹*encode p* = *encode q* ⟹ *p* = *q*›
⟨*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* ∈ *W M*. *range g* ⊆ *W M* ⟶
      (∀ (*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* ∈ *W M*. *range g* ⊆ *W M* ⟶ (*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 ‹[] $\vdash_{@}$ $(i,\ p)$›
⟨*proof*⟩

theorem *main*:
fixes p :: ‹$('i,\ 'p)$ *fm*›
assumes ‹$i \notin$ *nominals-fm* p› ‹*infinite* $(UNIV :: 'i\ set)$› ‹$|UNIV :: 'p\ set| \leq o\ |UNIV :: 'i\ set|$›
shows ‹*valid* $p \longleftrightarrow$ [] $\vdash_{@}$ $(i,\ p)$›
⟨*proof*⟩

corollary *main'*:
fixes p :: ‹$('i,\ 'i)$ *fm*›
assumes ‹$i \notin$ *nominals-fm* p› ‹*infinite* $(UNIV :: 'i\ set)$›
shows ‹*valid* $p \longleftrightarrow$ [] $\vdash_{@}$ $(i,\ p)$›
⟨*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*)
  | *Exi* ‹(*'f*, *'p*) *fm*› (‹∃›)

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

8.2 Semantics

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

fun *semantics-tm* :: ‹(*nat* ⇒ *'a*) × (*'f* ⇒ *'a list* ⇒ *'a*) ⇒ *'f tm* ⇒ *'a*› (‹⦇-⦈›) **where**
  ‹⦇(*E*, -)⦈ (#*n*) = *E n*›
| ‹⦇(*E*, *F*)⦈ (•*f ts*) = *F f* (*map* ⦇(*E*, *F*)⦈ *ts*)›

primrec *add-env* :: ‹*'a* ⇒ (*nat* ⇒ *'a*) ⇒ *nat* ⇒ *'a*› (**infix** ‹⨟› *0*) **where**
  ‹(*t* ⨟ *s*) *0* = *t*›
| ‹(*t* ⨟ *s*) (*Suc n*) = *s n*›

fun *semantics-fm* :: ‹(*'a*, *'f*, *'p*) *model* ⇒ (*'f*, *'p*) *fm* ⇒ *bool*› (**infix** ‹⊨∃› *50*) **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* ⨟ λ*n*. *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* ⨟ λ*n*. *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*) = ⦇(λ*n*. ⦇(*E*, *F*)⦈ (*s n*), *F*)⦈ *t*›
⟨*proof*⟩

lemma *sub-fm-semantics* [*simp*]: ‹(*E*, *F*, *G*) ⊨∃ *sub-fm s p* ⟷ (λ*n*. ⦇(*E*, *F*)⦈ (*s n*), *F*, *G*) ⊨∃ *p*›
⟨*proof*⟩

lemma *sub-tm-Var* [*simp*]: ‹*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*› (**infix** ‹⊢∃› *50*) **where**
Assm [*simp*]: ‹*p* ∈ *set A* ⟹ *A* ⊢∃ *p*›
| *FlsE* [*elim*]: ‹*A* ⊢∃ ⊥ ⟹ *A* ⊢∃ *p*›
| *ImpI* [*intro*]: ‹*p* # *A* ⊢∃ *q* ⟹ *A* ⊢∃ *p* ⟶ *q*›
| *ImpE* [*dest*]: ‹*A* ⊢∃ *p* ⟶ *q* ⟹ *A* ⊢∃ *p* ⟹ *A* ⊢∃ *q*›
| *ExiI* [*intro*]: ‹*A* ⊢∃ ⟨*t*⟩*p* ⟹ *A* ⊢∃ ∃ *p*›
| *ExiE* [*elim*]: ‹*A* ⊢∃ ∃ *p* ⟹ *a* ∉ *params* (*set* (*p* # *q* # *A*)) ⟹ ⟨⋆*a*⟩*p* # *A* ⊢∃ *q* ⟹ *A* ⊢∃ *q*›
| *Clas*: ‹(*p* ⟶ *q*) # *A* ⊢∃ *p* ⟹ *A* ⊢∃ *p*›

8.4.1 Weakening

abbreviation ‹*psub f* ≡ *map-fm f id*›

lemma *map-tm-sub-tm* [*simp*]: ‹*map-tm f* (*sub-tm g t*) = *sub-tm* (*map-tm f o g*) (*map-tm f t*)›
⟨*proof*⟩

lemma *map-tm-lift-tm* [*simp*]: ‹*map-tm f* (*lift-tm t*) = *lift-tm* (*map-tm f t*)›
⟨*proof*⟩

lemma *psub-sub-fm*: ‹*psub f* (*sub-fm g p*) = *sub-fm* (*map-tm f o g*) (*psub f p*)›
⟨*proof*⟩

lemma *map-tm-inst-single*: ‹(*map-tm f o* (*u* ⨟ #)) *t* = (*map-tm f u* ⨟ #) *t*›
⟨*proof*⟩

lemma *psub-inst-single* [*simp*]: ‹*psub f* (⟨*t*⟩*p*) = ⟨*map-tm f t*⟩(*psub f p*)›
⟨*proof*⟩

lemma *map-tm-upd* [*simp*]: ‹*a* ∉ *params-tm t* ⟹ *map-tm* (*f*(*a* := *b*)) *t* = *map-tm f t*›
⟨*proof*⟩

lemma *psub-upd* [*simp*]: ‹*a* ∉ *params-fm p* ⟹ *psub* (*f*(*a* := *b*)) *p* = *psub f p*›
⟨*proof*⟩

class *inf-univ* =
fixes *itself* :: ‹'*a itself*›
assumes *infinite-UNIV*: ‹*infinite* (*UNIV* :: '*a set*)›

lemma *Calculus-psub*:
fixes $f :: \langle 'f \Rightarrow 'g :: \textit{inf-univ} \rangle$
shows $\langle A \vdash_\exists p \Longrightarrow \textit{map} (\textit{psub } f)\ A \vdash_\exists \textit{psub } f\ p \rangle$
⟨*proof*⟩

lemma *Weaken*:
fixes $p :: \langle ('f :: \textit{inf-univ}, 'p)\ \textit{fm} \rangle$
shows $\langle A \vdash_\exists p \Longrightarrow \textit{set } A \subseteq \textit{set } B \Longrightarrow B \vdash_\exists p \rangle$
⟨*proof*⟩

8.5 Soundness

theorem *soundness*: $\langle A \vdash_\exists p \Longrightarrow \forall q \in \textit{set } A.\ (E, F, G) \models_\exists q \Longrightarrow (E, F, G) \models_\exists p \rangle$
⟨*proof*⟩

corollary *soundness'*: $\langle [] \vdash_\exists p \Longrightarrow M \models_\exists p \rangle$
⟨*proof*⟩

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

8.6 Admissible Rules

lemma *Assm-head*: $\langle p \# A \vdash_\exists p \rangle$
⟨*proof*⟩

lemma *Boole*: $\langle (\neg p) \# A \vdash_\exists \bot \Longrightarrow A \vdash_\exists p \rangle$
⟨*proof*⟩

corollary *Weak*:
fixes $p :: \langle ('f :: \textit{inf-univ}, 'p)\ \textit{fm} \rangle$
shows $\langle A \vdash_\exists p \Longrightarrow q \# A \vdash_\exists p \rangle$
⟨*proof*⟩

lemma *deduct1*:
fixes $p :: \langle ('f :: \textit{inf-univ}, 'p)\ \textit{fm} \rangle$
shows $\langle A \vdash_\exists p \longrightarrow q \Longrightarrow p \# A \vdash_\exists q \rangle$
⟨*proof*⟩

lemma *Weak'*:
fixes $p :: \langle ('f :: \textit{inf-univ}, 'p)\ \textit{fm} \rangle$
shows $\langle A \vdash_\exists p \Longrightarrow B\ @\ A \vdash_\exists p \rangle$
⟨*proof*⟩

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

8.7 Maximal Consistent Sets

definition *consistent* :: $\langle ('f, 'p)\ \textit{fm set} \Rightarrow \textit{bool} \rangle$ **where**
$\langle \textit{consistent } S \equiv \forall A.\ \textit{set } A \subseteq S \longrightarrow \neg\ A \vdash_\exists \bot \rangle$

fun *witness* :: $\langle ('f, 'p)\ \textit{fm} \Rightarrow ('f, 'p)\ \textit{fm set} \Rightarrow ('f, 'p)\ \textit{fm set} \rangle$ **where**
$\langle \textit{witness } (\exists\, p)\ S = (\textit{let } a = \textit{SOME } a.\ a \notin \textit{params } (\{p\} \cup S) \textit{ in } \{\langle \star a \rangle p\}) \rangle$

| ‹*witness* - - = {}›

lemma *consistent-add-witness*:
fixes p :: ‹('*f* :: *inf-univ*, '*p*) *fm*›
assumes ‹*consistent S*› ‹$\exists\, p \in S$› ‹$a \notin$ *params S*›
shows ‹*consistent* ($\{\langle\star a\rangle p\} \cup S$)›
⟨*proof*⟩

lemma *consistent-witness'*:
fixes p :: ‹('*f* :: *inf-univ*, '*p*) *fm*›
assumes ‹*consistent* ($\{p\} \cup S$)› ‹*infinite* (*UNIV* − *params S*)›
shows ‹*consistent* (*witness p S* $\cup$ $\{p\} \cup S$)›
⟨*proof*⟩

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

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

interpretation *Derivations-Bot consistent Calculus* ‹$\bot$ :: ('*f* :: *inf-univ*, '*p*) *fm*›
⟨*proof*⟩

interpretation *Derivations-Imp consistent Calculus* ‹$\lambda p\ q.\ p \longrightarrow q$ :: ('*f* :: *inf-univ*, '*p*) *fm*›
⟨*proof*⟩

interpretation *Derivations-Exi consistent witness params-fm Calculus* ‹$\exists$ › ‹$\lambda t\ p.\ \langle t\rangle p$ :: ('*f* :: *inf-univ*, '*p*) *fm*›
⟨*proof*⟩

8.8 Truth Lemma

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

fun *semics* ::
‹('*a*, '*f*, '*p*) *model* ⇒ (('*a*, '*f*, '*p*) *model* ⇒ ('*f*, '*p*) *fm* ⇒ *bool*) ⇒ ('*f*, '*p*) *fm* ⇒ *bool*›
(‹(- $[\![$-$]\!]_\exists$ -)› [*55*, *0*, *55*] *55*) **where**
‹- $[\![$-$]\!]_\exists$ $\bot \longleftrightarrow$ *False*›
| ‹(*E*, *F*, *G*) $[\![$-$]\!]_\exists$ $\cdot P\ ts \longleftrightarrow$ *G P* (*map* $(\!|(E, F)|\!)$ *ts*)›
| ‹(*E*, *F*, *G*) $[\![\mathcal{R}]\!]_\exists$ $p \longrightarrow q \longleftrightarrow \mathcal{R}$ (*E*, *F*, *G*) $p \longrightarrow \mathcal{R}$ (*E*, *F*, *G*) q›
| ‹(*E*, *F*, *G*) $[\![\mathcal{R}]\!]_\exists$ $\exists\, p \longleftrightarrow$ ($\exists\, x.\ \mathcal{R}$ (x ⨟ *E*, *F*, *G*) p)›

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

theorem *saturated-model*:
assumes ‹$\bigwedge p.\ \forall M \in \{\mathcal{M}_\exists\,(S)\}.\ M\ [\![\mathcal{R}_\exists\,(S)]\!]_\exists\ p \longleftrightarrow \mathcal{R}_\exists\,(S)\ M\ p$› ‹$M \in \{\mathcal{M}_\exists\,(S)\}$›
shows ‹$\mathcal{R}_\exists\,(S)\ M\ p \longleftrightarrow M \models_\exists p$›
⟨*proof*⟩

theorem *saturated-MCS*:
fixes p :: ‹('*f* :: *inf-univ*, '*p*) *fm*›
assumes ‹*MCS S*›
shows ‹$\mathcal{R}_\exists\,(S)$ ($\mathcal{M}_\exists\,(S)$) $p \longleftrightarrow \mathcal{M}_\exists\,(S)$ $[\![\mathcal{R}_\exists\,(S)]\!]_\exists$ p›

⟨*proof*⟩

interpretation *Truth-Witness semics semantics-fm* ‹$\lambda S.\ \{\mathcal{M}_{\exists}(S)\}$› *rel consistent witness*
‹*params-fm* :: (*'f* :: *inf-univ*, *'p*) *fm* ⇒ -›
⟨*proof*⟩

8.9 Cardinalities

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

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 ‹*EXI* ≡ *Inr* (*ExiM*, *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*) = *EXI* # *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*:
‹|*UNIV* :: (′*f* :: *inf-univ*, ′*p*) *fm set*| ≤*o* |*UNIV* :: ′*f set*| +*c* |*UNIV* :: ′*p set*|›
⟨*proof*⟩

8.10 Completeness

theorem *strong-completeness*:
assumes ‹∀ *M* :: (′*f tm*, ′*f* :: *inf-univ*, ′*p*) *model*. (∀ *q* ∈ *X*. *M* ⊨∃ *q*) ⟶ *M* ⊨∃ *p*›
‹|*UNIV* :: ′*f set*| +*c* |*UNIV* :: ′*p set*| ≤*o* |*UNIV* − *params X*|›
shows ‹∃ *A*. *set A* ⊆ *X* ∧ *A* ⊢∃ *p*›
⟨*proof*⟩

abbreviation *valid* :: ‹(′*f*, ′*p*) *fm* ⇒ *bool*› **where**
‹*valid p* ≡ ∀ *M* :: (′*f tm*, -, -) *model*. *M* ⊨∃ *p*›

theorem *completeness*:
fixes *p* :: ‹(′*f* :: *inf-univ*, ′*p*) *fm*›
assumes ‹*valid p*› ‹|*UNIV* :: ′*p set*| ≤*o* |*UNIV* :: ′*f set*|›
shows ‹[] ⊢∃ *p*›
⟨*proof*⟩

corollary *completeness′*:
fixes *p* :: ‹(′*f* :: *inf-univ*, ′*f*) *fm*›
assumes ‹*valid p*›
shows ‹[] ⊢∃ *p*›
⟨*proof*⟩

theorem *main*:
fixes *p* :: ‹(′*f* :: *inf-univ*, ′*p*) *fm*›
assumes ‹|*UNIV* :: ′*p set*| ≤*o* |*UNIV* :: ′*f set*|›
shows ‹*valid p* ⟷ [] ⊢∃ *p*›
⟨*proof*⟩

corollary *main′*:
fixes *p* :: ‹(′*f* :: *inf-univ*, ′*f*) *fm*›
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.