

# Abstract Consistency Properties

Asta Halkjær Boserup, Anders Schlichtkrull

May 8, 2026

## Abstract

Smullyan used abstract consistency properties to great effect in unifying meta-theoretical results in first-order logic. By abstractly specifying when a set of formulas is consistent, he proved a single model existence result for all sets that met the specification, reusing this result in proving completeness, compactness, etc. Fitting later defined abstract consistency properties for a range of other logics.

In this work we use locales to mechanize abstract consistency properties without fixing a particular logic or syntax, generalizing the work of Smullyan and Fitting. We use Fitting's technique for closing a consistency property under limits to guarantee the existence of a maximal element. This yields a maximal consistent set for any notion of consistency expressible in the framework. The usual conjunctive, disjunctive, universal and existential conditions of abstract consistency properties —based on Smullyan's uniform notation— arise as special cases of our abstract development. Users of the framework can define an abstract consistency property for their own syntax and logic, prove that it is well behaved, and receive a corresponding maximal consistent set from which to prove model existence.

We provide three example instantiations. First, compactness and completeness for a first-order logic where we only instantiate universal quantifiers with already occurring terms. Second, completeness over general models for a second-order logic. Third, completeness of our own natural deduction system for Prior's Ideal Language, a recently developed hybrid logic with propositional quantification.

# Bibliography

- [1] S. Berghofer. First-order logic according to Fitting. *Archive of Formal Proofs*, 2007, August 2007. Formal proof development.
- [2] P. Blackburn, T. Braüner, and J. L. Kofod. Prior’s ideal language. *Mathematical Structures in Computer Science*, 35, 2025.
- [3] M. Fitting. *First-Order Logic and Automated Theorem Proving, Second Edition*. Graduate Texts in Computer Science. Springer, 1996.
- [4] A. H. From. Soundness and completeness of an axiomatic system for first-order logic. *Archive of Formal Proofs*, 2021, September 2021. Formal proof development.
- [5] A. H. From. A succinct formalization of the completeness of first-order logic. In H. Basold, J. Cockx, and S. Ghilezan, editors, *27th International Conference on Types for Proofs and Programs, TYPES 2021, June 14-18, 2021, Leiden, The Netherlands (Virtual Conference)*, volume 239 of *LIPICs*, pages 8:1–8:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
- [6] A. H. From and A. Schlichtkrull. Abstract, Compositional Consistency: Isabelle/HOL Locales for Completeness à la Fitting. In Y. Forster and C. Keller, editors, *16th International Conference on Interactive Theorem Proving (ITP 2025)*, volume 352 of *Leibniz International Proceedings in Informatics (LIPICs)*, pages 8:1–8:20, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
- [7] R. M. Smullyan. A unifying principal in quantification theory. *Proceedings of the National Academy of Sciences*, 49(6):828–832, 1963.
- [8] R. M. Smullyan. *First-Order Logic*. Springer, Berlin, 1968.
- [9] J. Väänänen. Second-order and Higher-order Logic. In E. N. Zalta and U. Nodelman, editors, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Winter 2024 edition, 2024.

# Contents

<b>1</b>	<b>Abstract Consistency Properties</b>	<b>5</b>
1.1	Utility . . . . .	5
1.2	Finite Character . . . . .	6
1.3	Consistency Properties . . . . .	7
1.3.1	Consistency Kinds . . . . .	9
1.4	Hintikka Sets . . . . .	14
1.5	Derivational Consistency . . . . .	14
1.6	Weak Derivational Consistency . . . . .	15
1.7	Conflicts . . . . .	16
1.8	Alpha . . . . .	17
1.9	Beta . . . . .	18
1.10	Gamma . . . . .	19
1.11	Delta . . . . .	21
1.12	Modal . . . . .	22
<b>2</b>	<b>Example: First-Order Logic with Restricted Instantiation</b>	<b>25</b>
2.1	Syntax . . . . .	25
2.2	Semantics . . . . .	25
2.3	Operations . . . . .	26
2.3.1	Lemmas . . . . .	27
2.4	Terms . . . . .	28
2.4.1	Lemmas . . . . .	28
2.5	Guard . . . . .	29
2.6	Model Existence . . . . .	29
2.7	Compactness . . . . .	31
2.8	Natural Deduction . . . . .	32
2.8.1	Soundness . . . . .	32
2.8.2	Derivational Consistency . . . . .	32
2.8.3	Strong Completeness . . . . .	33
2.8.4	Natural Deduction with Lists . . . . .	33
2.9	Tableau . . . . .	35
2.9.1	Soundness . . . . .	35
2.9.2	Derivational Consistency . . . . .	35

2.9.3	Strong Completeness . . . . .	35
<b>3</b>	<b>Example: Second-Order Logic</b>	<b>37</b>
3.1	Syntax . . . . .	37
3.2	Semantics . . . . .	38
3.3	Operations . . . . .	38
3.3.1	Shift . . . . .	38
3.3.2	Parameters . . . . .	39
3.3.3	Instantiation . . . . .	39
3.3.4	Size . . . . .	41
3.4	Model Existence . . . . .	42
3.5	Propositional Semantics . . . . .	45
3.6	Calculus . . . . .	46
3.7	Soundness . . . . .	46
3.8	Derived Rules . . . . .	47
3.9	Derivational Consistency . . . . .	48
3.10	Natural Deduction . . . . .	49
3.10.1	Soundness . . . . .	50
3.10.2	Derivational Consistency . . . . .	50
3.10.3	Strong Completeness . . . . .	51
<b>4</b>	<b>Example: Prior's Ideal Logic</b>	<b>52</b>
4.1	Syntax . . . . .	52
4.2	Semantics . . . . .	53
4.3	Operations . . . . .	55
4.3.1	Lemmas . . . . .	58
4.3.2	softqdf . . . . .	60
4.3.3	Add env . . . . .	60
4.3.4	Sizes . . . . .	61
4.4	Propositional Quantification . . . . .	61
4.5	Model Existence . . . . .	62
4.6	Natural Deduction . . . . .	68
4.6.1	Soundness . . . . .	69
4.6.2	Derived Rules . . . . .	69
4.6.3	Derivational Consistency . . . . .	69
4.6.4	Strong Completeness . . . . .	71
4.7	Natural Deduction with Lists . . . . .	71
4.8	The Need for SQDFs . . . . .	73
4.8.1	Finite Unions of Arithmetic Progressions . . . . .	73
4.8.2	Counterexample . . . . .	76

# Chapter 1

## Abstract Consistency Properties

```
theory Abstract-Consistency-Property imports  
  HOL-Cardinals.Cardinal-Order-Relation  
begin
```

### 1.1 Utility

```
lemma Set-Diff-Un:  $\langle X - (Y \cup Z) = X - Y - Z \rangle$   
   $\langle \text{proof} \rangle$ 
```

```
lemma infinite-diff-finite:  $\langle \text{finite } A \implies \text{infinite } (- B) \implies \text{infinite } (- (A \cup B)) \rangle$   
   $\langle \text{proof} \rangle$ 
```

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

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

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

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

**lemma** *infinite-left*:  $\langle \text{finite } C \implies \text{infinite } A \implies |A| \leq o \mid - B \implies |A| \leq o \mid - (C \cup B) \rangle$   
 $\langle \text{proof} \rangle$

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

**context** *wo-rel*  
**begin**

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

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

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

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

**definition** *is-chain* ::  $\langle 'a \Rightarrow 'a \text{ set} \Rightarrow \text{bool} \rangle$  **where**  
 $\langle \text{is-chain } f \equiv \forall a \in \text{Field } r. \forall b \in \text{Field } r. b \in \text{under } a \longrightarrow f b \subseteq f a \rangle$

**lemma** *is-chainD*:  $\langle \text{is-chain } f \implies b \in \text{under } a \implies x \in f b \implies x \in f a \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *chain-index*:  
**assumes** *ch*:  $\langle \text{is-chain } f \rangle$  **and** *fin*:  $\langle \text{finite } F \rangle$  **and** *ne*:  $\langle \text{Field } r \neq \{\} \rangle$   
**shows**  $\langle F \subseteq (\bigcup a \in \text{Field } r. f a) \implies \exists a \in \text{Field } r. F \subseteq f a \rangle$   
 $\langle \text{proof} \rangle$

**end**

## 1.2 Finite Character

**definition** *close* ::  $\langle 'a \text{ set set} \Rightarrow 'a \text{ set set} \rangle$  **where**  
 $\langle \text{close } C \equiv \{S. (\exists S' \in C. S \subseteq S')\} \rangle$

**definition** *subset-closed* ::  $\langle 'a \text{ set set} \Rightarrow \text{bool} \rangle$  **where**  
 $\langle \text{subset-closed } C \equiv \forall S' \in C. \forall S \subseteq S'. S \in C \rangle$

**lemma** *subset-in-close*:  $\langle S \subseteq S' \implies x \cup S' \in C \implies x \cup S \in \text{close } C \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *close-closed*:  $\langle \text{subset-closed } (\text{close } C) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *close-subset*:  $\langle C \subseteq \text{close } C \rangle$   
 $\langle \text{proof} \rangle$

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

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

**lemma** *finite-char*:  $\langle \text{finite-char } (\text{mk-finite-char } C) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *finite-char-closed*:  $\langle \text{finite-char } C \implies \text{subset-closed } C \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *finite-char-subset*:  $\langle \text{subset-closed } C \implies C \subseteq \text{mk-finite-char } C \rangle$   
 $\langle \text{proof} \rangle$

**lemma** (**in** *wo-rel*) *chain-union-closed*:  
**assumes**  $\langle \text{finite-char } C \rangle \langle \text{is-chain } f \rangle \langle \forall a \in \text{Field } r. f a \in C \rangle \langle \text{Field } r \neq \{\} \rangle$   
**shows**  $\langle (\bigcup a \in \text{Field } r. f a) \in C \rangle$   
 $\langle \text{proof} \rangle$

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

### 1.3 Consistency Properties

**locale** *Params* =  
**fixes** *map-fm* ::  $\langle ('x \Rightarrow 'x) \Rightarrow 'fm \Rightarrow 'fm \rangle$   
**and** *params-fm* ::  $\langle 'fm \Rightarrow 'x \text{ set} \rangle$   
**and** *is-param* ::  $\langle 'x \Rightarrow \text{bool} \rangle$   
**assumes** *map-fm-id*:  $\langle \text{map-fm } id = id \rangle$   
**and** *finite-params-fm* [*simp*]:  $\langle \bigwedge p. \text{finite } (\text{params-fm } p) \rangle$   
**and** *map-params-fm*:  $\langle \bigwedge f g p. (\forall x \in \text{params-fm } p. f x = g x) \implies \text{map-fm } f p = \text{map-fm } g p \rangle$   
**begin**

**definition** *is-subst* ::  $\langle ('x \Rightarrow 'x) \Rightarrow \text{bool} \rangle$  **where**  
 $\langle \text{is-subst } f \equiv \forall x. \text{is-param } x \longleftrightarrow \text{is-param } (f x) \rangle$

**lemma** *is-subst-id* [*intro*]:  $\langle \text{is-subst } id \rangle$

⟨proof⟩

**definition** *mk-alt-consistency* :: ⟨'fm set set ⇒ 'fm set set⟩ **where**  
⟨*mk-alt-consistency*  $C \equiv \{S. (\exists f. \text{is-subst } f \wedge \text{map-fm } f ' S \in C)\}$ ⟩

**lemma** *mk-alt-consistency-subset*: ⟨ $C \subseteq \text{mk-alt-consistency } C$ ⟩  
⟨proof⟩

**lemma** *mk-alt-consistency-closed*:  
**assumes** ⟨*subset-closed*  $C$ ⟩  
**shows** ⟨*subset-closed* (*mk-alt-consistency*  $C$ )⟩  
⟨proof⟩

**abbreviation** *params* :: ⟨'fm set ⇒ 'x set⟩ **where**  
⟨*params*  $S \equiv \bigcup p \in S. \text{params-fm } p$ ⟩

**lemma** *infinite-params*: ⟨*infinite* ( $U - \text{params } B$ ) ⇒ *infinite* ( $U - \text{params } (\text{set } ps \cup B)$ )⟩  
⟨proof⟩

**lemma** *infinite-params-left*:  
**assumes** ⟨*infinite*  $A$ ⟩ ⟨ $|A| \leq o \mid U - \text{params } S$ ⟩  
**shows** ⟨ $|A| \leq o \mid U - \text{params } (\text{set } ps \cup S)$ ⟩  
⟨proof⟩

**definition** *enough-new* :: ⟨'fm set ⇒ bool⟩ **where**  
⟨*enough-new*  $S \equiv \mid \text{UNIV} :: 'fm \text{ set} \mid \leq o \mid \text{Collect is-param} - \text{params } S$ ⟩

**lemma** *enough-new-countable*:  
**assumes** ⟨ $\exists \text{to-nat} :: 'fm \Rightarrow \text{nat. inj to-nat}$ ⟩ ⟨*infinite* (*Collect is-param* - *params*  $S$ )⟩  
**shows** ⟨*enough-new*  $S$ ⟩  
⟨proof⟩

**lemma** *enough-new-all-param*:  
**assumes** ⟨ $\mid \text{UNIV} :: 'fm \text{ set} \mid \leq o \mid \text{UNIV} - \text{params } S$ ⟩ ⟨ $\bigwedge x. \text{is-param } x$ ⟩  
**shows** ⟨*enough-new*  $S$ ⟩  
⟨proof⟩

**end**

**datatype** ( $'x, 'fm$ ) *kind*  
= *Cond* ⟨'fm list ⇒ ('fm set set ⇒ 'fm set ⇒ bool) ⇒ bool⟩ ⟨'fm set ⇒ bool⟩  
| *Wits* ⟨'fm ⇒ 'x ⇒ 'fm list⟩

**inductive** (**in** *Params*) *sat<sub>E</sub>* :: ⟨('x, 'fm) *kind* ⇒ 'fm set set ⇒ bool⟩ **where**  
*sat<sub>E</sub>-Cond* [*intro!*]: ⟨ $(\bigwedge S \text{ ps } Q. S \in C \Longrightarrow \text{set } ps \subseteq S \Longrightarrow P \text{ ps } Q \Longrightarrow Q \text{ } C \text{ } S)$   
⇒ *sat<sub>E</sub>* (*Cond*  $P \text{ } H$ )  $C$ ⟩  
| *sat<sub>E</sub>-Wits* [*intro!*]: ⟨ $(\bigwedge S \text{ p. } S \in C \Longrightarrow p \in S \Longrightarrow (\exists x. \text{is-param } x \wedge \text{set } (W \text{ } p \text{ } x))$ ⟩

$\cup S \in C)) \implies \text{sat}_E (Wits\ W)\ C\rangle$

**inductive-cases** (in *Params*) *sat<sub>E</sub>-CondE*[*elim!*]:  $\langle \text{sat}_E (Cond\ P\ H)\ C \rangle$

**inductive-cases** (in *Params*) *sat<sub>E</sub>-WitsE*[*elim!*]:  $\langle \text{sat}_E (Wits\ W)\ C \rangle$

**inductive** (in *Params*) *sat<sub>A</sub>* ::  $\langle ('x, 'fm)\ kind \Rightarrow 'fm\ set\ set \Rightarrow bool \rangle$  **where**

*sat<sub>A</sub>-Cond* [*intro!*]:  $\langle (\bigwedge S\ ps\ Q. S \in C \implies set\ ps \subseteq S \implies P\ ps\ Q \implies Q\ C\ S) \implies \text{sat}_A (Cond\ P\ H)\ C \rangle$

| *sat<sub>A</sub>-Wits* [*intro!*]:  $\langle (\bigwedge S\ p\ x. S \in C \implies p \in S \implies x \notin params\ S \implies is-param\ x \implies set\ (W\ p\ x) \cup S \in C) \implies \text{sat}_A (Wits\ W)\ C \rangle$

**inductive-cases** (in *Params*) *sat<sub>A</sub>-CondE*[*elim!*]:  $\langle \text{sat}_A (Cond\ P\ H)\ C \rangle$

**inductive-cases** (in *Params*) *sat<sub>A</sub>-WitsE*[*elim!*]:  $\langle \text{sat}_A (Wits\ W)\ C \rangle$

**definition** (in *Params*) *prop<sub>E</sub>* ::  $\langle ('x, 'fm)\ kind\ list \Rightarrow 'fm\ set\ set \Rightarrow bool \rangle$  **where**

$\langle prop_E\ Ks\ C \equiv \forall K \in set\ Ks. \text{sat}_E\ K\ C \rangle$

**definition** (in *Params*) *prop<sub>A</sub>* ::  $\langle ('x, 'fm)\ kind\ list \Rightarrow 'fm\ set\ set \Rightarrow bool \rangle$  **where**

$\langle prop_A\ Ks\ C \equiv \forall K \in set\ Ks. \text{sat}_A\ K\ C \rangle$

**inductive** (in *Params*) *sat<sub>H</sub>* ::  $\langle ('x, 'fm)\ kind \Rightarrow 'fm\ set \Rightarrow bool \rangle$  **where**

*sat<sub>H</sub>-Cond* [*intro!*]:  $\langle H\ S \implies \text{sat}_H (Cond\ P\ H)\ S \rangle$

| *sat<sub>H</sub>-Wits* [*intro!*]:  $\langle (\bigwedge p. p \in S \implies (\exists x. is-param\ x \wedge set\ (W\ p\ x) \subseteq S)) \implies \text{sat}_H (Wits\ W)\ S \rangle$

**inductive-cases** (in *Params*) *sat<sub>H</sub>-CondE*[*elim!*]:  $\langle \text{sat}_H (Cond\ P\ H)\ C \rangle$

**inductive-cases** (in *Params*) *sat<sub>H</sub>-WitsE*[*elim!*]:  $\langle \text{sat}_H (Wits\ W)\ C \rangle$

**definition** (in *Params*) *prop<sub>H</sub>* ::  $\langle ('x, 'fm)\ kind\ list \Rightarrow 'fm\ set \Rightarrow bool \rangle$  **where**

$\langle prop_H\ Ks\ S \equiv \forall K \in set\ Ks. \text{sat}_H\ K\ S \rangle$

**theorem** (in *Params*) *sat<sub>H</sub>-Wits*:  $\langle \text{sat}_E (Wits\ W)\ C \implies S \in C \implies maximal\ C\ S \implies \text{sat}_H (Wits\ W)\ S \rangle$

$\langle proof \rangle$

### 1.3.1 Consistency Kinds

**locale** *Consistency-Kind* = *Params map-fm params-fm is-param*

**for**

*map-fm* ::  $\langle ('x \Rightarrow 'x) \Rightarrow 'fm \Rightarrow 'fm \rangle$  **and**

*params-fm* ::  $\langle 'fm \Rightarrow 'x\ set \rangle$  **and**

*is-param* ::  $\langle 'x \Rightarrow bool \rangle$  +

**fixes** *K* ::  $\langle ('x, 'fm)\ kind \rangle$

**assumes** *respects-close*:  $\langle \bigwedge C. \text{sat}_E\ K\ C \implies \text{sat}_E\ K\ (close\ C) \rangle$

**and** *respects-alt*:  $\langle \bigwedge C. \text{sat}_E\ K\ C \implies subset-closed\ C \implies \text{sat}_A\ K\ (mk-alt-consistency\ C) \rangle$

**and** *respects-fin*:  $\langle \bigwedge C. subset-closed\ C \implies \text{sat}_A\ K\ C \implies \text{sat}_A\ K\ (mk-finite-char\ C) \rangle$

**and** *hintikka*:  $\langle \bigwedge C\ S. \text{sat}_E\ K\ C \implies S \in C \implies maximal\ C\ S \implies \text{sat}_H\ K\ S \rangle$

**locale** *Consistency-Kinds* = *Params* *map-fm* *params-fm* *is-param*  
**for**  
    *map-fm* ::  $\langle ('x \Rightarrow 'x) \Rightarrow 'fm \Rightarrow 'fm \rangle$  **and**  
    *params-fm* ::  $\langle 'fm \Rightarrow 'x \text{ set} \rangle$  **and**  
    *is-param* ::  $\langle 'x \Rightarrow \text{bool} \rangle$  +  
**fixes** *Ks* ::  $\langle ('x, 'fm) \text{ kind list} \rangle$   
**assumes** *all-kinds*:  $\langle \bigwedge K. K \in \text{set } Ks \Longrightarrow \text{Consistency-Kind } \text{map-fm } \text{params-fm } \text{is-param } K \rangle$   
**begin**  
  
**lemma** *sat<sub>E</sub>*:  $\langle K \in \text{set } Ks \Longrightarrow \text{prop}_E Ks C \Longrightarrow \text{sat}_E K C \rangle$   
     $\langle \text{proof} \rangle$   
  
**lemma** *prop<sub>E</sub>-close*:  $\langle \text{prop}_E Ks C \Longrightarrow \text{prop}_E Ks (\text{close } C) \rangle$   
     $\langle \text{proof} \rangle$   
  
**lemma** *prop<sub>E</sub>-alt*:  $\langle \text{prop}_E Ks C \Longrightarrow \text{subset-closed } C \Longrightarrow \text{prop}_A Ks (\text{mk-alt-consistency } C) \rangle$   
     $\langle \text{proof} \rangle$   
  
**lemma** *prop<sub>E</sub>-fin*:  $\langle \text{subset-closed } C \Longrightarrow \text{prop}_A Ks C \Longrightarrow \text{prop}_A Ks (\text{mk-finite-char } C) \rangle$   
     $\langle \text{proof} \rangle$   
  
**definition** *mk-alt-fin* ::  $\langle 'fm \text{ set set} \Rightarrow 'fm \text{ set set} \rangle$  **where**  
     $\langle \text{mk-alt-fin } C \equiv \text{mk-finite-char } (\text{mk-alt-consistency } (\text{close } C)) \rangle$   
  
**lemma** *mk-alt-fin-subset-closed*:  $\langle \text{subset-closed } (\text{mk-alt-fin } C) \rangle$   
     $\langle \text{proof} \rangle$   
  
**lemma** *mk-alt-fin-finite-char*:  $\langle \text{finite-char } (\text{mk-alt-fin } C) \rangle$   
     $\langle \text{proof} \rangle$   
  
**lemma** *mk-alt-fin-in*:  $\langle S \in C \Longrightarrow S \in \text{mk-alt-fin } C \rangle$   
     $\langle \text{proof} \rangle$   
  
**theorem** *prop<sub>E</sub>*:  $\langle \text{prop}_E Ks C \Longrightarrow \text{prop}_A Ks (\text{mk-alt-fin } C) \rangle$   
     $\langle \text{proof} \rangle$   
  
**end**  
  
**fun** (**in** *Params*) *witness-kinds* ::  $\langle ('x, 'fm) \text{ kind list} \Rightarrow 'fm \Rightarrow 'fm \text{ set} \Rightarrow 'fm \text{ set} \rangle$   
**where**  
     $\langle \text{witness-kinds } [] p S = \{ \} \rangle$   
    |  $\langle \text{witness-kinds } (\text{Cond } - - \# Ks) p S = \text{witness-kinds } Ks p S \rangle$   
    |  $\langle \text{witness-kinds } (\text{Wits } W \# Ks) p S =$   
         $(\text{let}$   
             $\text{rest} = \text{witness-kinds } Ks p S;$

$a = \text{SOME } x. x \in \text{Collect is-param - params } (rest \cup \{p\} \cup S)$   
in set  $(W p a) \cup rest$

**lemma** (in Params) *witness-kinds-new*:

**assumes**  $\langle \text{infinite } (UNIV :: 'fm \text{ set}) \rangle$   $\langle \text{infinite } (\text{Collect is-param - params } S) \rangle$   
**shows**  $\langle \text{infinite } (\text{Collect is-param - params } (\text{witness-kinds } Ks p S \cup \{p\} \cup S)) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** (in Params) *witness-kinds*:

**assumes** *inf*:  $\langle \text{infinite } (UNIV :: 'fm \text{ set}) \rangle$  **and**  $\langle \text{infinite } (\text{Collect is-param - params } S) \rangle$   
 $\langle \text{Wits } W \in \text{set } Ks \rangle$   
**shows**  $\langle \exists x. \text{is-param } x \wedge \text{set } (W p x) \subseteq \text{witness-kinds } Ks p S \rangle$   
 $\langle \text{proof} \rangle$

**locale** *Maximal-Consistency* = *wo-rel*  $\langle |UNIV| :: 'fm \text{ rel} \rangle + \text{Consistency-Kinds}$   
*map-fm params-fm is-param Ks*

**for**

*map-fm* ::  $\langle ('x \Rightarrow 'x) \Rightarrow 'fm \Rightarrow 'fm \rangle$  **and**  
*params-fm* ::  $\langle 'fm \Rightarrow 'x \text{ set} \rangle$  **and**  
*is-param* ::  $\langle 'x \Rightarrow \text{bool} \rangle$  **and**  
*Ks* ::  $\langle ('x, 'fm) \text{ kind list} \rangle +$

**assumes** *inf-univ*:  $\langle \text{infinite } (UNIV :: 'fm \text{ set}) \rangle$

**begin**

**lemma** *Cinfinite-r*:  $\langle \text{Cinfinite } |UNIV| :: 'fm \text{ set} \rangle$   
 $\langle \text{proof} \rangle$

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

**lemma** *aboveS-ne*:  $\langle \text{aboveS } a \neq \{\} \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *params-left*:  $\langle \text{enough-new } S \Longrightarrow \text{enough-new } (\text{set } ps \cup S) \rangle$   
 $\langle \text{proof} \rangle$

**definition** *witness* ::  $\langle 'fm \Rightarrow 'fm \text{ set} \Rightarrow 'fm \text{ set} \rangle$  **where**  
 $\langle \text{witness} \equiv \text{witness-kinds } Ks \rangle$

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

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

**definition** *extend* ::  $\langle 'fm \text{ set set} \Rightarrow 'fm \text{ set} \Rightarrow 'fm \Rightarrow 'fm \text{ set} \rangle$  **where**  
 $\langle \text{extend } C S a \equiv \text{worecZSL } S (\text{extendS } C) (\text{extendL } C) a \rangle$

**lemma** *adm-woL-extendL*:  $\langle \text{adm-woL } (\text{extendL } C) \rangle$   
 $\langle \text{proof} \rangle$

**definition** *Extend* ::  $\langle 'fm \text{ set } \text{set} \Rightarrow 'fm \text{ set} \Rightarrow 'fm \text{ set} \rangle$  **where**  
 $\langle \text{Extend } C \ S \equiv \bigcup a. \text{extend } C \ S \ a \rangle$

**lemma** *finite-witness-kinds*:  $\langle \text{finite } (\text{witness-kinds } Qs \ p \ S) \rangle$   
 $\langle \text{proof} \rangle$

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

**lemma** *finite-witness-kinds-params*:  $\langle \text{finite } (\text{params } (\text{witness-kinds } Qs \ p \ S)) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *finite-witness-params*:  $\langle \text{finite } (\text{params } (\text{witness } p \ S)) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *extend-zero* [*simp*]:  $\langle \text{extend } C \ S \ \text{zero} = S \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *extend-succ* [*simp*]:  $\langle \text{extend } C \ S \ (\text{succ } a) =$   
 $(\text{if } \{a\} \cup \text{extend } C \ S \ a \in C \text{ then witness } a \ (\text{extend } C \ S \ a) \cup \{a\} \cup \text{extend } C \ S$   
 $a \text{ else } \text{extend } C \ S \ a) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *extend-isLim* [*simp*]:  
**assumes**  $\langle \text{isLim } a \rangle \langle a \neq \text{zero} \rangle$   
**shows**  $\langle \text{extend } C \ S \ a = (\bigcup b \in \text{underS } a. \text{extend } C \ S \ b) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *extend-subset*:  $\langle S \subseteq \text{extend } C \ S \ a \rangle$   
 $\langle \text{proof} \rangle$

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

**lemma** *extend-underS*:  $\langle b \in \text{underS } a \implies \text{extend } C \ S \ b \subseteq \text{extend } C \ S \ a \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *extend-under*:  $\langle b \in \text{under } a \implies \text{extend } C \ S \ b \subseteq \text{extend } C \ S \ a \rangle$   
 $\langle \text{proof} \rangle$

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

**lemma** *is-chain-extend*:  $\langle \text{is-chain } (\text{extend } C \ S) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *extend-in-C-step*:  
**assumes**  $\langle \text{prop}_A \ Ks \ C \rangle \langle \{a\} \cup \text{extend } C \ S \ a \in C \rangle$   
**and inf**:  $\langle \text{infinite } (\text{Collect is-param} - \text{params } (\{a\} \cup \text{extend } C \ S \ a)) \rangle$   
**shows**  $\langle \text{extend } C \ S \ (\text{succ } a) \in C \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *extend-in-C-stop*:  
**assumes**  $\langle \text{extend } C \ S \ a \in C \rangle$   
**and**  $\langle \{a\} \cup \text{extend } C \ S \ a \notin C \rangle$   
**shows**  $\langle \text{extend } C \ S \ (\text{succ } a) \in C \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *infinite-succ-extend*:  
**assumes**  $\langle S \in C \rangle \langle \text{enough-new } S \rangle \langle \text{isSucc } p \rangle$   
**shows**  $\langle \text{infinite } (\text{Collect is-param} - \text{params } (\text{extend } C \ S \ p)) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *extend-in-C*:  
**assumes**  $\langle \text{prop}_A \ Ks \ C \rangle \langle \text{finite-char } C \rangle \langle S \in C \rangle \langle \text{enough-new } S \rangle$   
**shows**  $\langle \text{extend } C \ S \ a \in C \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *Extend-in-C*:  
**assumes**  $\langle \text{prop}_A \ Ks \ C \rangle \langle \text{finite-char } C \rangle \langle S \in C \rangle \langle \text{enough-new } S \rangle$   
**shows**  $\langle \text{Extend } C \ S \in C \rangle$   
 $\langle \text{proof} \rangle$

**theorem** *Extend-maximal*:  
**assumes**  $\langle \text{subset-closed } C \rangle$   
**shows**  $\langle \text{maximal } C \ (\text{Extend } C \ S) \rangle$   
 $\langle \text{proof} \rangle$

**definition** *witnessed* ::  $\langle 'fm \ set \Rightarrow \text{bool} \rangle$  **where**  
 $\langle \text{witnessed } S \equiv \forall p \in S. \exists S'. \text{infinite } (\text{Collect is-param} - \text{params } S') \wedge \text{witness } p \ S' \subseteq S \rangle$

**theorem** *Extend-witnessed*:  
**assumes**  $\langle \text{prop}_A \ Ks \ C \rangle \langle \text{finite-char } C \rangle \langle S \in C \rangle \langle \text{enough-new } S \rangle$   
**shows**  $\langle \text{witnessed } (\text{Extend } C \ S) \rangle$   
 $\langle \text{proof} \rangle$

**abbreviation** *mk-mcs* ::  $\langle 'fm \ set \Rightarrow 'fm \ set \Rightarrow 'fm \ set \rangle$  **where**  
 $\langle \text{mk-mcs } C \ S \equiv \text{Extend } (\text{mk-alt-fin } C) \ S \rangle$

**theorem** *mk-mcs-rmaximal*:  $\langle \text{maximal } C \ (\text{mk-mcs } C \ S) \rangle$

⟨proof⟩

**theorem** *mk-mcs-witnessed*:

**assumes** ⟨*prop<sub>E</sub>* *Ks C*⟩ ⟨*S* ∈ *C*⟩ ⟨*enough-new S*⟩

**shows** ⟨*witnessed (mk-mcs C S)*⟩

⟨proof⟩

## 1.4 Hintikka Sets

**lemma** *mk-mcs-hintikka*:

**assumes** ⟨*prop<sub>E</sub>* *Ks C*⟩ ⟨*S* ∈ *C*⟩ ⟨*enough-new S*⟩

**shows** ⟨*prop<sub>H</sub>* *Ks (mk-mcs C S)*⟩

⟨proof⟩

**end**

**locale** *Hintikka = Maximal-Consistency map-fm params-fm is-param Ks*

**for**

*map-fm* :: ⟨*'x* ⇒ *'x*⟩ ⇒ *'fm* ⇒ *'fm*⟩ **and**

*params-fm* :: ⟨*'fm* ⇒ *'x set*⟩ **and**

*is-param* :: ⟨*'x* ⇒ *bool*⟩ **and**

*Ks* :: ⟨*'x, 'fm*⟩ *kind list*⟩ +

**fixes** *H* :: ⟨*'fm set*⟩

**assumes** *hintikka*: ⟨*prop<sub>H</sub>* *Ks H*⟩

**begin**

**lemma** *sat<sub>H</sub>*: ⟨*K* ∈ *set Ks* ⇒ *sat<sub>H</sub>* *K H*⟩

⟨proof⟩

**end**

**context** *Maximal-Consistency*

**begin**

**theorem** *mk-mcs-Hintikka*:

**assumes** ⟨*prop<sub>E</sub>* *Ks C*⟩ ⟨*S* ∈ *C*⟩ ⟨*enough-new S*⟩

**shows** ⟨*Hintikka map-fm params-fm is-param Ks (mk-mcs C S)*⟩

⟨proof⟩

**end**

## 1.5 Derivational Consistency

**locale** *Derivational-Kind = Consistency-Kind map-fm params-fm is-param K*

**for**

*map-fm* :: ⟨*'x* ⇒ *'x*⟩ ⇒ *'fm* ⇒ *'fm*⟩ **and**

*params-fm* :: ⟨*'fm* ⇒ *'x set*⟩ **and**

*is-param* :: ⟨*'x* ⇒ *bool*⟩ **and**

$K :: \langle ('x, 'fm) \text{ kind} \rangle +$   
**fixes** *consistent* ::  $\langle 'fm \text{ set} \Rightarrow \text{bool} \rangle (\langle \vdash \rightarrow [51] 50 \rangle)$   
**assumes** *kind*:  $\langle \text{infinite} (UNIV :: 'fm \text{ set}) \Rightarrow \text{sat}_E K \{A. \text{enough-new } A \wedge \vdash A\} \rangle$

**locale** *Derivational-Consistency = Maximal-Consistency map-fm params-fm is-param*  
*Ks*

**for**  
 $\text{map-fm} :: \langle ('x \Rightarrow 'x) \Rightarrow 'fm \Rightarrow 'fm \rangle$  **and**  
 $\text{params-fm} :: \langle 'fm \Rightarrow 'x \text{ set} \rangle$  **and**  
 $\text{is-param} :: \langle 'x \Rightarrow \text{bool} \rangle$  **and**  
 $Ks :: \langle ('x, 'fm) \text{ kind list} \rangle +$   
**fixes** *consistent* ::  $\langle 'fm \text{ set} \Rightarrow \text{bool} \rangle (\langle \vdash \rightarrow [51] 50 \rangle)$   
**assumes** *all-consistent*:  $\langle \text{infinite} (UNIV :: 'fm \text{ set}) \Rightarrow \text{prop}_E Ks \{A. \text{enough-new } A \wedge \vdash A\} \rangle$   
**begin**

**theorem** *Consistency*:  $\langle \text{prop}_E Ks \{A. \text{enough-new } A \wedge \vdash A\} \rangle$   
 $\langle \text{proof} \rangle$

**end**

## 1.6 Weak Derivational Consistency

**locale** *Weak-Derivational-Kind = Consistency-Kind map-fm params-fm is-param*  
*K*

**for**  
 $\text{map-fm} :: \langle ('x \Rightarrow 'x) \Rightarrow 'fm \Rightarrow 'fm \rangle$  **and**  
 $\text{params-fm} :: \langle 'fm \Rightarrow 'x \text{ set} \rangle$  **and**  
 $\text{is-param} :: \langle 'x \Rightarrow \text{bool} \rangle$  **and**  
 $K :: \langle ('x, 'fm) \text{ kind} \rangle +$   
**fixes** *consistent* ::  $\langle 'fm \text{ list} \Rightarrow \text{bool} \rangle (\langle \vdash \rightarrow [51] 50 \rangle)$   
**assumes** *kind*:  $\langle \text{infinite} (\text{Collect } \text{is-param}) \Rightarrow \text{sat}_E K \{S. \exists A. \text{set } A = S \wedge \vdash A\} \rangle$

**locale** *Weak-Derivational-Consistency = Maximal-Consistency map-fm params-fm*  
*is-param Ks*

**for**  
 $\text{map-fm} :: \langle ('x \Rightarrow 'x) \Rightarrow 'fm \Rightarrow 'fm \rangle$  **and**  
 $\text{params-fm} :: \langle 'fm \Rightarrow 'x \text{ set} \rangle$  **and**  
 $\text{is-param} :: \langle 'x \Rightarrow \text{bool} \rangle$  **and**  
 $Ks :: \langle ('x, 'fm) \text{ kind list} \rangle +$   
**fixes** *consistent* ::  $\langle 'fm \text{ list} \Rightarrow \text{bool} \rangle (\langle \vdash \rightarrow [51] 50 \rangle)$   
**assumes** *Consistency*:  $\langle \text{infinite} (UNIV :: 'x \text{ set}) \Rightarrow \text{prop}_E Ks \{S. \exists A. \text{set } A = S \wedge \vdash A\} \rangle$

## 1.7 Conflicts

**locale** *Confl* = *Params map-fm params-fm is-param*  
**for**  
  *map-fm* ::  $\langle 'x \Rightarrow 'x \rangle \Rightarrow 'fm \Rightarrow 'fm \rangle$  **and**  
  *params-fm* ::  $\langle 'fm \Rightarrow 'x \text{ set} \rangle$  **and**  
  *is-param* ::  $\langle 'x \Rightarrow \text{bool} \rangle$  +  
  **fixes** *classify* ::  $\langle 'fm \text{ list} \Rightarrow 'fm \text{ list} \Rightarrow \text{bool} \rangle$  (**infix**  $\langle \rightsquigarrow_{\mathbf{X}} \rangle$  50)  
  **assumes** *confl-map*:  $\langle \bigwedge ps \ qs \ f. ps \rightsquigarrow_{\mathbf{X}} qs \Longrightarrow \text{map} (\text{map-fm } f) ps \rightsquigarrow_{\mathbf{X}} \text{map} (\text{map-fm } f) qs \rangle$   
**begin**

**inductive** *cond* **where**  
  *cond* [*intro!*]:  $\langle ps \rightsquigarrow_{\mathbf{X}} qs \Longrightarrow \text{cond } ps (\lambda S. \text{set } qs \cap S = \{\}) \rangle$

**inductive-cases** *condE*[*elim!*]:  $\langle \text{cond } ps \ Q \rangle$

**inductive** *hint* **where**  
  *hint* [*intro!*]:  $\langle (\bigwedge ps \ qs \ q. ps \rightsquigarrow_{\mathbf{X}} qs \Longrightarrow \text{set } ps \subseteq H \Longrightarrow q \in \text{set } qs \Longrightarrow q \notin H) \Longrightarrow \text{hint } H \rangle$

**declare** *hint.simps*[*simp*]

**abbreviation** *kind* ::  $\langle ('x, 'fm) \text{ kind} \rangle$  **where**  
   $\langle \text{kind} \equiv \text{Cond } \text{cond } \text{hint} \rangle$

**end**

**sublocale** *Confl*  $\subseteq$  *Consistency-Kind map-fm params-fm is-param kind*  
*<proof>*

**locale** *Derivational-Confl* = *Confl map-fm params-fm is-param classify*  
**for**  
  *map-fm* ::  $\langle ('x \Rightarrow 'x) \Rightarrow 'fm \Rightarrow 'fm \rangle$  **and**  
  *params-fm* ::  $\langle 'fm \Rightarrow 'x \text{ set} \rangle$  **and**  
  *is-param* ::  $\langle 'x \Rightarrow \text{bool} \rangle$  **and**  
  *classify* ::  $\langle 'fm \text{ list} \Rightarrow 'fm \text{ list} \Rightarrow \text{bool} \rangle$  (**infix**  $\langle \rightsquigarrow_{\mathbf{X}} \rangle$  50) +  
  **fixes** *consistent* ::  $\langle 'fm \text{ set} \Rightarrow \text{bool} \rangle$  ( $\langle \vdash \rightarrow [51] 50 \rangle$ )  
  **assumes** *consistent*:  $\langle \bigwedge S \ ps \ qs \ x. \text{set } ps \subseteq S \Longrightarrow ps \rightsquigarrow_{\mathbf{X}} qs \Longrightarrow x \in \text{set } qs \Longrightarrow x \in S \Longrightarrow \neg \vdash S \rangle$

**sublocale** *Derivational-Confl*  $\subseteq$  *Derivational-Kind map-fm params-fm is-param kind consistent*  
*<proof>*

**locale** *Weak-Derivational-Confl* = *Confl map-fm params-fm is-param classify*  
**for**  
  *map-fm* ::  $\langle ('x \Rightarrow 'x) \Rightarrow 'fm \Rightarrow 'fm \rangle$  **and**

**params-fm** ::  $\langle 'fm \Rightarrow 'x \text{ set} \rangle$  **and**  
**is-param** ::  $\langle 'x \Rightarrow \text{bool} \rangle$  **and**  
**classify** ::  $\langle 'fm \text{ list} \Rightarrow 'fm \text{ list} \Rightarrow \text{bool} \rangle$  (**infix**  $\langle \rightsquigarrow_{\mathbf{x}} \rangle$  50) +  
**fixes consistent** ::  $\langle 'fm \text{ list} \Rightarrow \text{bool} \rangle$  ( $\langle \vdash - \rangle$  [51] 50)  
**assumes consistent**:  $\langle \bigwedge A \text{ ps qs } x. \text{ set ps} \subseteq \text{ set } A \implies \text{ ps} \rightsquigarrow_{\mathbf{x}} \text{ qs} \implies x \in \text{ set qs} \implies x \in \text{ set } A \implies \neg \vdash A \rangle$

**sublocale** *Weak-Derivational-Conf*  $\subseteq$  *Weak-Derivational-Kind map-fm params-fm is-param kind consistent*  
 $\langle \text{proof} \rangle$

## 1.8 Alpha

**locale** *Alpha* = *Params map-fm params-fm is-param*

**for**

**map-fm** ::  $\langle ('x \Rightarrow 'x) \Rightarrow 'fm \Rightarrow 'fm \rangle$  **and**

**params-fm** ::  $\langle 'fm \Rightarrow 'x \text{ set} \rangle$  **and**

**is-param** ::  $\langle 'x \Rightarrow \text{bool} \rangle$  +

**fixes classify** ::  $\langle 'fm \text{ list} \Rightarrow 'fm \text{ list} \Rightarrow \text{bool} \rangle$  (**infix**  $\langle \rightsquigarrow_{\alpha} \rangle$  50)

**assumes alpha-map**:  $\langle \bigwedge \text{ps qs } f. \text{ ps} \rightsquigarrow_{\alpha} \text{ qs} \longrightarrow \text{ map } (\text{map-fm } f) \text{ ps} \rightsquigarrow_{\alpha} \text{ map } (\text{map-fm } f) \text{ qs} \rangle$

**begin**

**inductive cond where**

**cond** [intro!]:  $\langle \text{ps} \rightsquigarrow_{\alpha} \text{ qs} \implies \text{ cond ps } (\lambda C \text{ S}. \text{ set qs} \cup \text{ S} \in C) \rangle$

**inductive-cases condE**[elim!]:  $\langle \text{cond ps } Q \rangle$

**inductive hint where**

**hint** [intro!]:  $\langle (\bigwedge \text{ps qs } q. \text{ ps} \rightsquigarrow_{\alpha} \text{ qs} \implies \text{ set ps} \subseteq H \implies q \in \text{ set qs} \implies q \in H) \implies \text{ hint } H \rangle$

**declare** *hint.simps*[simp]

**abbreviation kind** ::  $\langle ('x, 'fm) \text{ kind} \rangle$  **where**

$\langle \text{kind} \equiv \text{Cond cond hint} \rangle$

**end**

**sublocale** *Alpha*  $\subseteq$  *Consistency-Kind map-fm params-fm is-param kind*  
 $\langle \text{proof} \rangle$

**locale** *Derivational-Alpha* = *Alpha map-fm params-fm is-param classify*  
**for**

**map-fm** ::  $\langle ('x \Rightarrow 'x) \Rightarrow 'fm \Rightarrow 'fm \rangle$  **and**

**params-fm** ::  $\langle 'fm \Rightarrow 'x \text{ set} \rangle$  **and**

**is-param** ::  $\langle 'x \Rightarrow \text{bool} \rangle$  **and**

**classify** ::  $\langle 'fm \text{ list} \Rightarrow 'fm \text{ list} \Rightarrow \text{bool} \rangle$  (**infix**  $\langle \rightsquigarrow_{\alpha} \rangle$  50) +

**fixes consistent** ::  $\langle 'fm \text{ set} \Rightarrow \text{bool} \rangle$  ( $\langle \vdash - \rangle$  [51] 50)

**assumes consistent:**  $\langle \bigwedge S \text{ ps qs. set ps} \subseteq S \implies \text{ps} \rightsquigarrow_\alpha \text{qs} \implies \vdash S \implies \vdash \text{set qs} \cup S \rangle$

**sublocale** *Derivational-Alpha*  $\subseteq$  *Derivational-Kind map-fm params-fm is-param kind consistent*  
 $\langle \text{proof} \rangle$

**locale** *Weak-Derivational-Alpha* = *Alpha map-fm params-fm is-param classify*  
**for**

*map-fm* ::  $\langle 'x \Rightarrow 'x \rangle \Rightarrow 'fm \Rightarrow 'fm \rangle$  **and**

*params-fm* ::  $\langle 'fm \Rightarrow 'x \text{ set} \rangle$  **and**

*is-param* ::  $\langle 'x \Rightarrow \text{bool} \rangle$  **and**

*classify* ::  $\langle 'fm \text{ list} \Rightarrow 'fm \text{ list} \Rightarrow \text{bool} \rangle$  (**infix**  $\langle \rightsquigarrow_\alpha \rangle$  50) +

**fixes consistent** ::  $\langle 'fm \text{ list} \Rightarrow \text{bool} \rangle$  ( $\langle \vdash - \rangle$  [51] 50)

**assumes consistent:**  $\langle \bigwedge A \text{ ps qs. set ps} \subseteq \text{set } A \implies \text{ps} \rightsquigarrow_\alpha \text{qs} \implies \vdash A \implies \vdash \text{qs} \text{ @ } A \rangle$

**sublocale** *Weak-Derivational-Alpha*  $\subseteq$  *Weak-Derivational-Kind map-fm params-fm is-param kind consistent*  
 $\langle \text{proof} \rangle$

## 1.9 Beta

**locale** *Beta* = *Params map-fm params-fm is-param*

**for**

*map-fm* ::  $\langle 'x \Rightarrow 'x \rangle \Rightarrow 'fm \Rightarrow 'fm \rangle$  **and**

*params-fm* ::  $\langle 'fm \Rightarrow 'x \text{ set} \rangle$  **and**

*is-param* ::  $\langle 'x \Rightarrow \text{bool} \rangle$  +

**fixes classify** ::  $\langle 'fm \text{ list} \Rightarrow 'fm \text{ list} \Rightarrow \text{bool} \rangle$  (**infix**  $\langle \rightsquigarrow_\beta \rangle$  50)

**assumes beta-map:**  $\langle \bigwedge \text{ps qs } f. \text{ps} \rightsquigarrow_\beta \text{qs} \longrightarrow \text{map} (\text{map-fm } f) \text{ps} \rightsquigarrow_\beta \text{map} (\text{map-fm } f) \text{qs} \rangle$

**begin**

**inductive cond where**

*cond* [intro!]:  $\langle \text{ps} \rightsquigarrow_\beta \text{qs} \implies \text{cond ps } (\lambda C \text{ S. } \exists q \in \text{set qs. } \{q\} \cup S \in C) \rangle$

**inductive-cases condE[elim!]:**  $\langle \text{cond ps } Q \rangle$

**inductive hint where**

*hint* [intro!]:  $\langle (\bigwedge \text{ps qs. ps} \rightsquigarrow_\beta \text{qs} \implies \text{set ps} \subseteq H \implies \exists q \in \text{set qs. } q \in H) \implies \text{hint } H \rangle$

**declare** *hint.simps[simp]*

**abbreviation kind** ::  $\langle ('x, 'fm) \text{ kind} \rangle$  **where**

$\langle \text{kind} \equiv \text{Cond cond hint} \rangle$

**end**

**sublocale**  $Beta \subseteq Consistency\text{-}Kind\ map\text{-}fm\ params\text{-}fm\ is\text{-}param\ kind$   
 $\langle proof \rangle$

**locale**  $Derivational\text{-}Beta = Beta\ map\text{-}fm\ params\text{-}fm\ is\text{-}param\ classify$   
**for**

$map\text{-}fm :: \langle 'x \Rightarrow 'x \rangle \Rightarrow 'fm \Rightarrow 'fm \rangle$  **and**  
 $params\text{-}fm :: \langle 'fm \Rightarrow 'x\ set \rangle$  **and**  
 $is\text{-}param :: \langle 'x \Rightarrow bool \rangle$  **and**  
 $classify :: \langle 'fm\ list \Rightarrow 'fm\ list \Rightarrow bool \rangle$  (**infix**  $\langle \rightsquigarrow_\beta \rangle$  50) +  
**fixes**  $consistent :: \langle 'fm\ set \Rightarrow bool \rangle$  ( $\langle \vdash \rightarrow [51] 50 \rangle$ )  
**assumes**  $consistent: \langle \bigwedge S\ ps\ qs.\ set\ ps \subseteq S \Longrightarrow ps \rightsquigarrow_\beta qs \Longrightarrow \vdash S \Longrightarrow \exists q \in set\ qs.\ \vdash \{q\} \cup S \rangle$

**sublocale**  $Derivational\text{-}Beta \subseteq Derivational\text{-}Kind\ map\text{-}fm\ params\text{-}fm\ is\text{-}param\ kind$   
 $consistent$   
 $\langle proof \rangle$

**locale**  $Weak\text{-}Derivational\text{-}Beta = Beta\ map\text{-}fm\ params\text{-}fm\ is\text{-}param\ classify$   
**for**

$map\text{-}fm :: \langle 'x \Rightarrow 'x \rangle \Rightarrow 'fm \Rightarrow 'fm \rangle$  **and**  
 $params\text{-}fm :: \langle 'fm \Rightarrow 'x\ set \rangle$  **and**  
 $is\text{-}param :: \langle 'x \Rightarrow bool \rangle$  **and**  
 $classify :: \langle 'fm\ list \Rightarrow 'fm\ list \Rightarrow bool \rangle$  (**infix**  $\langle \rightsquigarrow_\beta \rangle$  50) +  
**fixes**  $consistent :: \langle 'fm\ list \Rightarrow bool \rangle$  ( $\langle \vdash \rightarrow [51] 50 \rangle$ )  
**assumes**  $consistent: \langle \bigwedge A\ ps\ qs.\ set\ ps \subseteq set\ A \Longrightarrow ps \rightsquigarrow_\beta qs \Longrightarrow \vdash A \Longrightarrow \exists q \in set\ qs.\ \vdash q \# A \rangle$

**sublocale**  $Weak\text{-}Derivational\text{-}Beta \subseteq Weak\text{-}Derivational\text{-}Kind\ map\text{-}fm\ params\text{-}fm\ is\text{-}param\ kind\ consistent$   
 $\langle proof \rangle$

## 1.10 Gamma

**locale**  $Gamma = Params\ map\text{-}fm\ params\text{-}fm\ is\text{-}param$   
**for**

$map\text{-}tm :: \langle 'x \Rightarrow 'x \rangle \Rightarrow 'tm \Rightarrow 'tm \rangle$  **and**  
 $map\text{-}fm :: \langle 'x \Rightarrow 'x \rangle \Rightarrow 'fm \Rightarrow 'fm \rangle$  **and**  
 $params\text{-}fm :: \langle 'fm \Rightarrow 'x\ set \rangle$  **and**  
 $is\text{-}param :: \langle 'x \Rightarrow bool \rangle$  +  
**fixes**  $classify :: \langle 'fm\ list \Rightarrow ('fm\ set \Rightarrow 'tm\ set) \times ('tm \Rightarrow 'fm\ list) \Rightarrow bool \rangle$  (**infix**  $\langle \rightsquigarrow_\gamma \rangle$  50)  
**assumes**  $gamma\text{-}map: \langle \bigwedge ps\ F\ qs\ f.\ ps \rightsquigarrow_\gamma (F, qs) \Longrightarrow (\exists G\ rs.\ map\ (map\text{-}fm\ f)\ ps \rightsquigarrow_\gamma (G, rs) \wedge$   
 $(\forall S.\ map\text{-}tm\ f\ 'F\ S \subseteq G\ (map\text{-}fm\ f\ 'S)) \wedge$   
 $(\forall t.\ map\ (map\text{-}fm\ f)\ (qs\ t) = rs\ (map\text{-}tm\ f\ t))) \rangle$   
**and**  $gamma\text{-}mono: \langle \bigwedge ps\ F\ qs\ S\ S'.\ ps \rightsquigarrow_\gamma (F, qs) \Longrightarrow S \subseteq S' \Longrightarrow F\ S \subseteq F\ S' \rangle$   
**and**  $gamma\text{-}fin: \langle \bigwedge ps\ F\ qs\ t\ A.\ ps \rightsquigarrow_\gamma (F, qs) \Longrightarrow t \in F\ A \Longrightarrow \exists B \subseteq A.\ finite$

```

B ∧ t ∈ F B⟩
begin

inductive cond where
  cond [intro!]: ⟨ps ∼γ (F, qs) ⟹ cond ps (λC S. ∀ t ∈ F S. set (qs t) ∪ S ∈ C)⟩

inductive-cases condE[elim!]: ⟨cond ps Q⟩

inductive hint where
  hint [intro!]: ⟨(∧ ps F qs. ps ∼γ (F, qs) ⟹ set ps ⊆ H ⟹ (∀ t ∈ F H. set (qs t) ⊆ H)) ⟹ hint H⟩

declare hint.simps[simp]

abbreviation kind :: ⟨('x, 'fm) kind⟩ where
  ⟨kind ≡ Cond cond hint⟩

end

sublocale Gamma ⊆ Consistency-Kind map-fm params-fm is-param kind
  ⟨proof⟩

locale Derivational-Gamma = Gamma map-tm map-fm params-fm is-param clas-
sify
  for
    map-tm :: ⟨('x ⇒ 'x) ⇒ 'tm ⇒ 'tm⟩ and
    map-fm :: ⟨('x ⇒ 'x) ⇒ 'fm ⇒ 'fm⟩ and
    params-fm :: ⟨'fm ⇒ 'x set⟩ and
    is-param :: ⟨'x ⇒ bool⟩ and
    classify :: ⟨'fm list ⇒ ('fm set ⇒ 'tm set) × ('tm ⇒ 'fm list) ⇒ bool⟩ (infix
  ⟨∼γ⟩ 50) +
    fixes consistent :: ⟨'fm set ⇒ bool⟩ (⟨| -> [51] 50)
    assumes consistent: ⟨∧ S ps F qs t. set ps ⊆ S ⟹ ps ∼γ (F, qs) ⟹ t ∈ F S
  ⟹ ⊢ S ⟹ ⊢ set (qs t) ∪ S⟩

sublocale Derivational-Gamma ⊆ Derivational-Kind map-fm params-fm is-param
kind consistent
  ⟨proof⟩

locale Weak-Derivational-Gamma = Gamma map-tm map-fm params-fm is-param
classify
  for
    map-tm :: ⟨('x ⇒ 'x) ⇒ 'tm ⇒ 'tm⟩ and
    map-fm :: ⟨('x ⇒ 'x) ⇒ 'fm ⇒ 'fm⟩ and
    params-fm :: ⟨'fm ⇒ 'x set⟩ and
    is-param :: ⟨'x ⇒ bool⟩ and
    classify :: ⟨'fm list ⇒ ('fm set ⇒ 'tm set) × ('tm ⇒ 'fm list) ⇒ bool⟩ (infix
  ⟨∼γ⟩ 50) +
    fixes consistent :: ⟨'fm list ⇒ bool⟩ (⟨| -> [51] 50)

```

**assumes** *consistent*:  $\langle \bigwedge A \ ps \ F \ qs \ t. \ set \ ps \subseteq \ set \ A \implies ps \rightsquigarrow_\gamma (F, qs) \implies t \in F \ (set \ A) \implies \vdash A \implies \vdash qs \ t \ @ \ A \rangle$

**sublocale** *Weak-Derivational-Gamma*  $\subseteq$  *Weak-Derivational-Kind map-fm params-fm is-param kind consistent*  
 $\langle proof \rangle$

**locale** *Gamma-UNIV* = *Params map-fm params-fm is-param*  
**for**

*map-tm* ::  $\langle 'x \Rightarrow 'x \rangle \Rightarrow 'tm \Rightarrow 'tm$  **and**

*map-fm* ::  $\langle 'x \Rightarrow 'x \rangle \Rightarrow 'fm \Rightarrow 'fm$  **and**

*params-fm* ::  $\langle 'fm \Rightarrow 'x \ set \rangle$  **and**

*is-param* ::  $\langle 'x \Rightarrow bool \rangle$  +

**fixes** *classify* ::  $\langle 'fm \ list \Rightarrow ('tm \Rightarrow 'fm \ list) \Rightarrow bool \rangle$  (**infix**  $\langle \rightsquigarrow_\gamma'' \rangle$  50)

**assumes** *gamma-map-UNIV*:  $\langle \bigwedge ps \ qs \ f. \ ps \rightsquigarrow_\gamma' qs \implies \exists rs. \ map \ (map-fm \ f) \ ps \rightsquigarrow_\gamma' rs \wedge$

$(\forall t. \ map \ (map-fm \ f) \ (qs \ t) = rs \ (map-tm \ f \ t)) \rangle$

**begin**

**abbreviation** (*input*) *classify-UNIV* **where**

$\langle classify-UNIV \equiv \lambda ps \ (F, qs). \ (F = (\lambda-. \ UNIV)) \wedge ps \rightsquigarrow_\gamma' qs \rangle$

**end**

**sublocale** *Gamma-UNIV*  $\subseteq$  *Gamma map-tm map-fm params-fm is-param classify-UNIV*  
 $\langle proof \rangle$

## 1.11 Delta

**locale** *Delta* = *Params map-fm params-fm is-param*  
**for**

*map-fm* ::  $\langle 'x \Rightarrow 'x \rangle \Rightarrow 'fm \Rightarrow 'fm$  **and**

*params-fm* ::  $\langle 'fm \Rightarrow 'x \ set \rangle$  **and**

*is-param* ::  $\langle 'x \Rightarrow bool \rangle$  +

**fixes**  $\delta$  ::  $\langle 'fm \Rightarrow 'x \Rightarrow 'fm \ list \rangle$

**assumes** *delta-map*:  $\langle \bigwedge p \ f \ x. \ is-param \ x \implies is-subst \ f \implies \delta \ (map-fm \ f \ p) \ (f \ x) = map \ (map-fm \ f) \ (\delta \ p \ x) \rangle$

**begin**

**abbreviation**  $\langle kind \equiv Wits \ \delta \rangle$

**end**

**sublocale** *Delta*  $\subseteq$  *Consistency-Kind map-fm params-fm is-param kind*  
 $\langle proof \rangle$

**locale** *Derivational-Delta* = *Delta map-fm params-fm is-param  $\delta$*   
**for**

*map-fm* ::  $\langle 'x \Rightarrow 'x \rangle \Rightarrow 'fm \Rightarrow 'fm$  **and**

**params-fm** ::  $\langle 'fm \Rightarrow 'x \text{ set} \rangle$  **and**  
**is-param** ::  $\langle 'x \Rightarrow \text{bool} \rangle$  **and**  
 $\delta$  ::  $\langle 'fm \Rightarrow 'x \Rightarrow 'fm \text{ list} \rangle +$   
**fixes** *consistent* ::  $\langle 'fm \text{ set} \Rightarrow \text{bool} \rangle$  ( $\langle \vdash - \rangle$  [51] 50)  
**assumes** *consistent*:  $\langle \bigwedge S p x. p \in S \Rightarrow \text{is-param } x \Rightarrow x \notin \text{params } S \Rightarrow \vdash S \Rightarrow \vdash \text{set } (\delta p x) \cup S \rangle$

**sublocale** *Derivational-Delta*  $\subseteq$  *Derivational-Kind map-fm params-fm is-param kind consistent*  
 $\langle \text{proof} \rangle$

**locale** *Weak-Derivational-Delta* = *Delta map-fm params-fm is-param*  $\delta$   
**for**  
**map-fm** ::  $\langle ('x \Rightarrow 'x) \Rightarrow 'fm \Rightarrow 'fm \rangle$  **and**  
**params-fm** ::  $\langle 'fm \Rightarrow 'x \text{ set} \rangle$  **and**  
**is-param** ::  $\langle 'x \Rightarrow \text{bool} \rangle$  **and**  
 $\delta$  ::  $\langle 'fm \Rightarrow 'x \Rightarrow 'fm \text{ list} \rangle +$   
**fixes** *consistent* ::  $\langle 'fm \text{ list} \Rightarrow \text{bool} \rangle$  ( $\langle \vdash - \rangle$  [51] 50)  
**assumes** *consistent*:  $\langle \bigwedge A p x. p \in \text{set } A \Rightarrow \text{is-param } x \Rightarrow x \notin \text{params } (\text{set } A) \Rightarrow \vdash A \Rightarrow \vdash \delta p x @ A \rangle$

**sublocale** *Weak-Derivational-Delta*  $\subseteq$  *Weak-Derivational-Kind map-fm params-fm is-param kind consistent*  
 $\langle \text{proof} \rangle$

## 1.12 Modal

The Hintikka property you want depends on the concrete logic. See Term-Modal Logics by Fitting, Thalmann and Voronkov, p. 156 bottom.

**locale** *Modal* = *Params map-fm params-fm is-param*  
**for**  
**map-fm** ::  $\langle ('x \Rightarrow 'x) \Rightarrow 'fm \Rightarrow 'fm \rangle$  **and**  
**params-fm** ::  $\langle 'fm \Rightarrow 'x \text{ set} \rangle$  **and**  
**is-param** ::  $\langle 'x \Rightarrow \text{bool} \rangle +$   
**fixes** *classify* ::  $\langle 'fm \text{ list} \Rightarrow ('fm \text{ set} \Rightarrow 'fm \text{ set}) \times 'fm \text{ list} \Rightarrow \text{bool} \rangle$  (**infix**  $\langle \rightsquigarrow_{\square} \rangle$  50)  
**and** *hint* ::  $\langle 'fm \text{ set} \Rightarrow \text{bool} \rangle$   
**assumes** *modal-map*:  $\langle \bigwedge ps F qs f. ps \rightsquigarrow_{\square} (F, qs) \Rightarrow \exists G. \text{map } (\text{map-fm } f) ps \rightsquigarrow_{\square} (G, \text{map } (\text{map-fm } f) qs) \wedge (\forall S. \text{map-fm } f ' F S \subseteq G (\text{map-fm } f ' S)) \rangle$   
**and** *modal-mono*:  $\langle \bigwedge ps F qs S S'. ps \rightsquigarrow_{\square} (F, qs) \Rightarrow S \subseteq S' \Rightarrow F S \subseteq F S' \rangle$   
**and** *modal-fin*:  $\langle \bigwedge ps F qs S A. ps \rightsquigarrow_{\square} (F, qs) \Rightarrow \text{finite } A \Rightarrow A \subseteq F S \Rightarrow \exists S' \subseteq S. \text{finite } S' \wedge A \subseteq F S' \rangle$   
**begin**

**inductive** *cond* **where**  
*cond* [*intro!*]:  $\langle ps \rightsquigarrow_{\square} (F, qs) \Rightarrow \text{cond } ps (\lambda C S. \text{set } qs \cup F S \in C) \rangle$

**inductive-cases**  $\text{condE}[\text{elim!}]$ :  $\langle \text{cond } ps \ Q \rangle$

**abbreviation**  $\text{kind} :: \langle ('x, 'fm) \ \text{kind} \rangle$  **where**  
 $\langle \text{kind} \equiv \text{Cond } \text{cond } \text{hint} \rangle$

**end**

**locale**  $\text{ModalH} = \text{Modal } \text{map-fm } \text{params-fm } \text{is-param } \text{classify } \text{hint}$   
**for**

$\text{map-fm} :: \langle ('x \Rightarrow 'x) \Rightarrow 'fm \Rightarrow 'fm \rangle$  **and**

$\text{params-fm} :: \langle 'fm \Rightarrow 'x \ \text{set} \rangle$  **and**

$\text{is-param} :: \langle 'x \Rightarrow \text{bool} \rangle$  **and**

$\text{classify} :: \langle 'fm \ \text{list} \Rightarrow ('fm \ \text{set} \Rightarrow 'fm \ \text{set}) \times 'fm \ \text{list} \Rightarrow \text{bool} \rangle$  (**infix**  $\langle \rightsquigarrow_{\square} \rangle$  50)

**and**

$\text{hint} :: \langle 'fm \ \text{set} \Rightarrow \text{bool} \rangle$  +

**assumes**  $\text{modal-hintikka}$ :  $\langle \bigwedge C \ S. \ \text{sat}_E \ \text{kind } C \Longrightarrow S \in C \Longrightarrow \text{maximal } C \ S \Longrightarrow \text{sat}_H \ \text{kind } S \rangle$

**sublocale**  $\text{ModalH} \subseteq \text{Consistency-Kind } \text{map-fm } \text{params-fm } \text{is-param } \text{kind}$   
 $\langle \text{proof} \rangle$

**locale**  $\text{Derivational-Modal} = \text{ModalH } \text{map-fm } \text{params-fm } \text{is-param } \text{classify}$   
**for**

$\text{map-fm} :: \langle ('x \Rightarrow 'x) \Rightarrow 'fm \Rightarrow 'fm \rangle$  **and**

$\text{params-fm} :: \langle 'fm \Rightarrow 'x \ \text{set} \rangle$  **and**

$\text{is-param} :: \langle 'x \Rightarrow \text{bool} \rangle$  **and**

$\text{classify} :: \langle 'fm \ \text{list} \Rightarrow ('fm \ \text{set} \Rightarrow 'fm \ \text{set}) \times 'fm \ \text{list} \Rightarrow \text{bool} \rangle$  (**infix**  $\langle \rightsquigarrow_{\square} \rangle$  50)

+

**fixes**  $\text{consistent} :: \langle 'fm \ \text{set} \Rightarrow \text{bool} \rangle$  ( $\langle \vdash \ - \rangle$  [51] 50)

**assumes**  $\text{consistent}$ :  $\langle \bigwedge S \ ps \ F \ qs. \ \text{set } ps \subseteq S \Longrightarrow ps \rightsquigarrow_{\square} (F, qs) \Longrightarrow \vdash S \Longrightarrow \vdash \text{set } qs \cup F \ S \rangle$

**and**  $\text{params-subset}$ :  $\langle \bigwedge ps \ F \ qs \ S. \ ps \rightsquigarrow_{\square} (F, qs) \Longrightarrow \text{params } (F \ S) \subseteq \text{params } S \rangle$

**sublocale**  $\text{Derivational-Modal} \subseteq \text{Derivational-Kind } \text{map-fm } \text{params-fm } \text{is-param}$   
 $\text{kind } \text{consistent}$   
 $\langle \text{proof} \rangle$

**locale**  $\text{Weak-Derivational-Modal} = \text{ModalH } \text{map-fm } \text{params-fm } \text{is-param } \text{classify}$   
**for**

$\text{map-fm} :: \langle ('x \Rightarrow 'x) \Rightarrow 'fm \Rightarrow 'fm \rangle$  **and**

$\text{params-fm} :: \langle 'fm \Rightarrow 'x \ \text{set} \rangle$  **and**

$\text{is-param} :: \langle 'x \Rightarrow \text{bool} \rangle$  **and**

$\text{classify} :: \langle 'fm \ \text{list} \Rightarrow ('fm \ \text{set} \Rightarrow 'fm \ \text{set}) \times 'fm \ \text{list} \Rightarrow \text{bool} \rangle$  (**infix**  $\langle \rightsquigarrow_{\square} \rangle$  50)

+

**fixes**  $\text{consistent} :: \langle 'fm \ \text{list} \Rightarrow \text{bool} \rangle$  ( $\langle \vdash \ - \rangle$  [51] 50)

**assumes**  $\text{consistent}$ :  $\langle \bigwedge S \ ps \ F \ qs \ S'. \ \text{set } ps \subseteq \text{set } S \Longrightarrow ps \rightsquigarrow_{\square} (F, qs) \Longrightarrow \vdash S \Longrightarrow \text{set } S' = F \ (\text{set } S) \Longrightarrow \vdash qs \ @ \ S' \rangle$

**and**  $\text{params-subset}$ :  $\langle \bigwedge ps \ F \ qs \ S. \ ps \rightsquigarrow_{\square} (F, qs) \Longrightarrow \text{params } (F \ S) \subseteq \text{params } S \rangle$

$S$

**and**  $F$ -size:  $\langle \bigwedge ps F qs S. ps \sim_{\square} (F, qs) \implies |F S| \leq o |S| \rangle$

**sublocale** *Weak-Derivational-Modal*  $\subseteq$  *Weak-Derivational-Kind map-fm params-fm*  
*is-param kind consistent*  
*\langle proof \rangle*

**end**

## Chapter 2

# Example: First-Order Logic with Restricted Instantiation

```
theory Example-Bounded-FOL imports  
  Abstract-Consistency-Property  
begin
```

### 2.1 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 :: <'f, 'p> fm ⇒ ('f, 'p) fm> (<¬ -> [70] 70) where  
  <¬ p ≡ p ⟶ ⊥>
```

```
abbreviation has-subterm :: <'f, 'p> fm> where  
  <has-subterm ≡ ·undefined [#0] ⟶ ·undefined [#0]>
```

```
abbreviation with-subterm :: <'f, 'p> fm ⇒ ('f, 'p) fm> where  
  <with-subterm p ≡ has-subterm ⟶ p>
```

### 2.2 Semantics

```
datatype ('a, 'f, 'p) model = Model <'a set> <nat ⇒ 'a> <'f ⇒ 'a list ⇒ 'a> <'p  
  ⇒ 'a list ⇒ bool>
```

**primrec** *wf-model* ::  $\langle 'a, 'f, 'p \rangle \text{ model} \Rightarrow \text{bool} \rangle$  **where**  
 $\langle \text{wf-model } (\text{Model } U \ E \ F \ G) \longleftrightarrow (\forall n. E \ n \in U) \wedge (\forall f \ ts. F \ f \ ts \in U) \rangle$

**fun** *semantics-tm* ::  $\langle (\text{nat} \Rightarrow 'a) \times ('f \Rightarrow 'a \ \text{list} \Rightarrow 'a) \Rightarrow 'f \ \text{tm} \Rightarrow 'a \rangle$  ( $\langle \langle - \rangle \rangle$ )  
**where**  
 $\langle \langle (E, -) \rangle (\#n) = E \ n \rangle$   
 $\langle \langle (E, F) \rangle (\text{Of } ts) = F \ f \ (\text{map } \langle (E, F) \rangle \ ts) \rangle$

**primrec** *add-env* ::  $\langle 'a \Rightarrow (\text{nat} \Rightarrow 'a) \Rightarrow \text{nat} \Rightarrow 'a \rangle$  (**infix**  $\langle \gg \rangle$  0) **where**  
 $\langle (t \gg s) \ 0 = t \rangle$   
 $\langle (t \gg s) \ (\text{Suc } n) = s \ n \rangle$

**fun** *semantics-fm* ::  $\langle 'a, 'f, 'p \rangle \text{ model} \Rightarrow ('f, 'p) \ \text{fm} \Rightarrow \text{bool} \rangle$  (**infix**  $\langle \models \rangle$  50) **where**  
 $\langle (- \models \perp) = \text{False} \rangle$   
 $\langle \langle (\text{Model } - \ E \ F \ G \models \cdot P \ ts) = G \ P \ (\text{map } \langle (E, F) \rangle \ ts) \rangle$   
 $\langle \langle (\text{Model } U \ E \ F \ G \models p \longrightarrow q) = (\text{Model } U \ E \ F \ G \models p \longrightarrow \text{Model } U \ E \ F \ G \models q) \rangle$   
 $\langle \langle (\text{Model } U \ E \ F \ G \models \forall p) = (\forall x \in U. \text{Model } U \ (x \gg E) \ F \ G \models p) \rangle$

## 2.3 Operations

**primrec** *lift-tm* ::  $\langle 'f \ \text{tm} \Rightarrow 'f \ \text{tm} \rangle$  **where**  
 $\langle \text{lift-tm } (\#n) = \#(n+1) \rangle$   
 $\langle \text{lift-tm } (\text{Of } ts) = \text{Of } (\text{map } \text{lift-tm } \ ts) \rangle$

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

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

**abbreviation** *inst-single* ::  $\langle 'f \ \text{tm} \Rightarrow ('f, 'p) \ \text{fm} \Rightarrow ('f, 'p) \ \text{fm} \rangle$  ( $\langle \langle - \rangle \rangle$ ) **where**  
 $\langle t \equiv \text{sub-fm } (t \gg \#) \rangle$

**abbreviation** *psub* ::  $\langle ('f \Rightarrow 'g) \Rightarrow ('f, 'p) \ \text{fm} \Rightarrow ('g, 'p) \ \text{fm} \rangle$  **where**  
 $\langle \text{psub } f \equiv \text{map-fm } f \ \text{id} \rangle$

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

### 2.3.1 Lemmas

**lemma** *wf-model-tm* [*simp*]:  $\langle \text{wf-model } (Model\ U\ E\ F\ G) \implies \llbracket (E, F) \rrbracket t \in U \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *size-sub-fm* [*simp*]:  $\langle \text{size-fm } (sub-fm\ s\ p) = \text{size-fm } p \rangle$   
 $\langle \text{proof} \rangle$

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

**lemma** *upd-params-fm* [*simp*]:  
**assumes**  $\langle f \notin \text{params-fm } p \rangle$   
**shows**  $\langle Model\ U\ E\ (F(f := x))\ G \models p \iff Model\ U\ E\ F\ G \models p \rangle$   
 $\langle \text{proof} \rangle$

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

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

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

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

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

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

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

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

**lemma** *psub-sub-fm*:  $\langle \text{psub } f\ (\text{sub-fm } g\ p) = \text{sub-fm } (\text{map-tm } f\ o\ g)\ (\text{psub } f\ p) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *map-tm-inst-single*:  $\langle (\text{map-tm } f\ o\ (u \gg \#))\ t = (\text{map-tm } f\ u \gg \#)\ t \rangle$   
 $\langle \text{proof} \rangle$

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

$\langle \text{proof} \rangle$

## 2.4 Terms

**primrec** *terms-tm* ::  $\langle 'f \text{ tm} \Rightarrow 'f \text{ tm set} \rangle$  **where**  
 $\langle \text{terms-tm } (\#n) = \{\#n\} \rangle$   
 $| \langle \text{terms-tm } (\text{Of } ts) = \{\text{Of } ts\} \cup \bigcup (\text{set } (\text{map } \text{terms-tm } ts)) \rangle$

**primrec** *terms-fm* ::  $\langle ('f, 'p) \text{ fm} \Rightarrow 'f \text{ tm set} \rangle$  **where**  
 $\langle \text{terms-fm } \perp = \{\} \rangle$   
 $| \langle \text{terms-fm } (\cdot - ts) = \bigcup (\text{set } (\text{map } \text{terms-tm } ts)) \rangle$   
 $| \langle \text{terms-fm } (p \longrightarrow q) = \text{terms-fm } p \cup \text{terms-fm } q \rangle$   
 $| \langle \text{terms-fm } (\forall p) = \text{terms-fm } p \rangle$

**definition** *terms* ::  $\langle ('f, 'p) \text{ fm set} \Rightarrow 'f \text{ tm set} \rangle$  **where**  
 $\langle \text{terms } S \equiv \bigcup p \in S. \text{terms-fm } p \rangle$

### 2.4.1 Lemmas

**lemma** *terms-mono*:  $\langle S \subseteq S' \Longrightarrow \text{terms } S \subseteq \text{terms } S' \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *terms-tm-refl* [*intro*]:  $\langle t \in \text{terms-tm } t \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *terms-tm-trans* [*trans*]:  $\langle s \in \text{terms-tm } t \Longrightarrow r \in \text{terms-tm } s \Longrightarrow r \in \text{terms-tm } t \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *map-terms-tm* [*simp*]:  $\langle \text{terms-tm } (\text{map-tm } f t) = \text{map-tm } f ' \text{terms-tm } t \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *map-terms-fm* [*simp*]:  $\langle \text{terms-fm } (\text{map-fm } f g p) = \text{map-tm } f ' \text{terms-fm } p \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *terms-tm-closed* [*dest*]:  $\langle t \in \text{terms-tm } s \Longrightarrow \text{terms-tm } t \subseteq \text{terms-tm } s \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *terms-fm-closed*:  $\langle t \in \text{terms-fm } p \Longrightarrow \text{terms-tm } t \subseteq \text{terms-fm } p \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *terms-source*:  $\langle t \in \text{terms } S \Longrightarrow \exists S' \subseteq S. \text{finite } S' \wedge t \in \text{terms } S' \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *terms-tm-Fun*:  $\langle \text{Of } ts \in \text{terms-tm } s \Longrightarrow t \in \text{set } ts \Longrightarrow t \in \text{terms-tm } s \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *terms-Fun*:  $\langle \text{Of } ts \in \text{terms } S \Longrightarrow \text{set } ts \subseteq \text{terms } S \rangle$   
 $\langle \text{proof} \rangle$

## 2.5 Guard

**definition** *guard* ::  $\langle 'a \Rightarrow 'a \text{ set} \Rightarrow 'a \rangle$  (**infix**  $\langle \in? \rangle$  50) **where**  
 $\langle x \in? S \equiv \text{if } x \in S \text{ then } x \text{ else } \text{SOME } y. y \in S \rangle$

**lemma** *guard-in*:  $\langle S \neq \{\} \implies (x \in? S) \in S \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *guard-refl* [*simp*]:  $\langle t \in S \implies t \in? S = t \rangle$   
 $\langle \text{proof} \rangle$

## 2.6 Model Existence

**inductive**

*confl-class* ::  $\langle ('f, 'p) \text{ fm list} \Rightarrow ('f, 'p) \text{ fm list} \Rightarrow \text{bool} \rangle$  (**infix**  $\langle \rightsquigarrow_{\mathbf{X}} \rangle$  50) **and**  
*alpha-class* ::  $\langle ('f, 'p) \text{ fm list} \Rightarrow ('f, 'p) \text{ fm list} \Rightarrow \text{bool} \rangle$  (**infix**  $\langle \rightsquigarrow_{\alpha} \rangle$  50) **and**  
*beta-class* ::  $\langle ('f, 'p) \text{ fm list} \Rightarrow ('f, 'p) \text{ fm list} \Rightarrow \text{bool} \rangle$  (**infix**  $\langle \rightsquigarrow_{\beta} \rangle$  50) **and**  
*gamma-class* ::  $\langle ('f, 'p) \text{ fm list} \Rightarrow ((f, 'p) \text{ fm set} \Rightarrow -) \times (f \text{ tm} \Rightarrow -) \Rightarrow \text{bool} \rangle$   
**(infix**  $\langle \rightsquigarrow_{\gamma} \rangle$  50)

**where**

*CFls* [*intro*]:  $\langle [ \perp ] \rightsquigarrow_{\mathbf{X}} [ \perp ] \rangle$   
 $|$  *CNeg* [*intro*]:  $\langle [ \neg (\cdot P \text{ ts}) ] \rightsquigarrow_{\mathbf{X}} [ \cdot P \text{ ts} ] \rangle$   
 $|$  *CImpN* [*intro*]:  $\langle [ \neg (p \longrightarrow q) ] \rightsquigarrow_{\alpha} [ p, \neg q ] \rangle$   
 $|$  *CImpP* [*intro*]:  $\langle [ p \longrightarrow q ] \rightsquigarrow_{\beta} [ \neg p, q ] \rangle$   
 $|$  *CAllP* [*intro*]:  $\langle [ \forall p ] \rightsquigarrow_{\gamma} (\text{terms}, \lambda t. [ \langle t \rangle p ]) \rangle$

**fun**  $\delta$  ::  $\langle ('f, 'p) \text{ fm} \Rightarrow 'f \Rightarrow ('f, 'p) \text{ fm list} \rangle$  **where**  
 $\langle \delta (\neg \forall p) x = [ \neg \langle \star x \rangle p ] \rangle$   
 $| \langle \delta - - = [] \rangle$

**interpretation** *P*: *Params psub params-fm*  $\langle \lambda -. \text{True} \rangle$   
 $\langle \text{proof} \rangle$

**interpretation** *C*: *Confl psub params-fm*  $\langle \lambda -. \text{True} \rangle$  *confl-class*  
 $\langle \text{proof} \rangle$

**interpretation** *A*: *Alpha psub params-fm*  $\langle \lambda -. \text{True} \rangle$  *alpha-class*  
 $\langle \text{proof} \rangle$

**interpretation** *B*: *Beta psub params-fm*  $\langle \lambda -. \text{True} \rangle$  *beta-class*  
 $\langle \text{proof} \rangle$

**interpretation** *G*: *Gamma map-tm psub params-fm*  $\langle \lambda -. \text{True} \rangle$  *gamma-class*  
 $\langle \text{proof} \rangle$

**interpretation** *D*: *Delta psub params-fm*  $\langle \lambda -. \text{True} \rangle$   $\delta$   
 $\langle \text{proof} \rangle$

**abbreviation** *Kinds* ::  $\langle ('f, ('f, 'p) \text{ fm}) \text{ kind list} \rangle$  **where**

$\langle Kinds \equiv [C.kind, A.kind, B.kind, G.kind, D.kind] \rangle$

**lemma** *prop<sub>E</sub>-Kinds* [intro]:

**assumes**  $\langle P.sat_E C.kind C \rangle \langle P.sat_E A.kind C \rangle \langle P.sat_E B.kind C \rangle \langle P.sat_E G.kind C \rangle \langle P.sat_E D.kind C \rangle$   
**shows**  $\langle P.prop_E Kinds C \rangle$   
 $\langle proof \rangle$

**interpretation** *Consistency-Kinds* *psub params-fm*  $\langle \lambda-. True \rangle Kinds$

$\langle proof \rangle$

**interpretation** *Maximal-Consistency* *psub params-fm*  $\langle \lambda-. True \rangle Kinds$

$\langle proof \rangle$

**abbreviation** *canonical* ::  $\langle ('f, 'p) fm set \Rightarrow ('f tm, 'f, 'p) model \rangle$  **where**

$\langle canonical H \equiv Model (terms H) (\lambda n. \#n \in ? terms H) (\lambda f ts. \bigcirc f ts \in ? terms H) (\lambda P ts. \cdot P ts \in H) \rangle$

**lemma** *wf-canonical*:

**assumes**  $\langle terms H \neq \{\} \rangle$   
**shows**  $\langle wf-model (canonical H) \rangle$   
 $\langle proof \rangle$

**lemma** *canonical-tm-id* [simp]:

$\langle t \in terms H \Longrightarrow \langle (\lambda n. \#n \in ? terms H, \lambda P ts. \bigcirc P ts \in ? terms H) \rangle t = t \rangle$   
 $\langle proof \rangle$

**lemma** *canonical-tm-id-map* [simp]:

$\langle set ts \subseteq terms H \Longrightarrow map \langle (\lambda n. \#n \in ? terms H, \lambda P ts. \bigcirc P ts \in ? terms H) \rangle ts = ts \rangle$   
 $\langle proof \rangle$

**locale** *MyHintikka* = *Hintikka* *psub params-fm*  $\langle \lambda-. True \rangle Kinds H$

**for**  $H :: \langle ('f, 'p) fm set \rangle +$   
**assumes** *terms-ne*:  $\langle terms H \neq \{\} \rangle$

**begin**

**lemmas**

*confl* = *sat<sub>H</sub>[of C.kind]* **and**  
*alpha* = *sat<sub>H</sub>[of A.kind]* **and**  
*beta* = *sat<sub>H</sub>[of B.kind]* **and**  
*gamma* = *sat<sub>H</sub>[of G.kind]* **and**  
*delta* = *sat<sub>H</sub>[of D.kind]*

**theorem** *model*:  $\langle (p \in H \longrightarrow canonical H \models p) \wedge (\neg p \in H \longrightarrow \neg canonical H \models p) \rangle$

$\langle proof \rangle$

**end**

**theorem** *model-existence*:

**fixes**  $S :: \langle ('f, 'p) \text{ fm set} \rangle$   
**assumes**  $\langle P.\text{prop}_E \text{ Kinds } C \rangle$   
**and**  $\langle S \in C \rangle$   
**and**  $\langle P.\text{enough-new } S \rangle$   
**and**  $\langle \text{terms } S \neq \{\} \rangle$   
**and**  $\langle p \in S \rangle$   
**shows**  $\langle \text{canonical } (mk\text{-mcs } C \ S) \models p \rangle$   
*\langle proof \rangle*

## 2.7 Compactness

**abbreviation** *semantics-set*  $:: \langle ('a, 'f, 'p) \text{ model} \Rightarrow ('f, 'p) \text{ fm set} \Rightarrow \text{bool} \rangle$  (**infix**  $\langle \models \rangle$  50) **where**  
 $\langle M \models S \equiv \forall p \in S. M \models p \rangle$

**lemma** *compact-C*:  $\langle P.\text{sat}_E \ C.\text{kind} \ \{S :: ('f, 'p) \text{ fm set}. P.\text{enough-new } S \wedge$   
 $(\forall S' \subseteq S. \text{finite } S' \longrightarrow (\exists (U :: 'a \text{ set}) \ E \ F \ G. \text{wf-model } (Model \ U \ E \ F \ G) \wedge$   
 $Model \ U \ E \ F \ G \models S')) \rangle$   
*\langle proof \rangle*

**lemma** *compact-A*:  $\langle P.\text{sat}_E \ A.\text{kind} \ \{S :: ('f, 'p) \text{ fm set}. P.\text{enough-new } S \wedge$   
 $(\forall S' \subseteq S. \text{finite } S' \longrightarrow (\exists (U :: 'a \text{ set}) \ E \ F \ G. \text{wf-model } (Model \ U \ E \ F \ G) \wedge$   
 $Model \ U \ E \ F \ G \models S')) \rangle$   
*\langle proof \rangle*

**lemma** *compact-B*:  $\langle P.\text{sat}_E \ B.\text{kind} \ \{S :: ('f, 'p) \text{ fm set}. P.\text{enough-new } S \wedge$   
 $(\forall S' \subseteq S. \text{finite } S' \longrightarrow (\exists (U :: 'a \text{ set}) \ E \ F \ G. \text{wf-model } (Model \ U \ E \ F \ G) \wedge$   
 $Model \ U \ E \ F \ G \models S')) \rangle$   
*\langle proof \rangle*

**lemma** *compact-G*:  $\langle P.\text{sat}_E \ G.\text{kind} \ \{S :: ('f, 'p) \text{ fm set}. P.\text{enough-new } S \wedge$   
 $(\forall S' \subseteq S. \text{finite } S' \longrightarrow (\exists (U :: 'a \text{ set}) \ E \ F \ G. \text{wf-model } (Model \ U \ E \ F \ G) \wedge$   
 $Model \ U \ E \ F \ G \models S')) \rangle$   
*\langle proof \rangle*

**lemma** *compact-D*:  $\langle P.\text{sat}_E \ D.\text{kind} \ \{S :: ('f, 'p) \text{ fm set}. P.\text{enough-new } S \wedge$   
 $(\forall S' \subseteq S. \text{finite } S' \longrightarrow (\exists (U :: 'a \text{ set}) \ E \ F \ G. \text{wf-model } (Model \ U \ E \ F \ G) \wedge$   
 $Model \ U \ E \ F \ G \models S')) \rangle$   
*\langle proof \rangle*

**lemma** *compact-prop*:  $\langle P.\text{prop}_E \ \text{Kinds} \ \{S :: ('f, 'p) \text{ fm set}. P.\text{enough-new } S \wedge$   
 $(\forall S' \subseteq S. \text{finite } S' \longrightarrow (\exists (U :: 'a \text{ set}) \ E \ F \ G. \text{wf-model } (Model \ U \ E \ F \ G) \wedge$   
 $Model \ U \ E \ F \ G \models S')) \rangle$   
*\langle proof \rangle*

**theorem** *compactness*:

**fixes**  $S :: \langle 'f, 'p \rangle \text{ fm set} \rangle$   
**assumes**  $\langle P.\text{enough-new } S \rangle$   
**shows**  $\langle (\exists (U :: 'f \text{ tm set}) E F G. \text{wf-model } (Model U E F G) \wedge Model U E F G \models S) \longleftrightarrow (\forall S' \subseteq S. \text{finite } S' \longrightarrow (\exists (U :: 'f \text{ tm set}) E F G. \text{wf-model } (Model U E F G) \wedge Model U E F G \models S')) \rangle$   
 $\langle \text{proof} \rangle$

## 2.8 Natural Deduction

**locale** *Natural-Deduction*

**begin**

**inductive** *ND-Set* ::  $\langle 'f, 'p \rangle \text{ fm set} \Rightarrow ('f, 'p) \text{ fm} \Rightarrow \text{bool} \rangle$  (**infix**  $\Vdash$  50) **where**  
*Assm* [*dest*]:  $\langle p \in A \Longrightarrow A \Vdash p \rangle$   
*FlsE* [*elim*]:  $\langle A \Vdash \perp \Longrightarrow A \Vdash p \rangle$   
*ImpI* [*intro*]:  $\langle \{p\} \cup A \Vdash q \Longrightarrow A \Vdash (p \longrightarrow q) \rangle$   
*ImpE* [*dest*]:  $\langle A \Vdash (p \longrightarrow q) \Longrightarrow A \Vdash p \Longrightarrow A \Vdash q \rangle$   
*UniI* [*intro*]:  $\langle A \Vdash (\star a)p \Longrightarrow a \notin P.\text{params } (\{p\} \cup A) \Longrightarrow A \Vdash \forall p \rangle$   
*UniE* [*dest*]:  $\langle A \Vdash \forall p \Longrightarrow t \in \text{terms } (\{p\} \cup A) \Longrightarrow A \Vdash \langle t \rangle p \rangle$   
*Clas*:  $\langle \{p \longrightarrow q\} \cup A \Vdash p \Longrightarrow A \Vdash p \rangle$

### 2.8.1 Soundness

**theorem** *soundness-set*:

**assumes**  $\langle A \Vdash p \rangle \langle \text{wf-model } (Model U E F G) \rangle$   
**shows**  $\langle \forall q \in A. Model U E F G \models q \Longrightarrow Model U E F G \models p \rangle$   
 $\langle \text{proof} \rangle$

### 2.8.2 Derivational Consistency

**lemma** *Boole*:  $\langle \{\neg p\} \cup A \Vdash \perp \Longrightarrow A \Vdash p \rangle$   
 $\langle \text{proof} \rangle$

**sublocale** *DC*: *Derivational-Confl* *psub* *params-fm*  $\langle \lambda-. True \rangle$  *confl-class*  $\langle \lambda A. \neg A \Vdash \perp \rangle$   
 $\langle \text{proof} \rangle$

**sublocale** *DA*: *Derivational-Alpha* *psub* *params-fm*  $\langle \lambda-. True \rangle$  *alpha-class*  $\langle \lambda A. \neg A \Vdash \perp \rangle$   
 $\langle \text{proof} \rangle$

**sublocale** *DB*: *Derivational-Beta* *psub* *params-fm*  $\langle \lambda-. True \rangle$  *beta-class*  $\langle \lambda A. \neg A \Vdash \perp \rangle$   
 $\langle \text{proof} \rangle$

**sublocale** *DG*: *Derivational-Gamma* *map-tm* *psub* *params-fm*  $\langle \lambda-. True \rangle$  *gamma-class*  $\langle \lambda A. \neg A \Vdash \perp \rangle$   
 $\langle \text{proof} \rangle$

**sublocale** *DD: Derivational-Delta* *psub params-fm*  $\langle \lambda-. \text{ True} \rangle \delta \langle \lambda A. \neg A \Vdash \perp \rangle$   
 $\langle \text{proof} \rangle$

**sublocale** *Derivational-Consistency* *psub params-fm*  $\langle \lambda-. \text{ True} \rangle$  *Kinds*  $\langle \lambda A. \neg A \Vdash \perp \rangle$   
 $\langle \text{proof} \rangle$

### 2.8.3 Strong Completeness

**lemma** *with-subterm-elim*:  $\langle A \Vdash \text{ with-subterm } p \implies A \Vdash p \rangle$   
 $\langle \text{proof} \rangle$

**theorem** *strong-completeness*:

**fixes**  $p :: \langle 'f, 'p \rangle \text{ fm} \rangle$

**assumes** *mod*:  $\langle \bigwedge (U :: 'f \text{ tm set}) E F G. \text{ wf-model } (\text{Model } U E F G) \implies$   
 $(\forall q \in A. \text{ Model } U E F G \models q) \implies \text{ Model } U E F G \models p \rangle$

**and**  $\langle P.\text{enough-new } A \rangle$

**shows**  $\langle A \Vdash p \rangle$

$\langle \text{proof} \rangle$

### 2.8.4 Natural Deduction with Lists

**inductive** *ND-List* ::  $\langle 'f, 'p \rangle \text{ fm list} \Rightarrow ('f, 'p) \text{ fm} \Rightarrow \text{bool} \rangle$  (**infix**  $\langle \vdash \rangle$  50) **where**

*Assm* [*simp*]:  $\langle p \in \text{ set } A \implies A \vdash p \rangle$

| *FlsE* [*elim*]:  $\langle A \vdash \perp \implies A \vdash p \rangle$

| *ImpI* [*intro*]:  $\langle p \# A \vdash q \implies A \vdash p \longrightarrow q \rangle$

| *ImpE* [*dest*]:  $\langle A \vdash p \longrightarrow q \implies A \vdash p \implies A \vdash q \rangle$

| *UniI* [*intro*]:  $\langle A \vdash \langle \star a \rangle p \implies a \notin P.\text{params } (\{p\} \cup \text{ set } A) \implies A \vdash \forall p \rangle$

| *UniE* [*dest*]:  $\langle A \vdash \forall p \implies t \in \text{ terms } (\{p\} \cup \text{ set } A) \implies A \vdash \langle t \rangle p \rangle$

| *Clas*:  $\langle (p \longrightarrow q) \# A \vdash p \implies A \vdash p \rangle$

**definition** *bounded* ::  $\langle 'a \text{ list} \Rightarrow 'a \text{ set} \Rightarrow ('a \text{ list} \Rightarrow \text{bool}) \Rightarrow \text{bool} \rangle$  **where**

$\langle \text{ bounded } K A P \equiv \text{ set } K \subseteq A \wedge (\forall B. \text{ set } K \subseteq \text{ set } B \longrightarrow \text{ set } B \subseteq A \longrightarrow P B) \rangle$

**lemma** *bounded-one* [*elim*]:

**assumes**  $\langle \text{ bounded } K A P \rangle \langle \bigwedge A. P A \implies Q A \rangle$

**shows**  $\langle \text{ bounded } K A Q \rangle$

$\langle \text{proof} \rangle$

**lemma** *bounded-two* [*elim*]:

**assumes**  $\langle \text{ bounded } K A P \rangle \langle \text{ bounded } K' A Q \rangle \langle \bigwedge A. P A \implies Q A \implies R A \rangle$

**shows**  $\langle \text{ bounded } (K @ K') A R \rangle$

$\langle \text{proof} \rangle$

**lemma** *bounded-removeAll* [*dest*]:

**assumes**  $\langle \text{ bounded } K (\{p\} \cup A) P \rangle$

**shows**  $\langle \text{ bounded } (\text{ remove.All } p K) A (\lambda B. P (p \# B)) \rangle$

$\langle \text{proof} \rangle$

**lemma** *bounded-terms*:

**assumes**  $\langle t \in \text{terms } (\{p\} \cup A) \rangle$   
**shows**  $\langle t \in \text{terms-fm } p \wedge \text{bounded } [] A (\lambda B. t \in \text{terms } (\text{set } (p \# B))) \vee$   
 $(\exists q \in A. t \in \text{terms-fm } q \wedge \text{bounded } [q] A (\lambda B. t \in \text{terms } (\text{set } (p \# B)))) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *bounded-params*:

**assumes**  $\langle a \notin P.\text{params } (\{p\} \cup A) \rangle \langle \text{bounded } K A P \rangle$   
**shows**  $\langle \text{bounded } K A (\lambda B. a \notin P.\text{params } (\text{set } (p \# B))) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *finite-kernel*:  $\langle A \Vdash p \implies \exists K. \text{bounded } K A (\lambda B. B \vdash p) \rangle$   
 $\langle \text{proof} \rangle$

**corollary** *finite-assumptions*:  $\langle A \Vdash p \implies \exists B. \text{set } B \subseteq A \wedge B \vdash p \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *to-set*:  $\langle A \vdash p \implies \text{set } A \Vdash p \rangle$   
 $\langle \text{proof} \rangle$

**corollary** *soundness-list*:

**assumes**  $\langle A \vdash p \rangle \langle \text{wf-model } (\text{Model } U E F G) \rangle \langle \forall q \in \text{set } A. \text{Model } U E F G \models q \rangle$   
**shows**  $\langle \text{Model } U E F G \models p \rangle$   
 $\langle \text{proof} \rangle$

**corollary** *soundness-nil*:  $\langle [] \vdash p \implies \text{wf-model } (\text{Model } U E F G) \implies \text{Model } U E F G \models p \rangle$   
 $\langle \text{proof} \rangle$

**corollary**  $\langle \neg ([] \vdash \perp) \rangle$   
 $\langle \text{proof} \rangle$

**corollary** *strong-completeness-list*:

**fixes**  $p :: \langle ('f, 'p) \text{ fm} \rangle$   
**assumes**  $\text{mod}: \langle \bigwedge (U :: 'f \text{ tm set}) E F G. \text{wf-model } (\text{Model } U E F G) \implies (\forall q \in A. \text{Model } U E F G \models q) \implies \text{Model } U E F G \models p \rangle$   
**and**  $\langle P.\text{enough-new } A \rangle$   
**shows**  $\langle \exists B. \text{set } B \subseteq A \wedge B \vdash p \rangle$   
 $\langle \text{proof} \rangle$

**theorem** *main*:

**fixes**  $p :: \langle ('f, 'p) \text{ fm} \rangle$   
**assumes**  $\langle |UNIV :: ('f, 'p) \text{ fm set}| \leq o \mid UNIV :: 'f \text{ set} \rangle$   
**shows**  $\langle [] \vdash p \iff (\forall (U :: 'f \text{ tm set}) E F G. \text{wf-model } (\text{Model } U E F G) \implies \text{Model } U E F G \models p) \rangle$   
 $\langle \text{proof} \rangle$

**end**

## 2.9 Tableau

**locale** *Tableau*

**begin**

**inductive** *TC* ::  $\langle ('f, 'p) \text{ fm set} \Rightarrow \text{bool} \rangle (\langle \vdash \rightarrow [51] 50 \rangle)$  **where**

*Axiom* [*simp*]:  $\langle \neg \cdot P \text{ ts} \in A \Longrightarrow \cdot P \text{ ts} \in A \Longrightarrow \vdash A \rangle$   
| *FlsP* [*simp*]:  $\langle \perp \in A \Longrightarrow \vdash A \rangle$   
| *FlsN* [*intro*]:  $\langle \vdash A \Longrightarrow \vdash \{ \neg \perp \} \cup A \rangle$   
| *ImpP* [*intro*]:  $\langle \vdash \{ \neg p \} \cup A \Longrightarrow \vdash \{ q \} \cup A \Longrightarrow \vdash \{ p \longrightarrow q \} \cup A \rangle$   
| *ImpN* [*intro*]:  $\langle \vdash \{ p, \neg q \} \cup A \Longrightarrow \vdash \{ \neg (p \longrightarrow q) \} \cup A \rangle$   
| *UniP* [*intro*]:  $\langle \vdash \{ \langle t \rangle p \} \cup A \Longrightarrow t \in \text{terms} (\{ p \} \cup A) \Longrightarrow \vdash \{ \forall p \} \cup A \rangle$   
| *UniN* [*intro*]:  $\langle \vdash \{ \neg \langle \star a \rangle p \} \cup A \Longrightarrow a \notin P.\text{params} (\{ p \} \cup A) \Longrightarrow \vdash \{ \neg \forall p \} \cup A \rangle$

### 2.9.1 Soundness

**theorem** *soundness*:

**assumes**  $\langle \vdash A \rangle \langle \text{wf-model} (\text{Model } U \ E \ F \ G) \rangle$

**shows**  $\langle \exists q \in A. \neg \text{Model } U \ E \ F \ G \models q \rangle$

$\langle \text{proof} \rangle$

### 2.9.2 Derivational Consistency

**sublocale** *DC*: *Derivational-Confl* *psub params-fm*  $\langle \lambda -. \text{True} \rangle$  *confl-class*  $\langle \lambda A. \neg \vdash A \rangle$

$\langle \text{proof} \rangle$

**sublocale** *DA*: *Derivational-Alpha* *psub params-fm*  $\langle \lambda -. \text{True} \rangle$  *alpha-class*  $\langle \lambda A. \neg \vdash A \rangle$

$\langle \text{proof} \rangle$

**sublocale** *DB*: *Derivational-Beta* *psub params-fm*  $\langle \lambda -. \text{True} \rangle$  *beta-class*  $\langle \lambda A. \neg \vdash A \rangle$

$\langle \text{proof} \rangle$

**sublocale** *DG*: *Derivational-Gamma* *map-tm* *psub params-fm*  $\langle \lambda -. \text{True} \rangle$  *gamma-class*  $\langle \lambda A. \neg \vdash A \rangle$

$\langle \text{proof} \rangle$

**sublocale** *DD*: *Derivational-Delta* *psub params-fm*  $\langle \lambda -. \text{True} \rangle$   $\delta$   $\langle \lambda A. \neg \vdash A \rangle$

$\langle \text{proof} \rangle$

**sublocale** *Derivational-Consistency* *psub params-fm*  $\langle \lambda -. \text{True} \rangle$  *Kinds*  $\langle \lambda A. \neg \vdash A \rangle$

$\langle \text{proof} \rangle$

### 2.9.3 Strong Completeness

**theorem** *strong-completeness*:

```

fixes  $p :: \langle 'f, 'p \rangle \text{ fm} \rangle$ 
assumes  $\text{mod}: \langle \bigwedge (U :: 'f \text{ tm set}) E F G. \text{wf-model } (\text{Model } U E F G) \implies (\forall q \in$ 
 $A. \text{Model } U E F G \models q) \implies \text{Model } U E F G \models p \rangle$ 
and  $\langle P.\text{enough-new } A \rangle$ 
shows  $\langle \vdash \{ \neg \text{with-subterm } p \} \cup A \rangle$ 
 $\langle \text{proof} \rangle$ 

end

end

```

## Chapter 3

# Example: Second-Order Logic

Generalizes [4] and [5] from first-order logic to second-order logic

```
theory Example-SOL imports  
  Abstract-Consistency-Property  
begin
```

### 3.1 Syntax

```
datatype (params-sym: 'f) sym  
  = VarS nat (<#>)  
  | SymS 'f (<O>)
```

```
datatype (params-tm: 'f) tm  
  = Var nat (<#>)  
  | Fun <'f sym> <'f tm list> (<O>)  
  | Cst 'f (<★>)
```

```
datatype (params-fm: 'f) fm  
  = Falsity (<⊥>)  
  | is-Pre: Pre <'f sym> <'f tm list> (<⋅>)  
  | Imp <'f fm> <'f fm> (infixr <⟶> 55)  
  | Uni <'f fm> (<∀>)  
  | UniP <'f fm> (<∀P>)  
  | UniF <'f fm> (<∀F>)
```

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

**abbreviation** And (**infix** <∧> 50) **where** <p ∧ q ≡ ¬ (p ⟶ ¬ q)>

**abbreviation** Iff (**infix** <⟷> 50) **where** <p ⟷ q ≡ (p ⟶ q) ∧ (q ⟶ p)>

**abbreviation** Eql (<- = ->) **where** <t1 = t2 ≡ (∀<sub>P</sub> ((.<#> 0) [t1]) ⟷ (<#>))>

$0) [t2]))\rangle$

**abbreviation**  $ExiF$  ( $\langle \exists_F \rangle$ ) **where**  $\langle \exists_F p \equiv \neg(\forall_F(\neg p)) \rangle$

**abbreviation**  $ExiP$  ( $\langle \exists_P \rangle$ ) **where**  $\langle \exists_P p \equiv \neg(\forall_P(\neg p)) \rangle$

**term**  $\langle \forall(\perp \longrightarrow (\cdot(\circ_2 ''P'') [\circ(\circ_2 ''f'') [\#0]])) \rangle$

## 3.2 Semantics

**definition** *shift* ( $\langle \cdot \langle - \rangle \rangle$ ) **where**

$\langle E \langle n : x \rangle m \equiv \text{if } m < n \text{ then } E m \text{ else if } m = n \text{ then } x \text{ else } E(m-1) \rangle$

**primrec** *semantics-fn* ( $\langle \langle -, - \rangle_2 \rangle$ ) **where**

$\langle \langle E_F, F \rangle_2 (\#_2 n) = E_F n \rangle$   
 $| \langle \langle E_F, F \rangle_2 (\circ_2 f) = F f \rangle$

**primrec** *semantics-tm* ( $\langle \langle -, -, -, - \rangle \rangle$ ) **where**

$\langle \langle E, E_F, C, F \rangle (\#n) = E n \rangle$   
 $| \langle \langle E, E_F, C, F \rangle (\circ f ts) = (\langle E_F, F \rangle_2 f) (\text{map } \langle E, E_F, C, F \rangle ts) \rangle$   
 $| \langle \langle E, E_F, C, F \rangle (\star c) = C c \rangle$

**fun** *semantics-fm* (**infix**  $\langle \models \rangle$  50) **where**

$\langle \langle (-, -, -, -, -, -, -) \models \perp \rangle = \text{False} \rangle$   
 $| \langle \langle (E, E_F, E_P, C, F, G, PS, FS) \models \cdot P ts \rangle = \langle E_P, G \rangle_2 P (\text{map } \langle E, E_F, C, F \rangle ts) \rangle$   
 $| \langle \langle (E, E_F, E_P, C, F, G, PS, FS) \models p \longrightarrow q \rangle = \langle (E, E_F, E_P, C, F, G, PS, FS) \models p \longrightarrow (E, E_F, E_P, C, F, G, PS, FS) \models q \rangle \rangle$   
 $| \langle \langle (E, E_F, E_P, C, F, G, PS, FS) \models \forall p \rangle = \langle \forall x. (E \langle 0 : x \rangle, E_F, E_P, C, F, G, PS, FS) \models p \rangle \rangle$   
 $| \langle \langle (E, E_F, E_P, C, F, G, PS, FS) \models \forall_P p \rangle = \langle \forall x \in PS. (E, E_F, E_P \langle 0 : x \rangle, C, F, G, PS, FS) \models p \rangle \rangle$   
 $| \langle \langle (E, E_F, E_P, C, F, G, PS, FS) \models \forall_{FP} p \rangle = \langle \forall x \in FS. (E, E_F \langle 0 : x \rangle, E_P, C, F, G, PS, FS) \models p \rangle \rangle$

**proposition**  $\langle (E, E_F, E_P, C, F, G, PS, FS) \models (\forall (\cdot P [\# 0]) \longrightarrow \cdot P [\star a]) \rangle$   
 $\langle \text{proof} \rangle$

## 3.3 Operations

### 3.3.1 Shift

**context** *fixes*  $n m :: \text{nat}$  **begin**

**lemma** *shift-eq* [*simp*]:  $\langle n = m \implies E \langle n : x \rangle m = x \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *shift-gt* [*simp*]:  $\langle m < n \implies E \langle n : x \rangle m = E m \rangle$

$\langle \text{proof} \rangle$

**lemma** *shift-lt* [*simp*]:  $\langle n < m \implies E\langle n:x \rangle m = E(m-1) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *shift-commute* [*simp*]:  $\langle (E\langle n:y \rangle\langle 0:x \rangle) = (E\langle 0:x \rangle\langle n+1:y \rangle) \rangle$   
 $\langle \text{proof} \rangle$

**end**

### 3.3.2 Parameters

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

**lemma** *upd-params-sym* [*simp*]:  $\langle f \notin \text{params-sym } fn \implies \langle E_F, F(f := x) \rangle_2 fn = \langle E_F, F \rangle_2 fn \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *upd-params-tm* [*simp*]:  $\langle f \notin \text{params-tm } t \implies \langle E, E_F, C, F(f := x) \rangle t = \langle E, E_F, C, F \rangle t \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *upd-params-tm-c* [*simp*]:  $\langle c \notin \text{params-tm } t \implies \langle E, E_F, C(c := x), F \rangle t = \langle E, E_F, C, F \rangle t \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *upd-params-fm* [*simp*]:  $\langle f \notin \text{params-fm } p \implies (E, E_F, E_P, C, F(f := x), G, PS, FS) \models p \iff (E, E_F, E_P, C, F, G, PS, FS) \models p \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *upd-params-fm-c* [*simp*]:  $\langle c \notin \text{params-fm } p \implies (E, E_F, E_P, C(c := x), F, G, PS, FS) \models p \iff (E, E_F, E_P, C, F, G, PS, FS) \models p \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *upd-params-fm-G* [*simp*]:  $\langle P \notin \text{params-fm } p \implies (E, E_F, E_P, C, F, G(P := x), PS, FS) \models p \iff (E, E_F, E_P, C, F, G, PS, FS) \models p \rangle$   
 $\langle \text{proof} \rangle$

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

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

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

### 3.3.3 Instantiation

**primrec** *lift-tm* ( $\langle \uparrow \rangle$ ) **where**

$$\begin{aligned}
& \langle \uparrow(\#n) = \#(n+1) \rangle \\
| & \langle \uparrow(\circ f \ ts) = \circ f \ (\text{map } \uparrow \ ts) \rangle \\
| & \langle \uparrow(\star c) = \star c \rangle
\end{aligned}$$

**primrec lift-sym** ( $\langle \uparrow_2 \rangle$ ) **where**

$$\begin{aligned}
& \langle \uparrow_2(\#_2 \ n) = \#_2 \ (n+1) \rangle \\
| & \langle \uparrow_2(\circ_2 \ p) = \circ_2 \ p \rangle
\end{aligned}$$

**primrec lift-fn** ( $\langle \uparrow_F \rangle$ ) **where**

$$\begin{aligned}
& \langle \uparrow_F(\#n) = \#n \rangle \\
| & \langle \uparrow_F(\circ f \ ts) = \circ(\uparrow_2 \ f) \ (\text{map } \uparrow_F \ ts) \rangle \\
| & \langle \uparrow_F(\star c) = \star c \rangle
\end{aligned}$$

**primrec inst-tm** ( $\langle \llbracket -' / - \rrbracket \rangle$ ) **where**

$$\begin{aligned}
& \langle \llbracket s/m \rrbracket(\#n) = (\text{if } n < m \text{ then } \#n \text{ else if } n = m \text{ then } s \text{ else } \#(n-1)) \rangle \\
| & \langle \llbracket s/m \rrbracket(\circ f \ ts) = \circ f \ (\text{map } \llbracket s/m \rrbracket \ ts) \rangle \\
| & \langle \llbracket s/m \rrbracket(\star c) = \star c \rangle
\end{aligned}$$

**primrec inst-sym** ( $\langle \llbracket -' / - \rrbracket_2 \rangle$ ) **where**

$$\begin{aligned}
& \langle \llbracket s/m \rrbracket_2(\#_2 \ n) = (\text{if } n < m \text{ then } \#_2 \ n \text{ else if } n = m \text{ then } s \text{ else } \#_2 \ (n-1)) \rangle \\
| & \langle \llbracket s/m \rrbracket_2(\circ_2 \ p) = \circ_2 \ p \rangle
\end{aligned}$$

**primrec inst-fn** ( $\langle \llbracket -' / - \rrbracket_F \rangle$ ) **where**

$$\begin{aligned}
& \langle \llbracket s/m \rrbracket_F(\#n) = (\#n) \rangle \\
| & \langle \llbracket s/m \rrbracket_F(\circ f \ ts) = \circ(\llbracket s/m \rrbracket_2 \ f) \ (\text{map } \llbracket s/m \rrbracket_F \ ts) \rangle \\
| & \langle \llbracket s/m \rrbracket_F(\star c) = (\star c) \rangle
\end{aligned}$$

**primrec inst-fm** ( $\langle \langle -' / - \rangle \rangle$ ) **where**

$$\begin{aligned}
& \langle \langle -' / - \rangle \perp = \perp \rangle \\
| & \langle \langle s/m \rangle(\cdot P \ ts) = \cdot P \ (\text{map } \llbracket s/m \rrbracket \ ts) \rangle \\
| & \langle \langle s/m \rangle(p \longrightarrow q) = \langle s/m \rangle p \longrightarrow \langle s/m \rangle q \rangle \\
| & \langle \langle s/m \rangle(\forall p) = \forall(\langle \uparrow s/m+1 \rangle p) \rangle \\
| & \langle \langle s/m \rangle(\forall_P p) = \forall_P(\langle s/m \rangle p) \rangle \\
| & \langle \langle s/m \rangle(\forall_F p) = \forall_F(\langle \uparrow_F s/m \rangle p) \rangle
\end{aligned}$$

**primrec inst-fm-P** ( $\langle \langle -' / - \rangle_P \rangle$ ) **where**

$$\begin{aligned}
& \langle \langle -' / - \rangle_P \perp = \perp \rangle \\
| & \langle \langle s/m \rangle_P(\cdot P \ ts) = \cdot(\llbracket s/m \rrbracket_2 \ P) \ ts \rangle \\
| & \langle \langle s/m \rangle_P(p \longrightarrow q) = \langle s/m \rangle_P p \longrightarrow \langle s/m \rangle_P q \rangle \\
| & \langle \langle s/m \rangle_P(\forall p) = \forall(\langle s/m \rangle_P p) \rangle \\
| & \langle \langle s/m \rangle_P(\forall_P p) = \forall_P(\langle \uparrow_2 s/m+1 \rangle_P p) \rangle \\
| & \langle \langle s/m \rangle_P(\forall_F p) = \forall_F(\langle s/m \rangle_P p) \rangle
\end{aligned}$$

**primrec inst-fm-F** ( $\langle \langle -' / - \rangle_F \rangle$ ) **where**

$$\begin{aligned}
& \langle \langle -' / - \rangle_F \perp = \perp \rangle \\
| & \langle \langle s/m \rangle_F(\cdot P \ ts) = \cdot P \ (\text{map } \llbracket s/m \rrbracket_F \ ts) \rangle \\
| & \langle \langle s/m \rangle_F(p \longrightarrow q) = \langle s/m \rangle_F p \longrightarrow \langle s/m \rangle_F q \rangle \\
| & \langle \langle s/m \rangle_F(\forall p) = \forall(\langle s/m \rangle_F p) \rangle \\
| & \langle \langle s/m \rangle_F(\forall_P p) = \forall_P(\langle s/m \rangle_F p) \rangle
\end{aligned}$$

$$| \langle \langle s/m \rangle_F (\forall_F p) = \forall_F (\langle \uparrow_2 s/m+1 \rangle_F p) \rangle$$

**lemma** *lift-lemma* [simp]:  $\langle \langle E \langle 0:x \rangle, E_F, C, F \rangle (\uparrow t) = \langle E, E_F, C, F \rangle t \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *lift-lemma-P* [simp]:  $\langle \langle E_P \langle 0:x \rangle, G \rangle_2 (\uparrow_2 P) = \langle E_P, G \rangle_2 P \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *lift-lemma-F* [simp]:  $\langle \langle E, E_F \langle 0:x \rangle, C, F \rangle (\uparrow_F tm) = \langle E, E_F, C, F \rangle tm \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *inst-tm-semantic* [simp]:  $\langle \langle E, E_F, C, F \rangle (\langle \langle s/m \rangle t) = \langle E \langle m: \langle E, E_F, C, F \rangle s \rangle, E_F, C, F \rangle t \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *inst-sym-semantic* [simp]:  $\langle \langle E_F, G \rangle_2 (\langle \langle s/m \rangle_2 fn) = \langle E_F \langle m: \langle E_F, G \rangle_2 s \rangle, G \rangle_2 fn \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *inst-tm-semantic-F* [simp]:  $\langle \langle E, E_F, C, F \rangle (\langle \langle s/m \rangle_F t) = \langle E, E_F \langle m: \langle E_F, F \rangle_2 s \rangle, C, F \rangle t \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *inst-fm-semantic-F* [simp]:  
 $\langle (E, E_F, E_P, C, F, G, PS, FS) \models (\langle t/m \rangle_F p) \longleftrightarrow (E, E_F \langle m: \langle E_F, F \rangle_2 t \rangle, E_P, C, F, G, PS, FS) \models p \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *inst-fm-semantic* [simp]:  
 $\langle (E, E_F, E_P, C, F, G, PS, FS) \models (\langle t/m \rangle p) \longleftrightarrow (E \langle m: \langle E, E_F, C, F \rangle t \rangle, E_F, E_P, C, F, G, PS, FS) \models p \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *inst-fm-semantic-P* [simp]:  $\langle (E, E_F, E_P, C, F, G, PS, FS) \models (\langle P/m \rangle_P p) \longleftrightarrow (E, E_F, E_P \langle m: \langle E_P, G \rangle_2 P \rangle, C, F, G, PS, FS) \models p \rangle$   
 $\langle \text{proof} \rangle$

### 3.3.4 Size

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

**primrec** *size-fm* where

$$\begin{aligned} & \langle \text{size-fm } \perp = 1 \rangle \\ & | \langle \text{size-fm } (\cdot -) = 1 \rangle \\ & | \langle \text{size-fm } (p \longrightarrow q) = 1 + \text{size-fm } p + \text{size-fm } q \rangle \\ & | \langle \text{size-fm } (\forall p) = 1 + \text{size-fm } p \rangle \\ & | \langle \text{size-fm } (\forall_P p) = 1 + \text{size-fm } p \rangle \\ & | \langle \text{size-fm } (\forall_F p) = 1 + \text{size-fm } p \rangle \end{aligned}$$

**lemma** *size-inst-fm* [simp]:  $\langle \text{size-fm } (\langle t/m \rangle p) = \text{size-fm } p \rangle$

$\langle \text{proof} \rangle$

**lemma** *size-inst-fm-P* [*simp*]:  $\langle \text{size-fm } (\langle t/m \rangle_P p) = \text{size-fm } p \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *size-inst-fm-F* [*simp*]:  $\langle \text{size-fm } (\langle t/m \rangle_F p) = \text{size-fm } p \rangle$   
 $\langle \text{proof} \rangle$

### 3.4 Model Existence

**inductive** *confl-class* ::  $\langle 'f \text{ fm list} \Rightarrow 'f \text{ fm list} \Rightarrow \text{bool} \rangle$  (**infix**  $\langle \rightsquigarrow_{\mathbf{x}} \rangle$  50) **where**  
  *CFIs*:  $\langle [ \perp ] \rightsquigarrow_{\mathbf{x}} [ \perp ] \rangle$   
  | *CNeg*:  $\langle [ \neg (\cdot P \text{ ts}) ] \rightsquigarrow_{\mathbf{x}} [ \cdot P \text{ ts} ] \rangle$

**inductive** *alpha-class* ::  $\langle 'f \text{ fm list} \Rightarrow 'f \text{ fm list} \Rightarrow \text{bool} \rangle$  (**infix**  $\langle \rightsquigarrow_{\alpha} \rangle$  50) **where**  
  *CImpN*:  $\langle [ \neg (p \longrightarrow q) ] \rightsquigarrow_{\alpha} [ p, \neg q ] \rangle$

**inductive** *beta-class* ::  $\langle 'f \text{ fm list} \Rightarrow 'f \text{ fm list} \Rightarrow \text{bool} \rangle$  (**infix**  $\langle \rightsquigarrow_{\beta} \rangle$  50) **where**  
  *CImpP*:  $\langle [ p \longrightarrow q ] \rightsquigarrow_{\beta} [ \neg p, q ] \rangle$

**inductive** *gamma-class* ::  $\langle 'f \text{ fm list} \Rightarrow ('f \text{ tm} \Rightarrow 'f \text{ fm list}) \Rightarrow \text{bool} \rangle$  (**infix**  $\langle \rightsquigarrow_{\gamma} \rangle$  50) **where**  
  *CALLP*:  $\langle [ \forall p ] \rightsquigarrow_{\gamma} (\lambda t. [ \langle t/0 \rangle p ]) \rangle$

**inductive** *gamma-class-P* ::  $\langle 'f \text{ fm list} \Rightarrow ('f \text{ sym} \Rightarrow 'f \text{ fm list}) \Rightarrow \text{bool} \rangle$  (**infix**  $\langle \rightsquigarrow_{\gamma P} \rangle$  50) **where**  
  *CALLPP*:  $\langle [ \forall_P p ] \rightsquigarrow_{\gamma P} (\lambda s. [ \langle s/0 \rangle_P p ]) \rangle$

**inductive** *gamma-class-F* ::  $\langle 'f \text{ fm list} \Rightarrow ('f \text{ sym} \Rightarrow 'f \text{ fm list}) \Rightarrow \text{bool} \rangle$  (**infix**  $\langle \rightsquigarrow_{\gamma F} \rangle$  50) **where**  
  *CALLFP*:  $\langle [ \forall_F p ] \rightsquigarrow_{\gamma F} (\lambda s. [ \langle s/0 \rangle_F p ]) \rangle$

**fun**  $\delta$  ::  $\langle 'f \text{ fm} \Rightarrow 'f \Rightarrow 'f \text{ fm list} \rangle$  **where**  
  *CALLN*:  $\langle \delta (\neg \forall p) x = [ \neg \langle \star x/0 \rangle p ] \rangle$   
  | *CALL2PN*:  $\langle \delta (\neg \forall_P p) x = [ \neg \langle \text{O}_2 x/0 \rangle_P p ] \rangle$   
  | *CALL2FN*:  $\langle \delta (\neg \forall_F p) x = [ \neg \langle \text{O}_2 x/0 \rangle_F p ] \rangle$   
  | *NOMATCH*:  $\langle \delta - - = [] \rangle$

**interpretation** *P*: *Params map-fm params-fm*  $\langle \lambda -. \text{True} \rangle$   
 $\langle \text{proof} \rangle$

**interpretation** *C*: *Confl map-fm params-fm*  $\langle \lambda -. \text{True} \rangle$  *confl-class*  
 $\langle \text{proof} \rangle$

**interpretation** *A*: *Alpha map-fm params-fm*  $\langle \lambda -. \text{True} \rangle$  *alpha-class*  
 $\langle \text{proof} \rangle$

**interpretation** *B*: *Beta map-fm params-fm*  $\langle \lambda -. \text{True} \rangle$  *beta-class*  
 $\langle \text{proof} \rangle$

**lemma** *map-tm-inst-tm* [*simp*]:

$$\text{map-tm } f \ (\langle\langle t/n \rangle\rangle x) = \langle\langle \text{map-tm } f \ t/n \rangle\rangle (\text{map-tm } f \ x)$$

*<proof>*

**lemma** *map-tm-lift-tm* [*simp*]:  $\text{map-tm } f \ (\uparrow t) = \uparrow (\text{map-tm } f \ t)$

*<proof>*

**lemma** *map-sym-lift-fn* [*simp*]:  $\langle \text{map-sym } f \ (\uparrow_2 t) = \uparrow_2 (\text{map-sym } f \ t) \rangle$

*<proof>*

**lemma** *map-tm-lift-fn* [*simp*]:  $\text{map-tm } f \ (\uparrow_F t) = \uparrow_F (\text{map-tm } f \ t)$

*<proof>*

**lemma** *map-fm-inst-single* [*simp*]:  $\langle \text{map-fm } f \ (\langle t/m \rangle p) = \langle \text{map-tm } f \ t/m \rangle (\text{map-fm } f \ p) \rangle$

*<proof>*

**lemma** *map-sym-inst-sym* [*simp*]:  $\langle \text{map-sym } f \ (\langle\langle t/m \rangle\rangle_2 p) = \langle\langle \text{map-sym } f \ t/m \rangle\rangle_2 (\text{map-sym } f \ p) \rangle$

*<proof>*

**lemma** *psub-inst-single'* [*simp*]:  $\langle \text{map-fm } f \ (\langle t/m \rangle_P p) = \langle \text{map-sym } f \ t/m \rangle_P (\text{map-fm } f \ p) \rangle$

*<proof>*

**lemma** *map-tm-inst-fn* [*simp*]:  $\langle \text{map-tm } f \ (\langle\langle t/m \rangle\rangle_F s) = \langle\langle \text{map-sym } f \ t/m \rangle\rangle_F (\text{map-tm } f \ s) \rangle$

*<proof>*

**lemma** *psub-inst-single''* [*simp*]:  $\langle \text{map-fm } f \ (\langle t/m \rangle_F p) = \langle \text{map-sym } f \ t/m \rangle_F (\text{map-fm } f \ p) \rangle$

*<proof>*

**interpretation** *G*: *Gamma-UNIV map-tm map-fm params-fm*  $\langle \lambda-. \text{True} \rangle$  *gamma-class*

*<proof>*

**interpretation** *G<sub>P</sub>*: *Gamma-UNIV map-sym map-fm params-fm*  $\langle \lambda-. \text{True} \rangle$  *gamma-class-P*

*<proof>*

**interpretation** *G<sub>F</sub>*: *Gamma-UNIV map-sym map-fm params-fm*  $\langle \lambda-. \text{True} \rangle$  *gamma-class-F*

*<proof>*

**interpretation** *D*: *Delta map-fm params-fm*  $\langle \lambda-. \text{True} \rangle$   $\delta$

*<proof>*

**abbreviation** *Kinds* ::  $\langle ('x, 'x \text{ fm}) \text{ kind list} \rangle$  **where**

$$\langle \text{Kinds} \equiv [C.\text{kind}, A.\text{kind}, B.\text{kind}, G.\text{kind}, G_P.\text{kind}, G_F.\text{kind}, D.\text{kind}] \rangle$$

**lemma** *prop<sub>E</sub>-Kinds*:

**assumes**  $\langle P.\text{sat}_E C.\text{kind } C \rangle \langle P.\text{sat}_E A.\text{kind } C \rangle \langle P.\text{sat}_E B.\text{kind } C \rangle \langle P.\text{sat}_E G.\text{kind } C \rangle \langle P.\text{sat}_E G_P.\text{kind } C \rangle$   
 $\langle P.\text{sat}_E G_F.\text{kind } C \rangle \langle P.\text{sat}_E D.\text{kind } C \rangle$   
**shows**  $\langle P.\text{prop}_E \text{Kinds } C \rangle$   
 $\langle \text{proof} \rangle$

**interpretation** *Consistency-Kinds map-fm params-fm*  $\langle \lambda-. \text{True} \rangle \text{Kinds}$   
 $\langle \text{proof} \rangle$

**interpretation** *Maximal-Consistency map-fm params-fm*  $\langle \lambda-. \text{True} \rangle \text{Kinds}$   
 $\langle \text{proof} \rangle$

**abbreviation** *henv<sub>P</sub>* **where**  $\text{henv}_P H == \lambda n \text{ts}. \cdot(\#_2 n) \text{ts} \in H$

**abbreviation** *hpred* **where**  $\text{hpred } H == \lambda P \text{ts}. \cdot(\circ_2 P) \text{ts} \in H$

**abbreviation** *hdom<sub>P</sub>* **where**  $\text{hdom}_P H == \text{range } (\text{henv}_P H) \cup \text{range } (\text{hpred } H)$

**abbreviation** *henv<sub>F</sub>* **where**  $\text{henv}_F == \lambda f. \circ (\#_2 f)$

**abbreviation** *hfun* **where**  $\text{hfun} == \lambda f. \circ (\circ_2 f)$

**definition** *hdom<sub>F</sub>* **where**  $\text{hdom}_F == \text{range } \text{henv}_F \cup \text{range } \text{hfun}$

**abbreviation** (*input*) *hmodel*  $\langle \llbracket - \rrbracket \rangle$  **where**  $\llbracket H \rrbracket \equiv (\#, \text{henv}_F, \text{henv}_P H, \star, \text{hfun}, \text{hpred } H, \text{hdom}_P H, \text{hdom}_F)$

**lemma** *semantics-tm-id [simp]*:  $\langle \llbracket \#, \text{henv}_F, \star, \lambda f. \circ (\circ_2 f) \rrbracket t = t \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *semantics-tm-id-map [simp]*:  $\langle \text{map } \llbracket \#, \lambda f. \circ (\#_2 f), \star, \lambda f. \circ (\circ_2 f) \rrbracket \text{ts} = \text{ts} \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *semantics-fn-h [simp]*:  $\langle \llbracket \text{henv}_P S, \text{hpred } S \rrbracket_2 P \text{ts} \longleftrightarrow \cdot P \text{ts} \in S \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *canonical-henv<sub>P</sub>*:

$\langle \forall t. (\#, \text{henv}_F, (\text{henv}_P S) \langle 0: \llbracket \text{henv}_P S, \text{hpred } S \rrbracket_2 t \rangle, \star, \text{hfun}, \text{hpred } S, \text{hdom}_P S, \text{hdom}_F) \models p \implies$   
 $(\#, \text{henv}_F, (\text{henv}_P S) \langle 0: \text{henv}_P S n \rangle, \star, \text{hfun}, \text{hpred } S, \text{hdom}_P S, \text{hdom}_F) \models p \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *canonical-hpred*:

$\langle \forall t. (\#, \text{henv}_F, (\text{henv}_P S) \langle 0: \llbracket \text{henv}_P S, \text{hpred } S \rrbracket_2 t \rangle, \star, \text{hfun}, \text{hpred } S, \text{hdom}_P S, \text{hdom}_F) \models p \implies$   
 $(\#, \text{henv}_F, (\text{henv}_P S) \langle 0: \text{hpred } S P \rangle, \star, \text{hfun}, \text{hpred } S, \text{hdom}_P S, \text{hdom}_F) \models p \rangle$

$p$   
 $\langle \text{proof} \rangle$

**lemma** *canonical-henv<sub>F</sub>*:

$\langle \forall t. (\#, \text{henv}_F \langle 0: \langle \text{henv}_F, \text{hfun} \rangle_2 t \rangle, \text{henv}_P S, \star, \text{hfun}, \text{hpred } S, \text{hdom}_P S, \text{hdom}_F) \models p \implies$   
 $(\#, \text{henv}_F \langle 0: \text{henv}_F f \rangle, \text{henv}_P S, \star, \text{hfun}, \text{hpred } S, \text{hdom}_P S, \text{hdom}_F) \models p \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *canonical-hfun*:

$\langle \forall t. (\#, \text{henv}_F \langle 0: \langle \text{henv}_F, \text{hfun} \rangle_2 t \rangle, \text{henv}_P S, \star, \text{hfun}, \text{hpred } S, \text{hdom}_P S, \text{hdom}_F) \models p \implies$   
 $(\#, \text{henv}_F \langle 0: \text{hfun } f \rangle, \text{henv}_P S, \star, \text{hfun}, \text{hpred } S, \text{hdom}_P S, \text{hdom}_F) \models p \rangle$   
 $\langle \text{proof} \rangle$

**locale** *MyHintikka* = *Hintikka map-fm params-fm*  $\langle \lambda -. \text{True} \rangle$  *Kinds S for S* ::  $\langle 'x$   
*fm set*  
**begin**

**lemmas**

*confl* = *sat<sub>H</sub>*[*of C.kind*] **and**  
*alpha* = *sat<sub>H</sub>*[*of A.kind*] **and**  
*beta* = *sat<sub>H</sub>*[*of B.kind*] **and**  
*gammaFO* = *sat<sub>H</sub>*[*of G.kind*] **and**  
*gamma2P* = *sat<sub>H</sub>*[*of G<sub>P</sub>.kind*] **and**  
*gamma2F* = *sat<sub>H</sub>*[*of G<sub>F</sub>.kind*] **and**  
*delta* = *sat<sub>H</sub>*[*of D.kind*]

**theorem** *model*:  $\langle (p \in S \longrightarrow \llbracket S \rrbracket \models p) \wedge (\neg p \in S \longrightarrow \neg \llbracket S \rrbracket \models p) \rangle$   
 $\langle \text{proof} \rangle$

**end**

**theorem** *model-existence*:

**fixes** *S* ::  $\langle 'x \text{ fm set} \rangle$   
**assumes**  $\langle P.\text{prop}_E \text{ Kinds } C \rangle$   
**and**  $\langle S \in C \rangle$   
**and**  $\langle P.\text{enough-new } S \rangle$   
**and**  $\langle p \in S \rangle$   
**shows**  $\langle \llbracket \text{mk-mcs } C \ S \rrbracket \models p \rangle$   
 $\langle \text{proof} \rangle$

### 3.5 Propositional Semantics

**primrec** *boolean where*

$\langle \text{boolean } - - \perp = \text{False} \rangle$   
 $| \langle \text{boolean } G - (\cdot P \text{ ts}) = G \ P \ \text{ts} \rangle$   
 $| \langle \text{boolean } G \ A \ (p \longrightarrow q) = (\text{boolean } G \ A \ p \longrightarrow \text{boolean } G \ A \ q) \rangle$   
 $| \langle \text{boolean } - \ A \ (\forall p) = A \ (\forall p) \rangle$

|  $\langle \text{boolean} - A (\forall_P p) = A (\forall_P p) \rangle$   
|  $\langle \text{boolean} - A (\forall_F p) = A (\forall_F p) \rangle$

**abbreviation**  $\langle \text{tautology } p \equiv \forall G A. \text{boolean } G A p \rangle$

**proposition**  $\langle \text{tautology } (\forall (\cdot P [\#0]) \longrightarrow \forall (\cdot P [\#0])) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *boolean-semantics*:  $\langle \text{boolean } (\lambda a. \langle E_P, G \rangle_2 a \circ \text{map } \langle E, E_F, C, F \rangle) (\lambda p. (E, E_F, E_P, C, F, G, PS, FS) \models p) = (\lambda p. (E, E_F, E_P, C, F, G, PS, FS) \models p) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *tautology[simp]*:  $\langle \text{tautology } p \implies (E, E_F, E_P, C, F, G, PS, FS) \models p \rangle$   
 $\langle \text{proof} \rangle$

**proposition**  $\langle \exists p. (\forall E E_F E_P C F G PS FS. (E, E_F, E_P, C, F, G, PS, FS) \models p) \wedge \neg \text{tautology } p \rangle$   
 $\langle \text{proof} \rangle$

### 3.6 Calculus

Adapted from System Q1 by Smullyan in First-Order Logic (1968).

**inductive** *Axiomatic* ( $\langle \vdash \rightarrow [50] 50 \rangle$ ) **where**

$TA: \langle \text{tautology } p \implies \vdash p \rangle$   
|  $IA: \langle \vdash \forall p \longrightarrow \langle t/0 \rangle p \rangle$   
|  $IA_P: \langle \vdash \forall_P p \longrightarrow \langle s/0 \rangle_P p \rangle$   
|  $IA_F: \langle \vdash \forall_F p \longrightarrow \langle s/0 \rangle_F p \rangle$   
|  $MP: \langle \vdash p \longrightarrow q \implies \vdash p \implies \vdash q \rangle$   
|  $GR: \langle \vdash q \longrightarrow \langle \star a/0 \rangle p \implies a \notin \text{params } \{p, q\} \implies \vdash q \longrightarrow \forall p \rangle$   
|  $GR_P: \langle \vdash q \longrightarrow \langle \circ_2 a/0 \rangle_P p \implies a \notin \text{params } \{p, q\} \implies \vdash q \longrightarrow \forall_P p \rangle$   
|  $GR_F: \langle \vdash q \longrightarrow \langle \circ_2 a/0 \rangle_F p \implies a \notin \text{params } \{p, q\} \implies \vdash q \longrightarrow \forall_F p \rangle$

We simulate assumptions on the lhs of  $\vdash$  with a chain of implications on the rhs.

**primrec** *imply* (**infixr**  $\langle \rightsquigarrow \rangle$  56) **where**

$\langle (\square \rightsquigarrow q) = q \rangle$   
|  $\langle (p \# ps \rightsquigarrow q) = (p \longrightarrow ps \rightsquigarrow q) \rangle$

**abbreviation** *Axiomatic-assms* ( $\langle \vdash \rightarrow [50, 50] 50 \rangle$ ) **where**

$\langle ps \vdash q \equiv \vdash ps \rightsquigarrow q \rangle$

### 3.7 Soundness

**fun** *wf-model* **where**

$\langle \text{wf-model } (E, E_F, E_P, C, F, G, PS, FS) \longleftrightarrow \text{range } G \subseteq PS \wedge \text{range } E_P \subseteq PS \wedge \text{range } F \subseteq FS \wedge \text{range } E_F \subseteq FS \rangle$

**theorem** *soundness*:

**shows**  $\langle \vdash p \implies \text{wf-model } (E, E_F, E_P, C, F, G, PS, FS) \implies (E, E_F, E_P, C, F, G, PS, FS) \models p \rangle$   
 $\langle \text{proof} \rangle$

**corollary**  $\langle \neg (\vdash \perp) \rangle$

$\langle \text{proof} \rangle$

### 3.8 Derived Rules

**lemma** *Imp1*:  $\langle \vdash q \longrightarrow p \longrightarrow q \rangle$

**and** *Imp2*:  $\langle \vdash (p \longrightarrow q \longrightarrow r) \longrightarrow (p \longrightarrow q) \longrightarrow p \longrightarrow r \rangle$

**and** *Neg*:  $\langle \vdash \neg \neg p \longrightarrow p \rangle$

$\langle \text{proof} \rangle$

The tautology axiom TA is not used directly beyond this point.

**lemma** *Tran'*:  $\langle \vdash (q \longrightarrow r) \longrightarrow (p \longrightarrow q) \longrightarrow p \longrightarrow r \rangle$

$\langle \text{proof} \rangle$

**lemma** *Swap*:  $\langle \vdash (p \longrightarrow q \longrightarrow r) \longrightarrow q \longrightarrow p \longrightarrow r \rangle$

$\langle \text{proof} \rangle$

**lemma** *Tran*:  $\langle \vdash (p \longrightarrow q) \longrightarrow (q \longrightarrow r) \longrightarrow p \longrightarrow r \rangle$

$\langle \text{proof} \rangle$

Note that contraposition in the other direction is an instance of the lemma Tran.

**lemma** *contraposition*:  $\langle \vdash (\neg q \longrightarrow \neg p) \longrightarrow p \longrightarrow q \rangle$

$\langle \text{proof} \rangle$

**lemma** *GR'*:  $\langle \vdash \neg \langle \star a / 0 \rangle p \longrightarrow q \implies a \notin \text{params } \{p, q\} \implies \vdash \neg (\forall p) \longrightarrow q \rangle$

$\langle \text{proof} \rangle$

**lemma** *GR<sub>P</sub>'*:  $\langle \vdash \neg \langle \odot_2 P / 0 \rangle_P p \longrightarrow q \implies P \notin \text{params } \{p, q\} \implies \vdash \neg (\forall_P p) \longrightarrow q \rangle$

$\langle \text{proof} \rangle$

**lemma** *GR<sub>F</sub>'*:  $\langle \vdash \neg \langle \odot_2 F / 0 \rangle_F p \longrightarrow q \implies F \notin \text{params } \{p, q\} \implies \vdash \neg (\forall_F p) \longrightarrow q \rangle$

$\langle \text{proof} \rangle$

**lemma** *imply-ImpE*:  $\langle \vdash ps \rightsquigarrow p \longrightarrow ps \rightsquigarrow (p \longrightarrow q) \longrightarrow ps \rightsquigarrow q \rangle$

$\langle \text{proof} \rangle$

**lemma** *MP'*:  $\langle ps \vdash p \longrightarrow q \implies ps \vdash p \implies ps \vdash q \rangle$

$\langle \text{proof} \rangle$

**lemma** *imply-Cons* [*intro*]:  $\langle ps \vdash q \implies p \# ps \vdash q \rangle$   
 $\langle proof \rangle$

**lemma** *imply-head* [*intro*]:  $\langle p \# ps \vdash p \rangle$   
 $\langle proof \rangle$

**lemma** *add-imply* [*simp*]:  $\langle \vdash q \implies ps \vdash q \rangle$   
 $\langle proof \rangle$

**lemma** *imply-mem* [*simp*]:  $\langle p \in set\ ps \implies ps \vdash p \rangle$   
 $\langle proof \rangle$

**lemma** *deduct1*:  $\langle ps \vdash p \longrightarrow q \implies p \# ps \vdash q \rangle$   
 $\langle proof \rangle$

**lemma** *imply-append* [*iff*]:  $\langle (ps @ qs \rightsquigarrow r) = (ps \rightsquigarrow qs \rightsquigarrow r) \rangle$   
 $\langle proof \rangle$

**lemma** *imply-swap-append*:  $\langle ps @ qs \vdash r \implies qs @ ps \vdash r \rangle$   
 $\langle proof \rangle$

**lemma** *deduct2*:  $\langle p \# ps \vdash q \implies ps \vdash p \longrightarrow q \rangle$   
 $\langle proof \rangle$

**lemmas** *deduct* [*iff*] = *deduct1* *deduct2*

**lemma** *cut*:  $\langle p \# ps \vdash r \implies q \# ps \vdash p \implies q \# ps \vdash r \rangle$   
 $\langle proof \rangle$

**lemma** *Boole*:  $\langle (\neg p) \# ps \vdash \perp \implies ps \vdash p \rangle$   
 $\langle proof \rangle$

**lemma** *imply-weaken*:  $\langle ps \vdash q \implies set\ ps \subseteq set\ ps' \implies ps' \vdash q \rangle$   
 $\langle proof \rangle$

### 3.9 Derivational Consistency

**interpretation** *DC*: *Weak-Derivational-Confl* map-fm params-fm  $\langle \lambda-. True \rangle$  confl-class  
 $\langle \lambda A. \neg A \vdash \perp \rangle$   
 $\langle proof \rangle$

**interpretation** *DA*: *Weak-Derivational-Alpha* map-fm params-fm  $\langle \lambda-. True \rangle$  al-  
pha-class  $\langle \lambda A. \neg A \vdash \perp \rangle$   
 $\langle proof \rangle$

**interpretation** *DB*: *Weak-Derivational-Beta* map-fm params-fm  $\langle \lambda-. True \rangle$  beta-class  
 $\langle \lambda A. \neg A \vdash \perp \rangle$   
 $\langle proof \rangle$

**interpretation**  $DG$ : *Weak-Derivational-Gamma map-tm map-fm params-fm*  $\langle \lambda-. True \rangle G.classify-UNIV \langle \lambda A. \neg A \vdash \perp \rangle$   
 $\langle proof \rangle$

**interpretation**  $DG_P$ : *Weak-Derivational-Gamma map-sym map-fm params-fm*  $\langle \lambda-. True \rangle G_P.classify-UNIV \langle \lambda A. \neg A \vdash \perp \rangle$   
 $\langle proof \rangle$

**interpretation**  $DG_F$ : *Weak-Derivational-Gamma map-sym map-fm params-fm*  $\langle \lambda-. True \rangle G_F.classify-UNIV \langle \lambda A. \neg A \vdash \perp \rangle$   
 $\langle proof \rangle$

**lemma** *imply-params-fm*:  $\langle params-fm (ps \rightsquigarrow q) = params-fm q \cup (\bigcup p \in set ps. params-fm p) \rangle$   
 $\langle proof \rangle$

**interpretation**  $DD$ : *Weak-Derivational-Delta map-fm params-fm*  $\langle \lambda-. True \rangle \delta \langle \lambda A. \neg A \vdash \perp \rangle$   
 $\langle proof \rangle$

**term**  $P.params$

**interpretation** *Weak-Derivational-Consistency map-fm params-fm*  $\langle \lambda-. True \rangle Kinds \langle \lambda A. \neg A \vdash \perp \rangle$   
 $\langle proof \rangle$

**theorem** *weak-completeness*:

**fixes**  $p :: \langle 'x fm \rangle$   
**assumes**  $mod: \langle \forall (E :: - \Rightarrow 'x tm) E_F E_P C F G PS FS. wf-model (E, E_F, E_P, C, F, G, PS, FS) \longrightarrow (\forall q \in set ps. (E, E_F, E_P, C, F, G, PS, FS) \models q) \longrightarrow (E, E_F, E_P, C, F, G, PS, FS) \models p \rangle$   
**and**  $\langle P.enough-new (set ps) \rangle$   
**shows**  $\langle ps \vdash p \rangle$   
 $\langle proof \rangle$

**theorem** *completeness*:

**fixes**  $p :: \langle 'x fm \rangle$   
**assumes**  $\langle \forall (E :: nat \Rightarrow 'x tm) E_P E_F C F G PS FS. wf-model (E, E_F, E_P, C, F, G, PS, FS) \longrightarrow (E, E_F, E_P, C, F, G, PS, FS) \models p \rangle$   
**and**  $\langle |UNIV :: 'x fm set| \leq o \quad |UNIV :: 'x set| \rangle$   
**shows**  $\langle \vdash p \rangle$   
 $\langle proof \rangle$

## 3.10 Natural Deduction

**locale** *Natural-Deduction*

**begin**

**inductive**  $ND-Set :: \langle 'x fm set \Rightarrow 'x fm \Rightarrow bool \rangle$  (**infix**  $\langle \Vdash \rangle$  50) **where**

$Assm [dest]: \langle p \in A \implies A \Vdash p \rangle$   
 $| FlsE [elim]: \langle A \Vdash \perp \implies A \Vdash p \rangle$   
 $| ImpI [intro]: \langle \{p\} \cup A \Vdash q \implies A \Vdash p \longrightarrow q \rangle$   
 $| ImpE [dest]: \langle A \Vdash p \longrightarrow q \implies A \Vdash p \implies A \Vdash q \rangle$   
 $| UniI [intro]: \langle A \Vdash \langle \star a / 0 \rangle p \implies a \notin P.params (\{p\} \cup A) \implies A \Vdash \forall p \rangle$   
 $| UniE [dest]: \langle A \Vdash \forall p \implies A \Vdash \langle t / 0 \rangle p \rangle$   
 $| UniI_P [intro]: \langle A \Vdash \langle \circ_2 a / 0 \rangle_P p \implies a \notin P.params (\{p\} \cup A) \implies A \Vdash \forall_P p \rangle$   
 $| UniE_P [dest]: \langle A \Vdash \forall_P p \implies A \Vdash \langle s / 0 \rangle_P p \rangle$   
 $| UniI_F [intro]: \langle A \Vdash \langle \circ_2 a / 0 \rangle_F p \implies a \notin P.params (\{p\} \cup A) \implies A \Vdash \forall_F p \rangle$   
 $| UniE_F [dest]: \langle A \Vdash \forall_F p \implies A \Vdash \langle s / 0 \rangle_F p \rangle$   
 $| Clas: \langle \{p \longrightarrow q\} \cup A \Vdash p \implies A \Vdash p \rangle$

### 3.10.1 Soundness

**theorem** *soundness-set:*

**assumes**  $\langle A \Vdash p \rangle \langle wf-model (E, E_F, E_P, C, F, G, PS, FS) \rangle$   
**shows**  $\langle \forall q \in A. (E, E_F, E_P, C, F, G, PS, FS) \models q \implies (E, E_F, E_P, C, F, G, PS, FS) \models p \rangle$   
 $\langle proof \rangle$

### 3.10.2 Derivational Consistency

**lemma** *Boole:*  $\langle \{\neg p\} \cup A \Vdash \perp \implies A \Vdash p \rangle$   
 $\langle proof \rangle$

**sublocale** *DC: Derivational-Confl map-fm params-fm  $\langle \lambda-. True \rangle$  confl-class  $\langle \lambda A. \neg A \Vdash \perp \rangle$*   
 $\langle proof \rangle$

**sublocale** *DA: Derivational-Alpha map-fm params-fm  $\langle \lambda-. True \rangle$  alpha-class  $\langle \lambda A. \neg A \Vdash \perp \rangle$*   
 $\langle proof \rangle$

**sublocale** *DB: Derivational-Beta map-fm params-fm  $\langle \lambda-. True \rangle$  beta-class  $\langle \lambda A. \neg A \Vdash \perp \rangle$*   
 $\langle proof \rangle$

**sublocale** *DG: Derivational-Gamma map-tm map-fm params-fm  $\langle \lambda-. True \rangle$  G.classify-UNIV  $\langle \lambda A. \neg A \Vdash \perp \rangle$*   
 $\langle proof \rangle$

**sublocale** *DGP: Derivational-Gamma map-sym map-fm params-fm  $\langle \lambda-. True \rangle$  G\_P.classify-UNIV  $\langle \lambda A. \neg A \Vdash \perp \rangle$*   
 $\langle proof \rangle$

**sublocale** *DGF: Derivational-Gamma map-sym map-fm params-fm  $\langle \lambda-. True \rangle$  G\_F.classify-UNIV  $\langle \lambda A. \neg A \Vdash \perp \rangle$*   
 $\langle proof \rangle$

**sublocale** *DD: Derivational-Delta map-fm params-fm  $\langle \lambda-. True \rangle$   $\delta \langle \lambda A. \neg A \Vdash \perp \rangle$*

*<proof>*

**sublocale** *Derivational-Consistency map-fm params-fm*  $\langle \lambda \cdot \text{True} \rangle$  *Kinds*  $\langle \lambda A. \neg A \Vdash \perp \rangle$   
*<proof>*

### 3.10.3 Strong Completeness

**theorem** *strong-completeness:*

**fixes**  $p :: \langle 'x \text{ fm} \rangle$   
**assumes** *mod:*  $\langle \bigwedge (E :: \text{nat} \Rightarrow 'x \text{ tm}) E_F E_P C F G PS FS. \text{wf-model } (E, E_F, E_P, C, F, G, PS, FS) \implies$   
 $(\forall q \in A. (E, E_F, E_P, C, F, G, PS, FS) \models q) \implies (E, E_F, E_P, C, F, G, PS, FS) \models p \rangle$   
**and**  $\langle P.\text{enough-new } A \rangle$   
**shows**  $\langle A \Vdash p \rangle$   
*<proof>*

**proposition**  $\langle \exists M. \text{wf-model } M \rangle$   
*<proof>*

**end**

**end**

## Chapter 4

# Example: Prior's Ideal Logic

```
theory Example-PIL imports
  Abstract-Consistency-Property
  HOL-Number-Theory.Number-Theory
begin
```

```
no-syntax
```

```
-Pi :: pttrn  $\Rightarrow$  'a set  $\Rightarrow$  'b set  $\Rightarrow$  ('a  $\Rightarrow$  'b) set
  ( $\langle$  ( $\langle$  indent=3 notation= $\langle$  binder  $\Pi \in \rangle \Pi$  - $\in$  ./ - $\rangle$  10)
```

### 4.1 Syntax

```
datatype (symbols-tm: 'x) tm
  = Var nat ( $\langle$  # $\rangle$ )
  | Sym 'x ( $\langle$   $\circ$  $\rangle$ )
```

```
datatype (symbols-fm: 'x) fm
  = TmI  $\langle$  'x tm $\rangle$  ( $\langle$   $\cdot$  $\rangle$ )
  | TmP  $\langle$  'x tm $\rangle$  ( $\langle$   $\cdot$  $\rangle$ )
  | Neg  $\langle$  'x fm $\rangle$  ( $\langle$   $\neg$  - $\rangle$  [70] 70)
  | Con  $\langle$  'x fm $\rangle$   $\langle$  'x fm $\rangle$  (infixr  $\langle$   $\wedge$  $\rangle$  35)
  | Box  $\langle$  'x fm $\rangle$  ( $\langle$   $\square$  $\rangle$ )
  | Sat  $\langle$  'x tm $\rangle$   $\langle$  'x fm $\rangle$  ( $\langle$   $\@$  $\rangle$ )
  | Glo  $\langle$  'x fm $\rangle$  ( $\langle$  A $\rangle$ )
  | Dwn  $\langle$  'x fm $\rangle$  ( $\langle$   $\downarrow$  $\rangle$ )
  | All  $\langle$  'x fm $\rangle$  ( $\langle$   $\forall$  $\rangle$ )
```

```
abbreviation Fls ::  $\langle$  'x fm $\rangle$  ( $\langle$   $\perp$  $\rangle$ ) where
   $\langle$   $\perp$   $\equiv$  undefined  $\wedge$   $\neg$  undefined $\rangle$ 
```

```
abbreviation Imp ::  $\langle$  'x fm  $\Rightarrow$  'x fm  $\Rightarrow$  'x fm $\rangle$  (infixr  $\langle$   $\longrightarrow$  $\rangle$  55) where
   $\langle$   $p \longrightarrow q \equiv \neg (p \wedge \neg q)$  $\rangle$ 
```

```
abbreviation Dia ::  $\langle$  'x fm  $\Rightarrow$  'x fm $\rangle$  ( $\langle$   $\diamond$  $\rangle$ ) where
   $\langle$   $\diamond p \equiv \neg (\square (\neg p))$  $\rangle$ 
```

**type-synonym**  $\langle 'x \text{ lbd} = \langle 'x \text{ tm} \times 'x \text{ fm} \rangle \rangle$

## 4.2 Semantics

**record**  $\langle 'w \text{ frame} =$   
 $\mathcal{W} :: \langle 'w \text{ set} \rangle$   
 $\mathcal{R} :: \langle 'w \Rightarrow 'w \text{ set} \rangle$

**record**  $\langle 'w \text{ gframe} = \langle 'w \text{ frame} \rangle +$   
 $\Pi :: \langle 'w \text{ set set} \rangle$

**record**  $\langle ('x, 'w) \text{ model} = \langle 'w \text{ gframe} \rangle +$   
 $\mathcal{N} :: \langle 'x \Rightarrow 'w \rangle$   
 $\mathfrak{N} :: \langle \text{nat} \Rightarrow 'w \rangle$   
 $\mathcal{V} :: \langle 'x \Rightarrow 'w \text{ set} \rangle$   
 $\mathfrak{V} :: \langle \text{nat} \Rightarrow 'w \text{ set} \rangle$

**abbreviation**  $\langle \text{Model } W R PI N e V f \equiv (\mathcal{W} = W, \mathcal{R} = R, \Pi = PI, \mathcal{N} = N, \mathfrak{N} = e, \mathcal{V} = V, \mathfrak{V} = f) \rangle$

**definition**  $\langle \text{admissible} :: \langle 'w \text{ frame} \Rightarrow 'w \text{ set set} \Rightarrow \text{bool} \rangle \text{ where}$   
 $\langle \text{admissible } M PI \equiv$   
 $(\forall X \in PI. \mathcal{W} M - X \in PI) \wedge$   
 $(\forall X \in PI. \forall Y \in PI. X \cap Y \in PI) \wedge$   
 $(\forall X \in PI. \{w \in \mathcal{W} M. \forall v \in \mathcal{R} M w. v \in X\} \in PI) \rangle$

**definition**  $\langle \text{wf-frame} :: \langle 'w \text{ frame} \Rightarrow \text{bool} \rangle \text{ where}$   
 $\langle \text{wf-frame } M \equiv \mathcal{W} M \neq \{\} \wedge \mathcal{R} M \subseteq \text{Pow } (\mathcal{W} M) \rangle$

**definition**  $\langle \text{wf-gframe} :: \langle 'w \text{ gframe} \Rightarrow \text{bool} \rangle \text{ where}$   
 $\langle \text{wf-gframe } M \equiv \text{wf-frame } (\text{frame.truncate } M) \wedge \Pi M \neq \{\} \wedge \Pi M \subseteq \text{Pow } (\mathcal{W} M) \wedge \text{admissible } (\text{frame.truncate } M) (\Pi M) \rangle$

**definition**  $\langle \text{wf-env} :: \langle ('x, 'w) \text{ model} \Rightarrow \text{bool} \rangle \text{ where}$   
 $\langle \text{wf-env } M \equiv \text{range } (\mathcal{N} M) \subseteq \mathcal{W} M \wedge \text{range } (\mathfrak{N} M) \subseteq \mathcal{W} M \wedge \text{range } (\mathcal{V} M) \subseteq \Pi M \wedge \text{range } (\mathfrak{V} M) \subseteq \Pi M \rangle$

**definition**  $\langle \text{wf-model} :: \langle ('x, 'w) \text{ model} \Rightarrow \text{bool} \rangle \text{ where}$   
 $\langle \text{wf-model } M \equiv \text{wf-gframe } (\text{gframe.truncate } M) \wedge \text{wf-env } M \rangle$

**lemmas**  $\text{unfolds} =$   
 $\text{model.defs gframe.defs frame.defs}$   
 $\text{model.select-convs gframe.select-convs frame.select-convs}$

**context**  
**fixes**  $M :: \langle ('x, 'w) \text{ model} \rangle$   
**assumes**  $\text{wf}: \langle \text{wf-model } M \rangle$   
**begin**

**lemma** *wf-compl*:  $\langle X \in \Pi M \implies \mathcal{W} M - X \in \Pi M \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *wf-inter*:  $\langle X \in \Pi M \implies Y \in \Pi M \implies X \cap Y \in \Pi M \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *wf-modal*:  $\langle X \in \Pi M \implies \{w \in \mathcal{W} M. \forall v \in \mathcal{R} M w. v \in X\} \in \Pi M \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *wf-empty*:  $\langle \{\} \in \Pi M \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *wf-univ*:  $\langle \mathcal{W} M \in \Pi M \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *wf- $\Pi$* :  $\langle P \in \Pi M \implies P \subseteq \mathcal{W} M \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *wf- $\mathcal{N}$* :  $\langle \mathcal{N} M i \in \mathcal{W} M \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *wf- $\mathfrak{N}$* :  $\langle \mathfrak{N} M n \in \mathcal{W} M \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *wf- $\mathcal{V}$* :  $\langle \mathcal{V} M P \in \Pi M \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *wf- $\mathcal{V}'$* :  $\langle \mathcal{V} M n \subseteq \mathcal{W} M \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *wf- $\mathfrak{V}$* :  $\langle \mathfrak{V} M n \in \Pi M \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *wf- $\mathfrak{V}'$* :  $\langle \mathfrak{V} M n \subseteq \mathcal{W} M \rangle$   
 $\langle \text{proof} \rangle$

**end**

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

**primrec** *add-env* ::  $\langle 'a \Rightarrow (\text{nat} \Rightarrow 'a) \Rightarrow \text{nat} \Rightarrow 'a \rangle$  (**infix**  $\langle \gg \rangle 0$ ) **where**  
 $\langle (t \gg e) 0 = t \rangle$   
 $| \langle (- \gg e) (\text{Suc } n) = e n \rangle$

**fun** *semantics* ::  $\langle ('x, 'w) \text{ ctx} \Rightarrow 'x \text{ fm} \Rightarrow \text{bool} \rangle$  (**infix**  $\langle \models \rangle 50$ ) **where**  
 $\langle ((M, w) \models \cdot t) = (\text{case-tm } (\mathfrak{N} M) (\mathcal{N} M) t = w) \rangle$   
 $| \langle ((M, w) \models \cdot P) = (w \in \text{case-tm } (\mathfrak{V} M) (\mathcal{V} M) P) \rangle$   
 $| \langle ((M, w) \models (\neg p)) = (\neg (M, w) \models p) \rangle$

$\langle (M, w) \models (p \wedge q) = (M, w) \models p \wedge (M, w) \models q \rangle$   
 $\langle (M, w) \models \Box p = (\forall v \in \mathcal{R} M w. (M, v) \models p) \rangle$   
 $\langle (M, -) \models @i p = ((M, \text{case-tm } (\mathfrak{N} M) (\mathcal{N} M) i) \models p) \rangle$   
 $\langle (M, -) \models \mathbf{A} p = (\forall v \in \mathcal{W} M. (M, v) \models p) \rangle$   
 $\langle (M, w) \models \Downarrow p = ((M(\mathfrak{N} := (w \gg \mathfrak{N} M)), w) \models p) \rangle$   
 $\langle (M, w) \models \forall p = (\forall P \in \Pi M. (M(\mathfrak{V} := (P \gg \mathfrak{V} M)), w) \models p) \rangle$

**lemma**  $\langle (M, w) \models p \longrightarrow q \longleftrightarrow (M, w) \models p \longrightarrow (M, w) \models q \rangle$   
 $\langle \text{proof} \rangle$

**lemma**  $\langle (M, w) \models \Downarrow(\#0) \rangle$   
 $\langle \text{proof} \rangle$

**lemma**  $\langle (M, w) \models \forall(@(\circ k) (\cdot(\#0))) \longrightarrow @(\circ k) (\forall(\cdot(\#0))) \rangle$   
 $\langle \text{proof} \rangle$

**lemma**  $\langle (M, w) \models @(\circ k) (\forall(\cdot(\#0))) \longrightarrow \forall(@(\circ k) (\cdot(\#0))) \rangle$   
 $\langle \text{proof} \rangle$

**lemma**  $\langle \text{wf-model } M \implies (M, w) \models \forall(\cdot(\#0)) \longrightarrow \cdot P \rangle$   
 $\langle \text{proof} \rangle$

**lemma**  $\langle \text{wf-model } M \implies (M, w) \models \forall(@(\circ k) (\cdot(\#0))) \longrightarrow \Downarrow(\circ k) \longrightarrow @(\#0) (\cdot P) \rangle$   
 $\langle \text{proof} \rangle$

### 4.3 Operations

**abbreviation**  $\text{map-lbd} :: \langle 'x \Rightarrow 'k \Rightarrow 'x \text{ lbd} \Rightarrow 'k \text{ lbd} \rangle$  **where**  
 $\langle \text{map-lbd } f \equiv \text{map-prod } (\text{map-tm } f) (\text{map-fm } f) \rangle$

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

**abbreviation**  $\text{symbols} :: \langle 'x \text{ lbd set} \Rightarrow 'x \text{ set} \rangle$  **where**  
 $\langle \text{symbols } S \equiv \bigcup ip \in S. \text{symbols-lbd } ip \rangle$

**abbreviation**  $\text{lift-tm} :: \langle \text{nat} \Rightarrow 'x \text{ tm} \Rightarrow 'x \text{ tm} \rangle$  **where**  
 $\langle \text{lift-tm } m \equiv \text{case-tm } (\lambda n. \text{if } n < m \text{ then } \#n \text{ else } \#(n + m + 1)) \circ \rangle$

**primrec**  $\text{vars-tm} :: \langle \text{nat} \Rightarrow 'x \text{ tm} \Rightarrow \text{nat set} \rangle$  **where**  
 $\langle \text{vars-tm } m (\#n) = (\text{if } n \leq m \text{ then } \{\} \text{ else } \{n\}) \rangle$   
 $\langle \text{vars-tm } - (\circ -) = \{\} \rangle$

**primrec**  $\text{vars-fm} :: \langle \text{nat} \Rightarrow 'x \text{ fm} \Rightarrow \text{nat set} \rangle$  **where**  
 $\langle \text{vars-fm } m (\cdot t) = \text{vars-tm } m t \rangle$   
 $\langle \text{vars-fm } m (\cdot P) = \text{vars-tm } m P \rangle$   
 $\langle \text{vars-fm } m (\neg p) = \text{vars-fm } m p \rangle$   
 $\langle \text{vars-fm } m (p \wedge q) = \text{vars-fm } m p \cup \text{vars-fm } m q \rangle$

$\langle \text{vars-fm } m (\Box p) = \text{vars-fm } m p \rangle$   
 $\langle \text{vars-fm } m (@i p) = \text{vars-tm } m i \cup \text{vars-fm } m p \rangle$   
 $\langle \text{vars-fm } m (\mathbf{A} p) = \text{vars-fm } m p \rangle$   
 $\langle \text{vars-fm } m (\Downarrow p) = \text{vars-fm } (m + 1) p \rangle$   
 $\langle \text{vars-fm } m (\forall p) = \text{vars-fm } (m + 1) p \rangle$

## Nominals

**primrec** *sub-tm* ::  $\langle \text{nat} \Rightarrow 'x \text{ tm} \Rightarrow 'x \text{ tm} \Rightarrow 'x \text{ tm} \rangle$  **where**  
 $\langle \text{sub-tm } s (\#n) = s n \rangle$   
 $\langle \text{sub-tm } - (\circ k) = \circ k \rangle$

**primrec** *sub-nom* ::  $\langle \text{nat} \Rightarrow 'x \text{ tm} \Rightarrow 'x \text{ fm} \Rightarrow 'x \text{ fm} \rangle$  **where**  
 $\langle \text{sub-nom } s (\cdot t) = \cdot(\text{sub-tm } s t) \rangle$   
 $\langle \text{sub-nom } - (\cdot P) = \cdot P \rangle$   
 $\langle \text{sub-nom } s (\neg p) = \neg \text{sub-nom } s p \rangle$   
 $\langle \text{sub-nom } s (p \wedge q) = (\text{sub-nom } s p \wedge \text{sub-nom } s q) \rangle$   
 $\langle \text{sub-nom } s (\Box p) = \Box(\text{sub-nom } s p) \rangle$   
 $\langle \text{sub-nom } s (@i p) = @(\text{sub-tm } s i) (\text{sub-nom } s p) \rangle$   
 $\langle \text{sub-nom } s (\mathbf{A} p) = \mathbf{A} (\text{sub-nom } s p) \rangle$   
 $\langle \text{sub-nom } s (\Downarrow p) = \Downarrow (\text{sub-nom } (\#0 \gg \lambda n. \text{lift-tm } 0 (s n)) p) \rangle$   
 $\langle \text{sub-nom } s (\forall p) = \forall (\text{sub-nom } s p) \rangle$

**abbreviation** *inst-single-nom* ::  $\langle 'x \text{ tm} \Rightarrow 'x \text{ fm} \Rightarrow 'x \text{ fm} \rangle (\langle \langle - \rangle_i \rangle)$  **where**  
 $\langle \langle t \rangle_i \equiv \text{sub-nom } (t \gg \#) \rangle$

## Propositions

**fun** *softqdf* ::  $\langle 'x \text{ fm} \Rightarrow \text{bool} \rangle$  **where**  
 $\langle \text{softqdf } (\cdot) = \text{False} \rangle$   
 $\langle \text{softqdf } (\cdot P) = \text{True} \rangle$   
 $\langle \text{softqdf } (\neg p) = \text{softqdf } p \rangle$   
 $\langle \text{softqdf } (p \wedge q) = (\text{softqdf } p \wedge \text{softqdf } q) \rangle$   
 $\langle \text{softqdf } (\Box p) = \text{softqdf } p \rangle$   
 $\langle \text{softqdf } (@i p) = \text{softqdf } p \rangle$   
 $\langle \text{softqdf } (\mathbf{A} p) = \text{softqdf } p \rangle$   
 $\langle \text{softqdf } (\Downarrow p) = \text{False} \rangle$   
 $\langle \text{softqdf } (\forall p) = \text{False} \rangle$

**abbreviation** *softqdf-sub* ::  $\langle \text{nat} \Rightarrow 'x \text{ fm} \Rightarrow \text{bool} \rangle$  **where**  
 $\langle \text{softqdf-sub } s \equiv \forall n. \text{softqdf } (s n) \rangle$

**primrec** *lift-fm-nom* ::  $\langle \text{nat} \Rightarrow 'x \text{ fm} \Rightarrow 'x \text{ fm} \rangle$  **where**  
 $\langle \text{lift-fm-nom } m (\cdot t) = \cdot(\text{lift-tm } m t) \rangle$   
 $\langle \text{lift-fm-nom } - (\cdot P) = \cdot P \rangle$   
 $\langle \text{lift-fm-nom } m (\neg p) = \neg \text{lift-fm-nom } m p \rangle$   
 $\langle \text{lift-fm-nom } m (p \wedge q) = (\text{lift-fm-nom } m p \wedge \text{lift-fm-nom } m q) \rangle$   
 $\langle \text{lift-fm-nom } m (\Box p) = \Box(\text{lift-fm-nom } m p) \rangle$   
 $\langle \text{lift-fm-nom } m (@i p) = @(\text{lift-tm } m i) (\text{lift-fm-nom } m p) \rangle$   
 $\langle \text{lift-fm-nom } m (\mathbf{A} p) = \mathbf{A} (\text{lift-fm-nom } m p) \rangle$

|  $\langle \text{lift-fm-nom } m (\downarrow p) = \downarrow (\text{lift-fm-nom } (m + 1) p) \rangle$   
|  $\langle \text{lift-fm-nom } m (\forall p) = \forall (\text{lift-fm-nom } m p) \rangle$

**primrec** *lift-fm-pro* ::  $\langle \text{nat} \Rightarrow 'x \text{ fm} \Rightarrow 'x \text{ fm} \rangle$  **where**

$\langle \text{lift-fm-pro } - (\cdot t) = \cdot t \rangle$   
|  $\langle \text{lift-fm-pro } m (\cdot P) = \cdot (\text{lift-tm } m P) \rangle$   
|  $\langle \text{lift-fm-pro } m (\neg p) = \neg \text{lift-fm-pro } m p \rangle$   
|  $\langle \text{lift-fm-pro } m (p \wedge q) = (\text{lift-fm-pro } m p \wedge \text{lift-fm-pro } m q) \rangle$   
|  $\langle \text{lift-fm-pro } m (\Box p) = \Box (\text{lift-fm-pro } m p) \rangle$   
|  $\langle \text{lift-fm-pro } m (@i p) = @i (\text{lift-fm-pro } m p) \rangle$   
|  $\langle \text{lift-fm-pro } m (\mathbf{A} p) = \mathbf{A} (\text{lift-fm-pro } m p) \rangle$   
|  $\langle \text{lift-fm-pro } m (\downarrow p) = \downarrow (\text{lift-fm-pro } m p) \rangle$   
|  $\langle \text{lift-fm-pro } m (\forall p) = \forall (\text{lift-fm-pro } (m + 1) p) \rangle$

**primrec** *sub-pro* ::  $\langle (\text{nat} \Rightarrow 'x \text{ fm}) \Rightarrow 'x \text{ fm} \Rightarrow 'x \text{ fm} \rangle$  **where**

$\langle \text{sub-pro } - (\cdot t) = \cdot t \rangle$   
|  $\langle \text{sub-pro } s (\cdot P) = \text{case-tm } s (\cdot o \circ) P \rangle$   
|  $\langle \text{sub-pro } s (\neg p) = \neg \text{sub-pro } s p \rangle$   
|  $\langle \text{sub-pro } s (p \wedge q) = (\text{sub-pro } s p \wedge \text{sub-pro } s q) \rangle$   
|  $\langle \text{sub-pro } s (\Box p) = \Box (\text{sub-pro } s p) \rangle$   
|  $\langle \text{sub-pro } s (@i p) = @i (\text{sub-pro } s p) \rangle$   
|  $\langle \text{sub-pro } s (\mathbf{A} p) = \mathbf{A} (\text{sub-pro } s p) \rangle$   
|  $\langle \text{sub-pro } s (\downarrow p) = \downarrow (\text{sub-pro } (\text{lift-fm-nom } 0 o s) p) \rangle$   
|  $\langle \text{sub-pro } s (\forall p) = \forall (\text{sub-pro } (\cdot (\#0) \gg \lambda n. \text{lift-fm-pro } 0 (s n)) p) \rangle$

**abbreviation** *inst-single-pro* ::  $\langle 'x \text{ fm} \Rightarrow 'x \text{ fm} \Rightarrow 'x \text{ fm} \rangle (\langle \cdot \rangle_p)$  **where**

$\langle \langle q \rangle_p \equiv \text{sub-pro } (q \gg \cdot o \#) \rangle$

**primrec** *sz-fm* ::  $\langle 'x \text{ fm} \Rightarrow \text{nat} \rangle$  **where**

$\langle \text{sz-fm } (\cdot t) = 1 \rangle$   
|  $\langle \text{sz-fm } (\cdot P) = 1 \rangle$   
|  $\langle \text{sz-fm } (\neg p) = \text{sz-fm } p + 1 \rangle$   
|  $\langle \text{sz-fm } (p \wedge q) = \text{sz-fm } p + \text{sz-fm } q + 1 \rangle$   
|  $\langle \text{sz-fm } (\Box p) = \text{sz-fm } p + 1 \rangle$   
|  $\langle \text{sz-fm } (@i p) = \text{sz-fm } p + 1 \rangle$   
|  $\langle \text{sz-fm } (\mathbf{A} p) = \text{sz-fm } p + 1 \rangle$   
|  $\langle \text{sz-fm } (\downarrow p) = \text{sz-fm } p + 1 \rangle$   
|  $\langle \text{sz-fm } (\forall p) = \text{sz-fm } p + 1 \rangle$

**primrec** *qs-fm* ::  $\langle 'x \text{ fm} \Rightarrow \text{nat} \rangle$  **where**

$\langle \text{qs-fm } (\cdot t) = 0 \rangle$   
|  $\langle \text{qs-fm } (\cdot P) = 0 \rangle$   
|  $\langle \text{qs-fm } (\neg p) = \text{qs-fm } p \rangle$   
|  $\langle \text{qs-fm } (p \wedge q) = \max (\text{qs-fm } p) (\text{qs-fm } q) \rangle$   
|  $\langle \text{qs-fm } (\Box p) = \text{qs-fm } p \rangle$   
|  $\langle \text{qs-fm } (@i p) = \text{qs-fm } p \rangle$   
|  $\langle \text{qs-fm } (\mathbf{A} p) = \text{qs-fm } p \rangle$   
|  $\langle \text{qs-fm } (\downarrow p) = \text{qs-fm } p \rangle$   
|  $\langle \text{qs-fm } (\forall p) = \text{qs-fm } p + 1 \rangle$

### 4.3.1 Lemmas

#### Finite

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

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

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

#### Terms

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

**lemma** *lift-lemma* [*simp*]:  $\langle \text{case-tm } (x \gg e) s (\text{lift-tm } 0 t) = \text{case-tm } e s t \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *sub-tm-semantic* [*simp*]:  $\langle \text{case-tm } e g (\text{sub-tm } s t) = \text{case-tm } (\lambda n. \text{case-tm } e g (s n)) g t \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *semantics-tm-id* [*simp*]:  $\langle \text{case-tm } \# \circ t = t \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *semantics-tm-id-map* [*simp*]:  $\langle \text{map} (\text{case-tm } \# \circ) ts = ts \rangle$   
 $\langle \text{proof} \rangle$

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

**lemma** *map-tm-lift-tm* [*simp*]:  $\langle \text{map-tm } f (\text{lift-tm } m t) = \text{lift-tm } m (\text{map-tm } f t) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *semantics-tm-fresh* [*simp*]:  $\langle x \notin \text{symbols-tm } t \implies \text{case-tm } e (g(x := a)) t = \text{case-tm } e g t \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *map-tm-inst-single*:  $\langle (\text{map-tm } f o (u \gg \#)) t = (\text{map-tm } f u \gg \#) t \rangle$   
 $\langle \text{proof} \rangle$

#### Nominals

**lemma** *size-sub-nom* [*simp*]:  $\langle \text{sz-fm} (\text{sub-nom } s p) = \text{sz-fm } p \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *semantics-symbols-cong-nom*:

**assumes**  $\langle \forall i \in \text{symbols-fm } p. N i = N' i \rangle$

**shows**  $\langle (\text{Model } W R P I N e V f, w) \models p \longleftrightarrow (\text{Model } W R P I N' e V f, w) \models p \rangle$   
 $\langle \text{proof} \rangle$

**corollary** *semantics-symbols-other-nom* [simp]:

**assumes**  $\langle i \notin \text{symbols-fm } p \rangle$

**shows**  $\langle (\text{Model } W R P I (N(i := x)) e V f, w) \models p \longleftrightarrow (\text{Model } W R P I N e V f, w) \models p \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *sub-nom-semantics* [simp]:

$\langle (\text{Model } W R P I N e V f, w) \models \text{sub-nom } s p \longleftrightarrow (\text{Model } W R P I N (\lambda n. \text{case-tm } e N (s n)) V f, w) \models p \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *map-fm-sub-nom*:  $\langle \text{map-fm } f (\text{sub-nom } s p) = \text{sub-nom } (\text{map-tm } f o s) (\text{map-fm } f p) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *map-fm-inst-single-nom* [simp]:  $\langle \text{map-fm } f (\langle t \rangle_i p) = \langle \text{map-tm } f t \rangle_i (\text{map-fm } f p) \rangle$   
 $\langle \text{proof} \rangle$

## Propositions

**lemma** *semantics-symbols-cong-pro*:

$\langle \forall P \in \text{symbols-fm } p. V P = V' P \implies$

$(\text{Model } W R P I N e V f, w) \models p \longleftrightarrow (\text{Model } W R P I N e V' f, w) \models p \rangle$   
 $\langle \text{proof} \rangle$

**corollary** *semantics-symbols-other-pro* [simp]:

**assumes**  $\langle P \notin \text{symbols-fm } p \rangle$

**shows**  $\langle (\text{Model } W R P I N e (V(P := x)) f, w) \models p \longleftrightarrow (\text{Model } W R P I N e V f, w) \models p \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *semantics-symbols-lbd-cong-pro*:

$\langle \forall P \in \text{symbols-lbd } (i, p). V P = V' P \implies$

$(\text{Model } W R P I N e V f, w) \models p \longleftrightarrow (\text{Model } W R P I N e V' f, w) \models p \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *map-lift-fm-pro* [simp]:  $\langle \text{map-fm } f (\text{lift-fm-pro } m p) = \text{lift-fm-pro } m (\text{map-fm } f p) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *map-lift-fm-nom* [simp]:  $\langle \text{map-fm } f (\text{lift-fm-nom } m p) = \text{lift-fm-nom } m (\text{map-fm } f p) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *map-fm-sub-pro*:  $\langle \text{map-fm } f \text{ (sub-pro } s \text{ } p) = \text{sub-pro (map-fm } f \text{ } o \text{ } s) \text{ (map-fm } f \text{ } p) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *map-fm-inst-single*:  $\langle (\text{map-fm } f \text{ } o \text{ } (q \gg \cdot o \#)) \text{ } p = (\text{map-fm } f \text{ } q \gg \cdot o \#) \text{ } p \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *map-fm-inst-single-pro* [*simp*]:  $\langle \text{map-fm } f \text{ } (\langle q \rangle_p \text{ } p) = \langle \text{map-fm } f \text{ } q \rangle_p \text{ (map-fm } f \text{ } p) \rangle$   
 $\langle \text{proof} \rangle$

### 4.3.2 softqdf

**lemma** *softqdf-map-fm* [*simp*]:  $\langle \text{softqdf (map-fm } f \text{ } p) \longleftrightarrow \text{softqdf } p \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *softqdf-lift-fm-nom* [*simp*]:  $\langle \text{softqdf (lift-fm-nom } m \text{ } q) \longleftrightarrow \text{softqdf } q \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *softqdf-lift-fm-pro* [*simp*]:  $\langle \text{softqdf (lift-fm-pro } m \text{ } q) \longleftrightarrow \text{softqdf } q \rangle$   
 $\langle \text{proof} \rangle$

### 4.3.3 Add env

**lemma** *range-add-env*:  
**assumes**  $\langle \text{range } f \subseteq A \rangle \langle a \in A \rangle$   
**shows**  $\langle \text{range } (a \gg f) \subseteq A \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *softqdf-add-env*:  $\langle \text{softqdf } q \implies \text{softqdf-sub } (q \gg \cdot o \#) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *wf-env-add-nom*:  $\langle \text{wf-env (Model } W \text{ } R \text{ } PI \text{ } N \text{ } e \text{ } V \text{ } f) \implies w \in W \implies \text{wf-env } (\mathcal{W} = W, \mathcal{R} = R, \Pi = PI, \mathcal{N} = N, \mathfrak{N} = w \gg e, \mathcal{V} = V, \mathfrak{V} = f) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *wf-model-add-nom*:  $\langle \text{wf-model (Model } W \text{ } R \text{ } PI \text{ } N \text{ } e \text{ } V \text{ } f) \implies w \in W \implies \text{wf-model } (\mathcal{W} = W, \mathcal{R} = R, \Pi = PI, \mathcal{N} = N, \mathfrak{N} = w \gg e, \mathcal{V} = V, \mathfrak{V} = f) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *wf-env-add-pro*:  $\langle \text{wf-env (Model } W \text{ } R \text{ } PI \text{ } N \text{ } e \text{ } V \text{ } f) \implies P \in PI \implies \text{wf-env } (\mathcal{W} = W, \mathcal{R} = R, \Pi = PI, \mathcal{N} = N, \mathfrak{N} = e, \mathcal{V} = V, \mathfrak{V} = P \gg f) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *wf-model-add-pro*:  
 $\langle \text{wf-model (Model } W \text{ } R \text{ } PI \text{ } N \text{ } e \text{ } V \text{ } f) \implies P \in PI \implies \text{wf-model (Model } W \text{ } R \text{ } PI \text{ } N \text{ } e \text{ } V \text{ } (P \gg f)) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *softqdf-lift-fm-nom-add-env* [*simp*]:  
 $\langle \text{softqdf } p \implies (\text{Model } W R P I N (v \gg e) V f, w) \models \text{lift-fm-nom } 0 p \iff (\text{Model } W R P I N e V f, w) \models p \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *softqdf-lift-fm-pro-add-env* [*simp*]:  
 $\langle \text{softqdf } p \implies (\text{Model } W R P I N e V (P \gg f), w) \models \text{lift-fm-pro } 0 p \iff (\text{Model } W R P I N e V f, w) \models p \rangle$   
 $\langle \text{proof} \rangle$

#### 4.3.4 Sizes

**lemma** *qs-fm-sub-nom* [*simp*]:  $\langle \text{qs-fm } (\text{sub-nom } s p) = \text{qs-fm } p \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *softqdf-qs-fm* [*simp*]:  $\langle \text{softqdf } q \implies \text{qs-fm } q = 0 \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *softqdf-sub-pro*:  $\langle \text{softqdf-sub } s \implies \text{softqdf-sub } (\cdot(\#0) \gg \lambda n. \text{lift-fm-pro } 0 (s n)) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *qs-fm-sub-pro* [*simp*]:  $\langle \text{softqdf-sub } s \implies \text{qs-fm } (\text{sub-pro } s p) = \text{qs-fm } p \rangle$   
 $\langle \text{proof} \rangle$

## 4.4 Propositional Quantification

**definition** *worlds* ::  $\langle ('x, 'w) \text{ model} \Rightarrow 'x \text{ fm} \Rightarrow 'w \text{ set} \rangle$  **where**  
 $\langle \text{worlds } M p \equiv \{ w \in \mathcal{W} M. (M, w) \models p \} \rangle$

**lemma** *worlds-op* [*simp*]:  
**assumes**  $\langle \text{wf-model } M \rangle$   
**shows**  
 $\langle \text{worlds } M (\neg p) = \mathcal{W} M - \text{worlds } M p \rangle$   
 $\langle \text{worlds } M (p \wedge q) = \text{worlds } M p \cap \text{worlds } M q \rangle$   
 $\langle \text{worlds } M (\Box p) = \{ w \in \mathcal{W} M. \forall v \in \mathcal{R} M w. v \in \text{worlds } M p \} \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *softqdf-worlds*:  
**assumes**  $\langle \text{wf-model } M \rangle \langle \text{softqdf } p \rangle$   
**shows**  $\langle \text{worlds } M p \in \Pi M \rangle$   
 $\langle \text{proof} \rangle$

**definition** *sqdfs* ::  $\langle ('x, 'w) \text{ model} \Rightarrow 'w \text{ set set} \rangle$  **where**  
 $\langle \text{sqdfs } M \equiv \{ \text{worlds } M p \mid p. \text{softqdf } p \} \rangle$

**lemma** *sqdfs*:  
**assumes**  $\langle \text{wf-model } M \rangle$

**shows**  $\langle sqdfs\ M \subseteq \Pi\ M \rangle$   
 $\langle proof \rangle$

**lemma** *sqdfs-admissible*:

**assumes**  $\langle wf\text{-}model\ M \rangle$   
**shows**  $\langle admissible\ (frame.truncate\ M)\ (sqdfs\ M) \rangle$   
 $\langle proof \rangle$

**definition** *with-worlds* ::  $\langle ('x, 'w)\ model \Rightarrow (nat \Rightarrow 'x\ fm) \Rightarrow ('x, 'w)\ model \rangle$   
**where**

$\langle with\text{-}worlds\ M\ s \equiv M(\mathfrak{W} := worlds\ M\ o\ s) \rangle$

**lemma** *sub-pro-with-worlds*:

**assumes**  $\langle wf\text{-}model\ (Model\ W\ R\ PI\ N\ e\ V\ f) \rangle$   $\langle w \in W \rangle$   $\langle softqdf\text{-}sub\ s \rangle$   
**shows**  $\langle (Model\ W\ R\ PI\ N\ e\ V\ f, w) \models sub\text{-}pro\ s\ p \iff (with\text{-}worlds\ (Model\ W\ R\ PI\ N\ e\ V\ f)\ s, w) \models p \rangle$   
 $\langle proof \rangle$

**lemma** *worlds-id-sub*:

**assumes**  $\langle wf\text{-}model\ (Model\ W\ R\ PI\ N\ e\ V\ f) \rangle$   
**shows**  $\langle worlds\ (Model\ W\ R\ PI\ N\ e\ V\ f)\ (\cdot\ (\#n)) = f\ n \rangle$   
 $\langle proof \rangle$

**lemma** *worlds-inst-single-pro*:

**assumes**  $\langle wf\text{-}model\ (Model\ W\ R\ PI\ N\ e\ V\ f) \rangle$   
**shows**  $\langle worlds\ (Model\ W\ R\ PI\ N\ e\ V\ f)\ o\ (q \gg \cdot\ o\ \#) = (worlds\ (Model\ W\ R\ PI\ N\ e\ V\ f)\ q \gg f) \rangle$   
 $\langle proof \rangle$

**corollary** *inst-single-worlds*:

**assumes**  $\langle wf\text{-}model\ (Model\ W\ R\ PI\ N\ e\ V\ f) \rangle$   $\langle w \in W \rangle$   $\langle softqdf\ q \rangle$   
**shows**  
 $\langle (Model\ W\ R\ PI\ N\ e\ V\ f, w) \models \langle q \rangle_p\ p \iff$   
 $(Model\ W\ R\ PI\ N\ e\ V\ (worlds\ (Model\ W\ R\ PI\ N\ e\ V\ f)\ q \gg f), w) \models p \rangle$   
 $\langle proof \rangle$

## 4.5 Model Existence

**inductive** *confl-class* ::  $\langle 'x\ lbd\ list \Rightarrow 'x\ lbd\ list \Rightarrow bool \rangle$  (**infix**  $\langle \rightsquigarrow_{\mathbf{x}} \rangle$  50) **where**

*CNegP*:  $\langle [(i, \neg \cdot P)] \rightsquigarrow_{\mathbf{x}} [(i, \cdot P)] \rangle$   
| *CNegI*:  $\langle [(i, \neg \cdot k)] \rightsquigarrow_{\mathbf{x}} [(i, \cdot k)] \rangle$

**inductive** *alpha-class* ::  $\langle 'x\ lbd\ list \Rightarrow 'x\ lbd\ list \Rightarrow bool \rangle$  (**infix**  $\langle \rightsquigarrow_{\alpha} \rangle$  50) **where**

*CSym*:  $\langle [(i, \cdot k)] \rightsquigarrow_{\alpha} [(k, \cdot i)] \rangle$   
| *CNom*:  $\langle [(i, \cdot k), (i, p)] \rightsquigarrow_{\alpha} [(k, p)] \rangle$   
| *CNegN*:  $\langle [(i, \neg \neg p)] \rightsquigarrow_{\alpha} [(i, p)] \rangle$   
| *CConP*:  $\langle [(i, p \wedge q)] \rightsquigarrow_{\alpha} [(i, p), (i, q)] \rangle$   
| *CSatP*:  $\langle [(i, @k\ p)] \rightsquigarrow_{\alpha} [(k, p)] \rangle$   
| *CSatN*:  $\langle [(i, \neg @k\ p)] \rightsquigarrow_{\alpha} [(k, \neg p)] \rangle$

| *CBoxP*:  $\langle [ (i, \Box p), (i, \Diamond(\cdot k)) ] \rightsquigarrow_{\alpha} [(k, p)] \rangle$   
| *CDwnP*:  $\langle [ (i, \Downarrow p) ] \rightsquigarrow_{\alpha} [ (i, \langle i \rangle_i p) ] \rangle$   
| *CDwnN*:  $\langle [ (i, \neg \Downarrow p) ] \rightsquigarrow_{\alpha} [ (i, \neg \langle i \rangle_i p) ] \rangle$

**inductive** *beta-class* ::  $\langle 'x \text{ lbd list} \Rightarrow 'x \text{ lbd list} \Rightarrow \text{bool} \rangle$  (**infix**  $\langle \rightsquigarrow_{\beta} \rangle$  50) **where**  
*CConN*:  $\langle [ (i, \neg (p \wedge q)) ] \rightsquigarrow_{\beta} [(i, \neg p), (i, \neg q)] \rangle$

**inductive** *gamma-class-nom* ::  $\langle 'x \text{ lbd list} \Rightarrow ('x \text{ tm} \Rightarrow -) \Rightarrow \text{bool} \rangle$  (**infix**  $\langle \rightsquigarrow_{\gamma i} \rangle$  50) **where**

*CRefI*:  $\langle [] \rightsquigarrow_{\gamma i} (\lambda i. [ (i, \cdot i) ]) \rangle$   
| *CGloP*:  $\langle [ (i, \mathbf{A} p) ] \rightsquigarrow_{\gamma i} (\lambda k. [ (k, p) ]) \rangle$

**inductive** *gamma-class-fm* ::  $\langle 'x \text{ lbd list} \Rightarrow ('x \text{ lbd set} \Rightarrow 'x \text{ fm set}) \times ('x \text{ fm} \Rightarrow -) \Rightarrow \text{bool} \rangle$  (**infix**  $\langle \rightsquigarrow_{\gamma p} \rangle$  50) **where**

*CALLP*:  $\langle [ (i, \forall p) ] \rightsquigarrow_{\gamma p} (\lambda -. \{q. \text{softqdf } q\}, \lambda q. [ (i, \langle q \rangle_p p) ]) \rangle$

**fun**  $\delta$  ::  $\langle 'x \text{ lbd} \Rightarrow 'x \Rightarrow 'x \text{ lbd list} \rangle$  **where**

*CBoxN*:  $\langle \delta (i, \neg \Box p) k = [(\Box k, \neg p), (i, \Diamond(\cdot (\Box k)))] \rangle$   
| *CGloN*:  $\langle \delta (i, \neg \mathbf{A} p) k = [(\Box k, \neg p)] \rangle$   
| *CALLN*:  $\langle \delta (i, \neg \forall p) P = [ (i, \neg \langle \cdot (\Box P) \rangle_p p) ] \rangle$   
|  $\langle \delta \text{ --} = [] \rangle$

**interpretation** *P*: *Params map-lbd symbols-lbd*  $\langle \lambda -. \text{True} \rangle$   
 $\langle \text{proof} \rangle$

**interpretation** *C*: *Confl map-lbd symbols-lbd*  $\langle \lambda -. \text{True} \rangle$  *confl-class*  
 $\langle \text{proof} \rangle$

**interpretation** *A*: *Alpha map-lbd symbols-lbd*  $\langle \lambda -. \text{True} \rangle$  *alpha-class*  
 $\langle \text{proof} \rangle$

**interpretation** *B*: *Beta map-lbd symbols-lbd*  $\langle \lambda -. \text{True} \rangle$  *beta-class*  
 $\langle \text{proof} \rangle$

**interpretation** *GI*: *Gamma-UNIV map-tm map-lbd symbols-lbd*  $\langle \lambda -. \text{True} \rangle$  *gamma-class-nom*  
 $\langle \text{proof} \rangle$

**interpretation** *GP*: *Gamma map-fm map-lbd symbols-lbd*  $\langle \lambda -. \text{True} \rangle$  *gamma-class-fm*  
 $\langle \text{proof} \rangle$

**interpretation** *D*: *Delta map-lbd symbols-lbd*  $\langle \lambda -. \text{True} \rangle$   $\delta$   
 $\langle \text{proof} \rangle$

**abbreviation** *Kinds* ::  $\langle ('x, 'x \text{ lbd}) \text{ kind list} \rangle$  **where**

$\langle \text{Kinds} \equiv [C.\text{kind}, A.\text{kind}, B.\text{kind}, GI.\text{kind}, GP.\text{kind}, D.\text{kind}] \rangle$

**lemma** *prop<sub>E</sub>-Kinds*:

**assumes**  $\langle P.\text{sat}_E C.\text{kind } C \rangle \langle P.\text{sat}_E A.\text{kind } C \rangle \langle P.\text{sat}_E B.\text{kind } C \rangle \langle P.\text{sat}_E GI.\text{kind } C \rangle \langle P.\text{sat}_E GP.\text{kind } C \rangle \langle P.\text{sat}_E D.\text{kind } C \rangle$

**shows**  $\langle P.prop_E \text{ Kinds } C \rangle$   
 $\langle proof \rangle$

**interpretation** *Consistency-Kinds map-lbd symbols-lbd*  $\langle \lambda-. \text{ True} \rangle \text{ Kinds}$   
 $\langle proof \rangle$

**interpretation** *Maximal-Consistency map-lbd symbols-lbd*  $\langle \lambda-. \text{ True} \rangle \text{ Kinds}$   
 $\langle proof \rangle$

**context begin**

**lemma**  $\langle P.prop_E \text{ Kinds } C \implies S \in C \implies (i, \cdot P) \notin S \vee (i, \neg \cdot P) \notin S \rangle$   
 $\langle proof \rangle$

**lemma**  $\langle P.prop_E \text{ Kinds } C \implies S \in C \implies (i, p \wedge q) \in S \implies \{(i, p), (i, q)\} \cup S \in C \rangle$   
 $\langle proof \rangle$

**lemma**  $\langle P.prop_E \text{ Kinds } C \implies S \in C \implies (i, \neg (p \wedge q)) \in S \implies \{(i, \neg p)\} \cup S \in C \vee \{(i, \neg q)\} \cup S \in C \rangle$   
 $\langle proof \rangle$

**lemma**  $\langle P.prop_E \text{ Kinds } C \implies S \in C \implies (i, \Box p) \in S \implies (i, \Diamond(\cdot k)) \in S \implies \{(k, p)\} \cup S \in C \rangle$   
 $\langle proof \rangle$

**lemma**  $\langle P.prop_E \text{ Kinds } C \implies S \in C \implies (i, \neg \Box p) \in S \implies \exists k. \{(k, \neg p), (i, \Diamond(\cdot k))\} \cup S \in C \rangle$   
 $\langle proof \rangle$

**lemma**  $\langle P.prop_E \text{ Kinds } C \implies S \in C \implies \{(i, \cdot i)\} \cup S \in C \rangle$   
 $\langle proof \rangle$

**lemma**  $\langle P.prop_E \text{ Kinds } C \implies S \in C \implies (i, \mathbf{A} p) \in S \implies \{(k, p)\} \cup S \in C \rangle$   
 $\langle proof \rangle$

**lemma**  $\langle P.prop_E \text{ Kinds } C \implies S \in C \implies (i, \neg \mathbf{A} p) \in S \implies \exists k. \{(k, \neg p)\} \cup S \in C \rangle$   
 $\langle proof \rangle$

**lemma**  $\langle P.prop_E \text{ Kinds } C \implies S \in C \implies (i, \forall p) \in S \implies softqdf q \implies \{(i, \langle q \rangle_p p)\} \cup S \in C \rangle$   
 $\langle proof \rangle$

**lemma**  $\langle P.prop_E \text{ Kinds } C \implies S \in C \implies (i, \neg \forall p) \in S \implies \exists P. \{(i, \neg \langle \cdot(\circ P) \rangle_p p)\} \cup S \in C \rangle$   
 $\langle proof \rangle$

**end**

**definition** *equiv-nom* ::  $\langle 'x \text{ lbd set} \Rightarrow 'x \text{ tm} \Rightarrow 'x \text{ tm} \Rightarrow \text{bool} \rangle$  **where**  
 $\langle \text{equiv-nom } S \ i \ k \equiv (i, \cdot k) \in S \rangle$

**definition** *assign* ::  $\langle 'x \text{ tm} \Rightarrow 'x \text{ lbd set} \Rightarrow 'x \text{ tm} \rangle$  ( $\langle [-] \cdot \rangle [0, 100] 100$ ) **where**  
 $\langle [i]_S \equiv \text{wo-rel.minim} (|UNIV|) \{k. \text{equiv-nom } S \ i \ k\} \rangle$

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

**definition** *val* ::  $\langle 'x \text{ lbd set} \Rightarrow 'x \text{ tm} \Rightarrow 'x \text{ tm set} \rangle$  **where**  
 $\langle \text{val } S \ P \equiv \{[i]_S \mid i. (i, \cdot P) \in S\} \rangle$

**lemma** *range-val-ne*:  $\langle \text{range} (\text{val } S) \neq \{\} \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *admissible-Pow*:  $\langle \text{admissible } F \ (\text{Pow } (\mathcal{W} \ F)) \rangle$   
 $\langle \text{proof} \rangle$

**definition** *admits* ::  $\langle 'w \text{ frame} \Rightarrow 'w \text{ set set} \Rightarrow 'w \text{ set set} \Rightarrow \text{bool} \rangle$  **where**  
 $\langle \text{admits } F \ B \ PI \equiv PI \neq \{\} \wedge PI \subseteq \text{Pow } (\mathcal{W} \ F) \wedge B \subseteq PI \wedge \text{admissible } F \ PI \rangle$

**definition** *adm- $\delta$*  ::  $\langle 'w \text{ frame} \Rightarrow 'w \text{ set set} \Rightarrow 'w \text{ set set} \rangle$  **where**  
 $\langle \text{adm-}\delta \ M \ PI \equiv$   
 $\{ \mathcal{W} \ M - X \mid X. X \in PI \} \cup$   
 $\{ X \cap Y \mid X \ Y. X \in PI \wedge Y \in PI \} \cup$   
 $\{ \{w \in \mathcal{W} \ M. \forall v \in \mathcal{R} \ M \ w. v \in X\} \mid X. X \in PI \} \rangle$

**abbreviation** *grow*  $F \ B \equiv \lambda PI. B \cup PI \cup \text{adm-}\delta \ F \ (B \cup PI)$

**definition** *admit* ::  $\langle 'w \text{ frame} \Rightarrow 'w \text{ set set} \Rightarrow 'w \text{ set set} \rangle$  **where**  
 $\langle \text{admit } F \ B \equiv \text{lfp} (\text{grow } F \ B) \rangle$

**lemma** *mono-grow*:  $\langle \text{mono} (\text{grow } F \ B) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *admissible- $\delta$* :  $\langle \text{admissible } F \ B \longleftrightarrow \text{adm-}\delta \ F \ B \subseteq B \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *admit-B*:  $\langle B \subseteq \text{admit } F \ B \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *admit-Pow*:  $\langle B \subseteq \text{Pow } (\mathcal{W} \ F) \implies \text{admit } F \ B \subseteq \text{Pow } (\mathcal{W} \ F) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *admissible-grow*:  $\langle \text{admissible } F \ B \longleftrightarrow \text{grow } F \ B \ B = B \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *lfp-grow*:  $\langle \text{grow } F \ B \ (\text{lfp} (\text{grow } F \ B)) = \text{lfp} (\text{grow } F \ B) \rangle$

*<proof>*

**lemma** *admit-admissible*:  $\langle \text{admissible } F \text{ (admit } F B) \rangle$   
*<proof>*

**lemma** *admits-admit*:  $\langle B \neq \{\} \implies B \subseteq \text{Pow } (\mathcal{W} F) \implies \text{admits } F B \text{ (admit } F B) \rangle$   
*<proof>*

**lemma** *admissible-admit*:  
**assumes**  $\langle B \neq \{\} \rangle \langle B \subseteq \text{Pow } (\mathcal{W} F) \rangle$   
**shows**  
 $\langle \text{admit } F B \neq \{\} \rangle$   
 $\langle \text{admit } F B \subseteq \text{Pow } (\mathcal{W} F) \rangle$   
 $\langle \text{admissible } F \text{ (admit } F B) \rangle$   
*<proof>*

**abbreviation** *canonical-frame* ::  $\langle 'x \text{ lbd set} \Rightarrow ('x \text{ tm}) \text{ frame} \rangle$  **where**  
 $\langle \text{canonical-frame } S \equiv \langle \mathcal{W} = \{[k]_S \mid k. \text{True}\}, \mathcal{R} = \text{reach } S \rangle \rangle$

**abbreviation** *canonical-gframe* ::  $\langle 'x \text{ lbd set} \Rightarrow ('x \text{ tm}) \text{ gframe} \rangle$  **where**  
 $\langle \text{canonical-gframe } S \equiv \text{frame.extend (canonical-frame } S) \rangle$   
 $\langle \Pi = \text{admit (canonical-frame } S) \text{ (range (val } S)) \rangle \rangle$

**definition** *canonical* ::  $\langle 'x \text{ lbd set} \Rightarrow ('x, 'x \text{ tm}) \text{ model} \rangle$  **where**  
 $\langle \text{canonical } S \equiv$   
 $\text{gframe.extend (canonical-gframe } S)$   
 $\langle \mathcal{N} = \lambda i. [\circ i]_S,$   
 $\mathcal{N} = \lambda i. [\# i]_S,$   
 $\mathcal{V} = \text{val } S \circ \circ,$   
 $\mathcal{V} = \text{val } S \circ \#$   
 $\rangle \rangle$

**lemma** *wf-canonical-frame*:  $\langle \text{wf-frame (canonical-frame } S) \rangle$   
*<proof>*

**lemma** *val-Pow*:  $\langle \text{range (val } S) \subseteq \text{Pow } (\mathcal{W} \text{ (canonical-frame } S)) \rangle$   
*<proof>*

**lemma** *wf-cannonical-gframe*:  $\langle \text{wf-gframe (canonical-gframe } S) \rangle$   
*<proof>*

**lemma** *admits-val*:  $\langle \text{admits (canonical-frame } S) \text{ (range (val } S)) \text{ } PI \implies \text{val } S P \in PI \rangle$   
*<proof>*

**lemma** *admit-val*:  $\langle \text{val } S P \in \text{admit (canonical-frame } S) \text{ (range (val } S)) \rangle$   
*<proof>*

**lemma** *wf-canonical-env*:  $\langle \text{wf-env } (\text{canonical } S) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *wf-gframe-canonical*:  $\langle \text{wf-gframe } (\text{gframe.truncate } (\text{canonical } S)) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *wf-canonical*:  $\langle \text{wf-model } (\text{canonical } S) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *admissible-sqdfs*:  $\langle \text{admissible } (\text{canonical-frame } S) (\text{sqdfs } (\text{canonical } S)) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *sqdfs-Pow*:  $\langle \text{sqdfs } (\text{canonical } S) \subseteq \text{Pow } (\mathcal{W} (\text{canonical-frame } S)) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *val-sqdfs*:  $\langle \text{val } S \ P \in \text{sqdfs } (\text{canonical } S) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *admits-canonical-sqdfs*:  $\langle \text{admits } (\text{canonical-frame } S) (\text{range } (\text{val } S)) (\text{sqdfs } (\text{canonical } S)) \rangle$   
 $\langle \text{proof} \rangle$

**definition** *canonical-ctx* ::  $\langle 'x \text{ lbd set} \Rightarrow 'x \text{ tm} \Rightarrow ('x, 'x \text{ tm}) \text{ ctx} \rangle (\langle \llbracket -, \_ \rrbracket \rangle)$  **where**  
 $\langle \llbracket S, i \rrbracket \equiv (\text{canonical } S, [i]_S) \rangle$

**lemma** *sqdfs-canonical*:  $\langle \text{sqdfs } (\text{canonical } S) = \Pi (\text{canonical } S) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *canonical-tm-eta [simp]*:  $\langle \text{case-tm } (\lambda i. [\# i]_S) (\lambda n. [\circ n]_S) k = [k]_S \rangle$   
 $\langle \text{proof} \rangle$

**corollary** *canonical-tm-eta' [simp]*:  $\langle \text{case-tm } (\mathfrak{N} (\text{canonical } S)) (\mathcal{N} (\text{canonical } S))$   
 $k = [k]_S \rangle$   
 $\langle \text{proof} \rangle$

**locale** *MyHintikka* = *Hintikka map-lbd symbols-lbd*  $\langle \lambda -. \text{True} \rangle$  *Kinds* *S*  
**for** *S* ::  $\langle 'x \text{ lbd set} \rangle$   
**begin**

**lemmas**  
*confl* = *sat<sub>H</sub>[of C.kind]* **and**  
*alpha* = *sat<sub>H</sub>[of A.kind]* **and**  
*beta* = *sat<sub>H</sub>[of B.kind]* **and**  
*gammaI* = *sat<sub>H</sub>[of GI.kind]* **and**  
*gammaP* = *sat<sub>H</sub>[of GP.kind]* **and**  
*δ* = *sat<sub>H</sub>[of D.kind]*

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

**lemma** *Nom-sym*:

**assumes**  $\langle (i, \cdot k) \in S \rangle$   
**shows**  $\langle (k, \cdot i) \in S \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *Nom-trans*:

**assumes**  $\langle (i, \cdot j) \in S \rangle \langle (j, \cdot k) \in S \rangle$   
**shows**  $\langle (i, \cdot k) \in S \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *equiv-nom-ne*:  $\langle \{k. \text{equiv-nom } S \ i \ k\} \neq \{\} \rangle$

$\langle \text{proof} \rangle$

**lemma** *equiv-nom-assign*:  $\langle \text{equiv-nom } S \ i \ ([i]_S) \rangle$

$\langle \text{proof} \rangle$

**lemma** *equiv-nom-Nom*:

**assumes**  $\langle \text{equiv-nom } S \ i \ k \rangle \langle (i, p) \in S \rangle$   
**shows**  $\langle (k, p) \in S \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *assign-in-W*:  $\langle [i]_S \in \mathcal{W} \ (\text{canonical } S) \rangle$

$\langle \text{proof} \rangle$

**theorem** *model*:  $\langle ((i, p) \in S \longrightarrow \llbracket S, i \rrbracket \models p) \wedge ((i, \neg p) \in S \longrightarrow \neg \llbracket S, i \rrbracket \models p) \rangle$

$\langle \text{proof} \rangle$

**end**

**theorem** *model-existence*:

**fixes**  $S :: \langle 'x \text{ lbd set} \rangle$   
**assumes**  $\langle P.\text{prop}_E \ \text{Kinds } C \rangle$   
**and**  $\langle S \in C \rangle$   
**and**  $\langle P.\text{enough-new } S \rangle$   
**and**  $\langle (i, p) \in S \rangle$   
**shows**  $\langle \llbracket \text{mk-mcs } C \ S, i \rrbracket \models p \rangle$   
 $\langle \text{proof} \rangle$

## 4.6 Natural Deduction

**inductive** *Calculus-Set* ::  $\langle 'x \text{ lbd set} \Rightarrow 'x \text{ lbd} \Rightarrow \text{bool} \rangle \ (\langle \cdot \Vdash \cdot \rangle [50, 50] \ 50)$  **where**

*Assm* [*dest*]:  $\langle (i, p) \in A \Longrightarrow A \Vdash (i, p) \rangle$   
*Ref* [*simp*]:  $\langle A \Vdash (i, \cdot i) \rangle$   
*Nom* [*dest*]:  $\langle A \Vdash (i, \cdot k) \Longrightarrow A \Vdash (i, p) \Longrightarrow A \Vdash (k, p) \rangle$   
*NotI* [*intro*]:  $\langle \{(i, p)\} \cup A \Vdash (i, \perp) \Longrightarrow A \Vdash (i, \neg p) \rangle$   
*NotE* [*elim*]:  $\langle A \Vdash (i, \neg p) \Longrightarrow A \Vdash (i, p) \Longrightarrow A \Vdash (k, q) \rangle$   
*AndI* [*intro*]:  $\langle A \Vdash (i, p) \Longrightarrow A \Vdash (i, q) \Longrightarrow A \Vdash (i, p \wedge q) \rangle$   
*AndD1* [*dest*]:  $\langle A \Vdash (i, p \wedge q) \Longrightarrow A \Vdash (i, p) \rangle$

| *AndD2* [*dest*]:  $\langle A \Vdash (i, p \wedge q) \implies A \Vdash (i, q) \rangle$   
 | *SatI* [*intro*]:  $\langle A \Vdash (i, p) \implies A \Vdash (k, @i p) \rangle$   
 | *SatE* [*dest*]:  $\langle A \Vdash (i, @k p) \implies A \Vdash (k, p) \rangle$   
 | *BoxI* [*intro*]:  $\langle \{(i, \diamond (\cdot \circ k))\} \cup A \Vdash (\circ k, p) \implies k \notin \text{symbols} (\{(i, p)\} \cup A) \implies A \Vdash (i, \square p) \rangle$   
 | *BoxE* [*elim*]:  $\langle A \Vdash (i, \square p) \implies A \Vdash (i, \diamond (\cdot k)) \implies A \Vdash (k, p) \rangle$   
 | *GloI* [*intro*]:  $\langle A \Vdash (\circ k, p) \implies k \notin \text{symbols} (\{(i, p)\} \cup A) \implies A \Vdash (i, \mathbf{A} p) \rangle$   
 | *GloE* [*dest*]:  $\langle A \Vdash (i, \mathbf{A} p) \implies A \Vdash (k, p) \rangle$   
 | *DwnI* [*intro*]:  $\langle A \Vdash (i, \langle i \rangle_i p) \implies A \Vdash (i, \downarrow p) \rangle$   
 | *DwnE* [*dest*]:  $\langle A \Vdash (i, \downarrow p) \implies A \Vdash (i, \langle i \rangle_i p) \rangle$   
 | *AllI* [*intro*]:  $\langle A \Vdash (i, \langle \cdot (\circ P) \rangle_p p) \implies P \notin \text{symbols} (\{(i, p)\} \cup A) \implies A \Vdash (i, \forall p) \rangle$   
 | *AllE* [*dest*]:  $\langle A \Vdash (i, \forall p) \implies \text{softqdf } q \implies A \Vdash (i, \langle q \rangle_p p) \rangle$   
 | *Clas*:  $\langle \{(i, \neg p)\} \cup A \Vdash (i, p) \implies A \Vdash (i, p) \rangle$

### 4.6.1 Soundness

**theorem** *soundness*:

**assumes**  $\langle A \Vdash (i, p) \rangle \langle \forall (k, q) \in A. (\text{Model } W R P I N e V f, \text{case-tm } e N k) \models q \rangle$

$\langle \text{wf-model } (\text{Model } W R P I N e V f) \rangle$

**shows**  $\langle (\text{Model } W R P I N e V f, \text{case-tm } e N i) \models p \rangle$

$\langle \text{proof} \rangle$

**corollary** *soundness'*:

**assumes**  $\langle \{\} \Vdash (\circ i, p) \rangle \langle i \notin \text{symbols-fm } p \rangle$

**and**  $\langle \text{wf-model } M \rangle \langle w \in \mathcal{W} M \rangle$

**shows**  $\langle (M, w) \models p \rangle$

$\langle \text{proof} \rangle$

**lemma** *no-bot*:  $\langle \neg (M, w) \models \perp \rangle$

$\langle \text{proof} \rangle$

**corollary**  $\langle \neg (\{\} \Vdash (\circ i, \perp)) \rangle$

$\langle \text{proof} \rangle$

### 4.6.2 Derived Rules

**lemma** *Assm-head* [*simp*]:  $\langle \{(p, i)\} \cup A \Vdash (p, i) \rangle$

$\langle \text{proof} \rangle$

**lemma** *Boole*:  $\langle \{(i, \neg p)\} \cup A \Vdash (i, \perp) \implies A \Vdash (i, p) \rangle$

$\langle \text{proof} \rangle$

**lemma** *FlsE* [*dest*]:  $\langle A \Vdash (i, \perp) \implies A \Vdash (k, p) \rangle$

$\langle \text{proof} \rangle$

### 4.6.3 Derivational Consistency

**lemma** *calculus-conf*:

**assumes**  $\langle ps \rightsquigarrow_{\chi} qs \rangle \langle set\ ps \subseteq A \rangle \langle q \in set\ qs \rangle \langle q \in A \rangle$   
**shows**  $\langle A \Vdash (i, \perp) \rangle$   
 $\langle proof \rangle$

**lemma** *calculus-alpha:*

**assumes**  $\langle ps \rightsquigarrow_{\alpha} qs \rangle \langle set\ ps \subseteq A \rangle \langle set\ qs \cup A \Vdash (i, \perp) \rangle$   
**shows**  $\langle A \Vdash (i, \perp) \rangle$   
 $\langle proof \rangle$

**lemma** *calculus-beta:*

**assumes**  $\langle ps \rightsquigarrow_{\beta} qs \rangle \langle set\ ps \subseteq A \rangle \langle \forall q \in set\ qs. \{q\} \cup A \Vdash (i, \perp) \rangle$   
**shows**  $\langle A \Vdash (i, \perp) \rangle$   
 $\langle proof \rangle$

**lemma** *calculus-gammaI:*

**assumes**  $\langle ps \rightsquigarrow_{\gamma_i} qs \rangle \langle set\ ps \subseteq A \rangle \langle set\ (qs\ k) \cup A \Vdash (i, \perp) \rangle$   
**shows**  $\langle A \Vdash (i, \perp) \rangle$   
 $\langle proof \rangle$

**lemma** *calculus-gammaP:*

**assumes**  $\langle ps \rightsquigarrow_{\gamma_P} (F, qs) \rangle \langle set\ ps \subseteq A \rangle \langle k \in F\ A \rangle \langle set\ (qs\ k) \cup A \Vdash (i, \perp) \rangle$   
**shows**  $\langle A \Vdash (i, \perp) \rangle$   
 $\langle proof \rangle$

**lemma** *calculus-delta:*

**assumes**  $\langle p \in A \rangle \langle k \notin symbols\ A \rangle \langle set\ (\delta\ p\ k) \cup A \Vdash (a, \perp) \rangle$   
**shows**  $\langle A \Vdash (a, \perp) \rangle$   
 $\langle proof \rangle$

**interpretation** *DC: Derivational-Confl map-lbd symbols-lbd*  $\langle \lambda-. True \rangle$  *confl-class*

$\langle \lambda A. \neg A \Vdash (a, \perp) \rangle$   
 $\langle proof \rangle$

**interpretation** *DA: Derivational-Alpha map-lbd symbols-lbd*  $\langle \lambda-. True \rangle$  *alpha-class*

$\langle \lambda A. \neg A \Vdash (a, \perp) \rangle$   
 $\langle proof \rangle$

**interpretation** *DB: Derivational-Beta map-lbd symbols-lbd*  $\langle \lambda-. True \rangle$  *beta-class*

$\langle \lambda A. \neg A \Vdash (a, \perp) \rangle$   
 $\langle proof \rangle$

**interpretation** *DGI: Derivational-Gamma map-tm map-lbd symbols-lbd*  $\langle \lambda-. True \rangle$

*GI.classify-UNIV*  $\langle \lambda A. \neg A \Vdash (a, \perp) \rangle$   
 $\langle proof \rangle$

**interpretation** *DGP: Derivational-Gamma map-fm map-lbd symbols-lbd*  $\langle \lambda-. True \rangle$

*gamma-class-fm*  $\langle \lambda A. \neg A \Vdash (a, \perp) \rangle$   
 $\langle proof \rangle$

**interpretation** *DD: Derivational-Delta map-lbd symbols-lbd*  $\langle \lambda\text{-}. \text{True} \rangle \delta \langle \lambda A. \neg A \vdash (a, \perp) \rangle$   
 $\langle \text{proof} \rangle$

**interpretation** *Derivational-Consistency map-lbd symbols-lbd*  $\langle \lambda\text{-}. \text{True} \rangle \text{Kinds}$   
 $\langle \lambda A. \neg A \vdash (a, \perp) \rangle$   
 $\langle \text{proof} \rangle$

#### 4.6.4 Strong Completeness

**theorem** *strong-completeness:*

**fixes**  $p :: \langle 'x \text{ fm} \rangle$   
**assumes** *mod:*  $\langle \bigwedge (M :: \langle 'x, 'x \text{ tm} \rangle \text{ model}) w. \text{wf-model } M \implies w \in \mathcal{W} \ M \implies$   
 $\forall (k, q) \in A. (M, \text{case-tm } (\mathfrak{N} \ M) (\mathcal{N} \ M) \ k) \models q \implies$   
 $(M, w) \models p \rangle$   
**and**  $\langle P.\text{enough-new } A \rangle$   
**shows**  $\langle A \vdash (i, p) \rangle$   
 $\langle \text{proof} \rangle$

#### 4.7 Natural Deduction with Lists

**inductive** *Calculus* ::  $\langle 'x \text{ lbd list} \Rightarrow 'x \text{ lbd} \Rightarrow \text{bool} \rangle$  (**infix**  $\langle \vdash \rangle$  50) **where**

*Assm* [*dest*]:  $\langle (i, p) \in \text{set } A \implies A \vdash (i, p) \rangle$   
| *Ref* [*simp*]:  $\langle A \vdash (i, \cdot i) \rangle$   
| *Nom* [*dest*]:  $\langle A \vdash (i, \cdot k) \implies A \vdash (i, p) \implies A \vdash (k, p) \rangle$   
| *NotI* [*intro*]:  $\langle (i, p) \# A \vdash (i, \perp) \implies A \vdash (i, \neg p) \rangle$   
| *NotE* [*elim*]:  $\langle A \vdash (i, \neg p) \implies A \vdash (i, p) \implies A \vdash (k, q) \rangle$   
| *AndI* [*intro*]:  $\langle A \vdash (i, p) \implies A \vdash (i, q) \implies A \vdash (i, p \wedge q) \rangle$   
| *AndD1* [*dest*]:  $\langle A \vdash (i, p \wedge q) \implies A \vdash (i, p) \rangle$   
| *AndD2* [*dest*]:  $\langle A \vdash (i, p \wedge q) \implies A \vdash (i, q) \rangle$   
| *SatI* [*intro*]:  $\langle A \vdash (i, p) \implies A \vdash (k, @i \ p) \rangle$   
| *SatE* [*dest*]:  $\langle A \vdash (i, @k \ p) \implies A \vdash (k, p) \rangle$   
| *BoxI* [*intro*]:  $\langle (i, \diamond (\cdot (\circ k))) \# A \vdash (\circ k, p) \implies k \notin \text{symbols } (\{(i, p)\} \cup \text{set } A) \implies A \vdash (i, \square p) \rangle$   
| *BoxE* [*elim*]:  $\langle A \vdash (i, \square p) \implies A \vdash (i, \diamond (\cdot k)) \implies A \vdash (k, p) \rangle$   
| *GloI* [*intro*]:  $\langle A \vdash (\circ k, p) \implies k \notin \text{symbols } (\{(i, p)\} \cup \text{set } A) \implies A \vdash (i, \mathbf{A} \ p) \rangle$   
| *GloE* [*dest*]:  $\langle A \vdash (i, \mathbf{A} \ p) \implies A \vdash (k, p) \rangle$   
| *DwnI* [*intro*]:  $\langle A \vdash (i, \langle i \rangle_i \ p) \implies A \vdash (i, \downarrow p) \rangle$   
| *DwnE* [*dest*]:  $\langle A \vdash (i, \downarrow p) \implies A \vdash (i, \langle i \rangle_i \ p) \rangle$   
| *AllI* [*intro*]:  $\langle A \vdash (i, \langle \cdot (\circ P) \rangle_p \ p) \implies P \notin \text{symbols } (\{(i, p)\} \cup \text{set } A) \implies A \vdash (i, \forall p) \rangle$   
| *AllE* [*dest*]:  $\langle A \vdash (i, \forall p) \implies \text{softqdf } q \implies A \vdash (i, \langle q \rangle_p \ p) \rangle$   
| *Clas*:  $\langle (i, \neg p) \# A \vdash (i, p) \implies A \vdash (i, p) \rangle$

**definition** *bounded* ::  $\langle 'a \text{ list} \Rightarrow 'a \text{ set} \Rightarrow ('a \text{ list} \Rightarrow \text{bool}) \Rightarrow \text{bool} \rangle$  **where**

$\langle \text{bounded } K \ A \ P \equiv \text{set } K \subseteq A \wedge (\forall B. \text{set } K \subseteq \text{set } B \implies \text{set } B \subseteq A \implies P \ B) \rangle$

**lemma** *bounded-one* [*elim*]:

**assumes**  $\langle \text{bounded } K \ A \ P \rangle \langle \bigwedge A. P \ A \implies Q \ A \rangle$

**shows**  $\langle \text{bounded } K \ A \ Q \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *bounded-two* [*elim*]:  
**assumes**  $\langle \text{bounded } K \ A \ P \rangle \langle \text{bounded } K' \ A \ Q \rangle \langle \bigwedge A. P \ A \implies Q \ A \implies R \ A \rangle$   
**shows**  $\langle \text{bounded } (K \ @ \ K') \ A \ R \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *bounded-removeAll* [*dest*]:  
**assumes**  $\langle \text{bounded } K \ (\{p\} \cup A) \ P \rangle$   
**shows**  $\langle \text{bounded } (\text{removeAll } p \ K) \ A \ (\lambda B. P \ (p \ # \ B)) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *bounded-params*:  
**assumes**  $\langle a \notin P.\text{params} \ (\{p\} \cup A) \rangle \langle \text{bounded } K \ A \ P \rangle$   
**shows**  $\langle \text{bounded } K \ A \ (\lambda B. a \notin P.\text{params} \ (\text{set } (p \ # \ B))) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *finite-kernel*:  $\langle A \Vdash (i, p) \implies \exists K. \text{bounded } K \ A \ (\lambda B. B \vdash (i, p)) \rangle$   
 $\langle \text{proof} \rangle$

**corollary** *finite-assumptions*:  $\langle A \Vdash (i, p) \implies \exists B. \text{set } B \subseteq A \wedge B \vdash (i, p) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *calculus-set*:  $\langle A \vdash (i, p) \implies \text{set } A \Vdash (i, p) \rangle$   
 $\langle \text{proof} \rangle$

**corollary** *soundness-list*:  
**assumes**  $\langle A \vdash (i, p) \rangle \langle \forall (k, q) \in \text{set } A. (M, \text{case-tm } (\mathfrak{N} \ M) \ (\mathcal{N} \ M) \ k) \models q \rangle$   
**and**  $\langle \text{wf-model } M \rangle$   
**shows**  $\langle (M, \text{case-tm } (\mathfrak{N} \ M) \ (\mathcal{N} \ M) \ i) \models p \rangle$   
 $\langle \text{proof} \rangle$

**corollary** *soundness-nil*:  
 $\langle [] \vdash (\circ i, p) \implies i \notin \text{symbols-fm } p \implies \text{wf-model } M \implies w \in \mathcal{W} \ M \implies (M, w) \models p \rangle$   
 $\langle \text{proof} \rangle$

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

**corollary** *strong-completeness-list*:  
**fixes**  $p :: \langle 'x \ \text{fm} \rangle$   
**assumes**  $\langle \bigwedge (M :: ('x, 'x \ \text{tm}) \ \text{model}) \ w. \text{wf-model } M \implies w \in \mathcal{W} \ M \implies \forall (k, q) \in A. (M, \text{case-tm } (\mathfrak{N} \ M) \ (\mathcal{N} \ M) \ k) \models q \implies (M, w) \models p \rangle$   
**and**  $\langle P.\text{enough-new } A \rangle$   
**shows**  $\langle \exists B. \text{set } B \subseteq A \wedge B \vdash (i, p) \rangle$   
 $\langle \text{proof} \rangle$

**theorem** *main*:

**fixes**  $p :: \langle 'x \text{ fm} \rangle$   
**assumes**  $\langle i \notin \text{symbols-fm } p \rangle \langle |UNIV :: 'x \text{ lbd set}| \leq o \mid UNIV :: 'x \text{ set} \rangle$   
**shows**  $\langle [] \vdash (\odot i, p) \longleftrightarrow (\forall (M :: ('x, 'x \text{ tm}) \text{ model}). \forall w \in \mathcal{W} M. \text{wf-model } M \longrightarrow (M, w) \models p) \rangle$   
 $\langle \text{proof} \rangle$

## 4.8 The Need for SQDFs

### 4.8.1 Finite Unions of Arithmetic Progressions

**From Manuel Eberl's Furstenberg-Topology AFP entry**

**definition** *arith-prog* ::  $\text{int} \Rightarrow \text{nat} \Rightarrow \text{int set}$  **where**

$\text{arith-prog } a \ b = \{x. [x = a] \ (\text{mod } \text{int } b)\}$

**lemma** *arith-prog-0-right* [*simp*]:  $\text{arith-prog } a \ 0 = \{a\}$

$\langle \text{proof} \rangle$

**lemma** *arith-prog-Suc-0-right* [*simp*]:  $\text{arith-prog } a \ (\text{Suc } 0) = UNIV$

$\langle \text{proof} \rangle$

**lemma** *in-arith-progI* [*intro*]:  $[x = a] \ (\text{mod } b) \Longrightarrow x \in \text{arith-prog } a \ b$

$\langle \text{proof} \rangle$

**lemma** *arith-prog-disjoint*:

**assumes**  $[a \neq a'] \ (\text{mod } \text{int } b)$  **and**  $b > 0$

**shows**  $\text{arith-prog } a \ b \cap \text{arith-prog } a' \ b = \{\}$

$\langle \text{proof} \rangle$

**lemma** *arith-prog-dvd-mono*:  $b \ \text{dvd} \ b' \Longrightarrow \text{arith-prog } a \ b' \subseteq \text{arith-prog } a \ b$

$\langle \text{proof} \rangle$

**lemma** *bij-betw-arith-prog*:

**assumes**  $b > 0$

**shows**  $\text{bij-betw } (\lambda n. a + \text{int } b * n) \ UNIV \ (\text{arith-prog } a \ b)$

$\langle \text{proof} \rangle$

**lemma** *arith-prog-altdef*:  $\text{arith-prog } a \ b = \text{range } (\lambda n. a + \text{int } b * n)$

$\langle \text{proof} \rangle$

**lemma** *infinite-arith-prog*:  $b > 0 \Longrightarrow \text{infinite } (\text{arith-prog } a \ b)$

$\langle \text{proof} \rangle$

**lemma** *arith-prog-complement*:

**assumes**  $\langle b > 0 \rangle$

**shows**  $\neg \text{arith-prog } a \ b = \bigcup i \in \{1..<b\}. \text{arith-prog } (a + \text{int } i) \ b$

$\langle \text{proof} \rangle$

**lemma** *arith-prog-distinguish*:

**assumes**  $x \neq y$

**shows**  $\exists a c b. b > 0 \wedge x \in \text{arith-prog } a \ b \wedge y \in \text{arith-prog } c \ b \wedge \text{arith-prog } a \ b \cap \text{arith-prog } c \ b = \{\}$   
*<proof>*

## Unions of Arithmetic Progressions

**lemma** *arith-prog-offset-in*:  $\langle k \in \text{arith-prog } a \ b \implies \text{arith-prog } k \ b = \text{arith-prog } a \ b \rangle$   
*<proof>*

**lemma** *arith-prog-mod*:  $\langle \text{arith-prog } (a \ \text{mod } \text{int } b) \ b = \text{arith-prog } a \ b \rangle$   
*<proof>*

**lemma** *mod-bounds*:  $\langle b > 0 \implies a \ \text{mod } \text{int } b \geq 0 \wedge a \ \text{mod } \text{int } b < b \rangle$   
*<proof>*

**lemma** *mod-range*:  $\langle \{a \ \text{mod } \text{int } b \mid a. b > 0\} \subseteq \{0..<\text{int } b\} \rangle$   
*<proof>*

**lemma** *finite-mod-range*:  $\langle \text{finite } \{a \ \text{mod } \text{int } b \mid a. b > 0\} \rangle$   
*<proof>*

**definition** *arith* ::  $\langle \text{nat set} \Rightarrow \text{int set} \Rightarrow \text{bool} \rangle$  **where**  
 $\langle \text{arith } B \ U \equiv \forall a \in U. \exists b \in B. b > 0 \wedge \text{arith-prog } a \ b \subseteq U \rangle$

**lemma** *arith-mono*:  $\langle \text{arith } B \ U \implies B \subseteq C \implies \text{arith } C \ U \rangle$   
*<proof>*

**lemma** *arith-empty-steps*:  $\langle \text{arith } B \ U \implies B = \{\} \implies U = \{\} \rangle$   
*<proof>*

**lemma** *arith-ne-steps*:  $\langle \text{arith } B \ U \implies U \neq \{\} \implies B \neq \{\} \rangle$   
*<proof>*

**lemma** *arith-decomp*:

**assumes**  $\langle \text{arith } B \ U \rangle$

**obtains** *abs* **where**

$\langle \text{finite } B \implies \text{finite } \text{abs} \rangle$

$\langle \bigwedge a b. (a, b) \in \text{abs} \implies b > 0 \wedge b \in B \rangle$

$\langle U = (\bigcup (a, b) \in \text{abs}. \text{arith-prog } a \ b) \rangle$

*<proof>*

**lemma** *arith-UNIV*:  $\langle \text{arith } \{1\} \ \text{UNIV} \rangle$   
*<proof>*

**lemma** *arith-empty*:  $\text{arith } B \ \{\}$   
*<proof>*

**lemma** *finite-case-prod-lcm*:  $\langle \text{finite } B \implies \text{finite } C \implies \text{finite } (\text{case-prod lcm } \langle (B \times C) \rangle) \rangle$   
*<proof>*

**lemma** *arith-inter*:  
**assumes**  $U: \langle \text{arith } B \ U \rangle$  **and**  $V: \langle \text{arith } C \ V \rangle$   
**shows**  $\langle \text{arith } (\text{case-prod lcm } \langle (B \times C) \rangle) (U \cap V) \rangle$   
*<proof>*

**lemma** *arith-Inter*:  
**assumes**  $\langle \text{finite } X \rangle$  **and**  $X: \langle \forall U \in X. \text{arith } B \ U \rangle$   $\langle X \neq \{\} \rangle$   
**shows**  $\langle \exists B'. (\text{finite } B \longrightarrow \text{finite } B') \wedge \text{arith } B' (\bigcap X) \rangle$   
*<proof>*

**lemma** *arith-union*:  
**assumes**  $\langle \text{arith } B \ U \rangle$   $\langle \text{arith } C \ V \rangle$   
**shows**  $\langle \text{arith } (B \cup C) (U \cup V) \rangle$   
*<proof>*

**lemma** *arith-ne-infinite*:  
**assumes**  $\langle \text{arith } B \ U \rangle$   $\langle U \neq \{\} \rangle$   
**shows**  $\langle \text{infinite } U \rangle$   
*<proof>*

**lemma** *arith-prog-arith* [*intro*]:  
**assumes**  $\langle b > 0 \rangle$   
**shows**  $\langle \text{arith } \{b\} (\text{arith-prog } a \ b) \rangle$   
*<proof>*

**lemma** *arith-prog-complement-arith* [*intro*]:  
**assumes**  $\langle b > 0 \rangle$   
**shows**  $\langle \text{arith } \{b\} (\neg \text{arith-prog } a \ b) \rangle$   
*<proof>*

**lemma** *arith-complement-arith* [*intro*]:  
**assumes**  $\langle \text{arith } B \ U \rangle$   $\langle \text{finite } B \rangle$   
**shows**  $\langle \exists B'. \text{finite } B' \wedge \text{arith } B' (\neg U) \rangle$   
*<proof>*

**lemma** *arith-distinguish*:  
**assumes**  $\langle x \neq y \rangle$   
**shows**  $\langle \exists B \ U \ V. \text{finite } B \wedge \text{arith } B \ U \wedge \text{arith } B \ V \wedge x \in U \wedge y \in V \wedge U \cap V = \{\} \rangle$   
*<proof>*

## Finite Unions of Arithmetic Progressions

**definition** *fin-arith* ::  $\langle \text{int set} \Rightarrow \text{bool} \rangle$  **where**  
 $\langle \text{fin-arith } U \equiv \exists B. \text{finite } B \wedge \text{arith } B \ U \rangle$

**lemma** *fin-arith-UNIV* [intro]:  $\langle \text{fin-arith } \text{UNIV} \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *fin-arith-empty* [intro]:  $\langle \text{fin-arith } \{\} \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *fin-arith-inter* [intro]:  $\langle \text{fin-arith } U \Longrightarrow \text{fin-arith } V \Longrightarrow \text{fin-arith } (U \cap V) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *fin-arith-union* [intro]:  $\langle \text{fin-arith } U \Longrightarrow \text{fin-arith } V \Longrightarrow \text{fin-arith } (U \cup V) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *fin-arith-compl* [intro]:  $\langle \text{fin-arith } U \Longrightarrow \text{fin-arith } (- \ U) \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *fin-arith-distinguish*:

**assumes**  $\langle x \neq y \rangle$

**shows**  $\langle \exists U \ V. \text{fin-arith } U \wedge \text{fin-arith } V \wedge x \in U \wedge y \in V \wedge U \cap V = \{\} \rangle$

$\langle \text{proof} \rangle$

## Singletons

**lemma** *arith-prog-singleton*:  $\langle \bigcap \{ \text{arith-prog } a \ b \mid a \ b. \ b > 0 \wedge x \in \text{arith-prog } a \ b \} = \{x\} \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *fin-arith-Inter-singleton*:  $\langle \bigcap \{ U \mid U. \text{fin-arith } U \wedge x \in U \} = \{x\} \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *singleton-not-finarith*:  $\langle \neg \text{fin-arith } \{x\} \rangle$   
 $\langle \text{proof} \rangle$

## 4.8.2 Counterexample

**definition** *Pss* ::  $\langle \text{int set set} \rangle$  **where**  
 $\langle Pss \equiv \{ U. \text{fin-arith } U \} \rangle$

**lemma** *Pss-empty*:  $\langle \{\} \in Pss \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *Pss-UNIV*:  $\langle \text{UNIV} \in Pss \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *Pss-union*:  $\langle X \in Pss \Longrightarrow Y \in Pss \Longrightarrow X \cup Y \in Pss \rangle$   
 $\langle \text{proof} \rangle$



*<proof>*

However, propositional quantification lets us form a singleton.

**abbreviation**  $\langle \text{singleton } x \equiv \forall ( @(\circ x) (\cdot(\#0)) \longrightarrow \cdot(\#0) ) \rangle$

**lemma** *singleton*:  $\langle ((\text{my-model}, x) \models \text{singleton } y) \longleftrightarrow x = y \rangle$

*<proof>*

**lemma** *fin-arith-distinguish'*:

$\langle \forall P. \text{fin-arith } P \longrightarrow y \in P \longrightarrow v \in P \implies v \neq w \implies \exists P. y \in P \wedge \text{fin-arith } P \wedge w \notin P \rangle$

*<proof>*

The singleton does not hold nowhere-or-twice.

**lemma** *not-nowhere-or-twice-singleton*:  $\langle \neg ((\text{my-model}, x) \models \text{nowhere-or-twice } (\text{singleton } y)) \rangle$

*<proof>*

So we cannot always eliminate a quantifier with a non-quantifier-free formula.

**theorem** *counter*:

**shows**  $\langle \neg ((\text{my-model}, x) \models \forall (\text{nowhere-or-twice } (\cdot(\#0))) \longrightarrow \text{nowhere-or-twice } (\text{singleton } y)) \rangle$

*<proof>*

**end**