

# Syntax-Independent Logic Infrastructure

Andrei Popescu      Dmitriy Traytel

March 29, 2023

## Abstract

We formalize a notion of logic whose terms and formulas are kept abstract. In particular, logical connectives, substitution, free variables, and provability are not defined, but characterized by their general properties as locale assumptions. Based on this abstract characterization, we develop further reusable reasoning infrastructure. For example, we define parallel substitution (along with proving its characterizing theorems) from single-point substitution. Similarly, we develop a natural deduction style proof system starting from the abstract Hilbert-style one. These one-time efforts benefit different concrete logics satisfying our locales' assumptions.

We instantiate the syntax-independent logic infrastructure to Robinson arithmetic (also known as  $Q$ ) in the AFP entry [Robinson\\_Arithmetic](#) and to hereditarily finite set theory in the AFP entries [Goedel\\_HFSet\\_Semantic](#) and [Goedel\\_HFSet\\_Semanticless](#), which are part of our formalization of Gödel's Incompleteness Theorems described in our CADE-27 paper [1].

# Contents

<b>1</b>	<b>Preliminaries</b>	<b>3</b>
1.1	Triviality	3
1.2	Some Proof Infrastructure	4
<b>2</b>	<b>Syntax</b>	<b>6</b>
2.1	Generic Syntax	6
2.1.1	Instance Operator	8
2.1.2	Fresh Variables	9
2.1.3	Parallel Term Substitution	10
2.1.4	Parallel Formula Substitution	11
2.2	Adding Numerals to the Generic Syntax	15
2.3	Adding Connectives and Quantifiers	15
2.3.1	Iterated conjunction	20
2.3.2	Parallel substitution versus the new connectives	20
2.4	Adding Disjunction	21
2.4.1	Iterated disjunction	22
2.4.2	Parallel substitution versus the new connectives	22
2.5	Adding an Ordering-Like Formula	22
2.6	Allowing the Renaming of Quantified Variables	24
2.7	The Exists-Unique Quantifier	24
<b>3</b>	<b>Deduction</b>	<b>26</b>
3.1	Positive Logic Deduction	26
3.1.1	Properties of the propositional fragment	27
3.1.2	Properties involving quantifiers	33
3.1.3	Properties concerning equality	35
3.1.4	The equivalence between soft substitution and substitution	37
3.2	Deduction Considering False	37
3.2.1	Basic properties of False (fls)	38
3.2.2	Properties involving negation	38
3.2.3	Properties involving True (tru)	40
3.2.4	Property of set-based conjunctions	41
3.2.5	Consistency and $\omega$ -consistency	43
3.3	Deduction Considering False and Disjunction	45
3.3.1	Disjunction vs. disjunction	45
3.3.2	Disjunction vs. conjunction	46
3.3.3	Disjunction vs. True and False	47
3.3.4	Set-based disjunctions	47
3.4	Deduction with Quantified Variable Renaming Included	49
3.5	Deduction with PseudoOrder Axioms Included	49

<b>4</b>	<b>Natural Deduction</b>	<b>51</b>
4.1	Natural Deduction from the Hilbert System . . . . .	51
4.2	Structural Rules for the Natural Deduction Relation . . . . .	51
4.3	Back and Forth between Hilbert and Natural Deduction . . . . .	52
4.4	More Structural Properties . . . . .	52
4.5	Properties Involving Substitution . . . . .	54
4.6	Introduction and Elimination Rules . . . . .	54
4.7	Adding Lemmas of Various Shapes into the Proof Context . . . . .	58
4.8	Rules for Equality . . . . .	60
4.9	Other Rules . . . . .	60
4.10	Natural Deduction for the Exists-Unique Quantifier . . . . .	61
4.11	Eisbach Notation for Natural Deduction Proofs . . . . .	62
<b>5</b>	<b>Pseudo-Terms</b>	<b>63</b>
5.1	Basic Setting . . . . .	63
5.2	The $\forall$ - $\exists$ Equivalence . . . . .	64
5.3	Instantiation . . . . .	65
5.3.1	Instantiation with terms . . . . .	65
5.3.2	Instantiation with pseudo-terms . . . . .	66
5.3.3	Closure and compositionality properties of instantiation . . . . .	66
5.4	Equality between Pseudo-Terms and Terms . . . . .	67
<b>6</b>	<b>Truth in a Standard Model</b>	<b>69</b>
<b>7</b>	<b>Arithmetic Constructs</b>	<b>72</b>
7.1	Arithmetic Terms . . . . .	74
7.2	The (Nonstrict and Strict) Order Relations . . . . .	80
7.3	Bounded Quantification . . . . .	82
<b>8</b>	<b>Deduction in a System Embedding the Intuitionistic Robinson Arithmetic</b>	<b>84</b>
8.1	Natural Deduction with the Bounded Quantifiers . . . . .	84
8.2	Deduction with the Robinson Arithmetic Axioms . . . . .	85
8.3	Properties Provable in $Q$ . . . . .	90
8.3.1	General properties, unconstrained by numerals . . . . .	90
8.3.2	Representability properties . . . . .	91
8.3.3	The "order-adequacy" properties . . . . .	92
8.3.4	Verifying the abstract ordering assumptions . . . . .	96

# Chapter 1

## Preliminaries

### 1.1 Trivia

**abbreviation** (*input*) *any* **where** *any*  $\equiv$  *undefined*

**lemma** *Un\_Diff2*:  $B \cap C = \{\} \implies A \cup B - C = A - C \cup B$  *<proof>*

**lemma** *Diff\_Diff\_Un*:  $A - B - C = A - (B \cup C)$  *<proof>*

**fun** *first* :: *nat*  $\Rightarrow$  *nat list* **where**

*first* 0 = []  
| *first* (Suc n) = n # *first* n

Facts about zipping lists:

**lemma** *fst\_set\_zip\_map\_fst*:

*length xs = length ys  $\implies$  fst ' (set (zip (map fst xs) ys)) = fst ' (set xs)*  
*<proof>*

**lemma** *snd\_set\_zip\_map\_snd*:

*length xs = length ys  $\implies$  snd ' (set (zip xs (map snd ys))) = snd ' (set ys)*  
*<proof>*

**lemma** *snd\_set\_zip*:

*length xs = length ys  $\implies$  snd ' (set (zip xs ys)) = set ys*  
*<proof>*

**lemma** *set\_zip\_D*:  $(x, y) \in \text{set } (\text{zip } xs \text{ } ys) \implies x \in \text{set } xs \wedge y \in \text{set } ys$

*<proof>*

**lemma** *inj\_on\_set\_zip\_map*:

**assumes** *i*: *inj\_on* *f* *X*

**and** *a*:  $(f \ x1, \ y1) \in \text{set } (\text{zip } (\text{map } f \ xs) \ ys)$   $\text{set } xs \subseteq X$   $x1 \in X$   $\text{length } xs = \text{length } ys$

**shows**  $(x1, \ y1) \in \text{set } (\text{zip } xs \ ys)$

*<proof>*

**lemma** *set\_zip\_map\_fst\_snd*:

**assumes**  $(u, x) \in \text{set } (\text{zip } us \ (\text{map } \text{snd } txs))$

**and**  $(t, u) \in \text{set } (\text{zip } (\text{map } \text{fst } txs) \ us)$

**and** *distinct*  $(\text{map } \text{snd } txs)$

**and** *distinct* *us* **and**  $\text{length } us = \text{length } txs$

**shows**  $(t, x) \in \text{set } txs$

*<proof>*

**lemma** *set\_zip\_map\_fst\_snd2*:  
**assumes**  $(u, x) \in \text{set } (\text{zip } us \ (\text{map } \text{snd } txs))$   
**and**  $(t, x) \in \text{set } txs$   
**and** *distinct*  $(\text{map } \text{snd } txs)$   
**and** *distinct*  $us$  **and**  $\text{length } us = \text{length } txs$   
**shows**  $(t, u) \in \text{set } (\text{zip } (\text{map } \text{fst } txs) \ us)$   
 $\langle \text{proof} \rangle$

**lemma** *set\_zip\_length\_map*:  
**assumes**  $(x1, y1) \in \text{set } (\text{zip } xs \ ys)$  **and**  $\text{length } xs = \text{length } ys$   
**shows**  $(f \ x1, y1) \in \text{set } (\text{zip } (\text{map } f \ xs) \ ys)$   
 $\langle \text{proof} \rangle$

**definition** *asList* :: 'a set  $\Rightarrow$  'a list **where**  
*asList*  $A \equiv \text{SOME } as. \text{set } as = A$

**lemma** *asList[simp,intro!]*:  $\text{finite } A \Longrightarrow \text{set } (\text{asList } A) = A$   
 $\langle \text{proof} \rangle$

**lemma** *triv\_Un\_imp\_aux*:  
 $(\bigwedge a. \varphi \Longrightarrow a \notin A \Longrightarrow a \in B \longleftrightarrow a \in C) \Longrightarrow \varphi \longrightarrow A \cup B = A \cup C$   
 $\langle \text{proof} \rangle$

**definition** *toN* **where**  $\text{toN } n \equiv [0..<(\text{Suc } n)]$

**lemma** *set\_toN[simp]*:  $\text{set } (\text{toN } n) = \{0..n\}$   
 $\langle \text{proof} \rangle$

**declare** *list.map\_cong*[*cong*]

## 1.2 Some Proof Infrastructure

$\langle ML \rangle$

**method** *RULE* **methods** *meth* **uses** *rule* =  
 $(\text{rule } rule; (\text{solves } meth)?)$

TryUntilFail:

**method** *TUF* **methods** *meth* =  
 $((meth;fail)+)?$

Helping a method, usually simp or auto, with specific substitutions inserted. For auto, this is a bit like a "simp!" analogue of "intro!" and "dest!": It forces the application of an indicated simplification rule, if this is possible.

**method** *variousSubsts1* **methods** *meth* **uses** *s1* =  
 $(meth?, (subst \ s1)?, meth?)$

**method** *variousSubsts2* **methods** *meth* **uses** *s1 s2* =  
 $(meth?, (subst \ s1)?, meth?, subst \ s2, meth?)$

**method** *variousSubsts3* **methods** *meth* **uses** *s1 s2 s3* =  
 $(meth?, (subst \ s1)?, meth?, (subst \ s2)?, meth?, (subst \ s3)?, meth?)$

**method** *variousSubsts4* **methods** *meth* **uses** *s1 s2 s3 s4* =  
 $(meth?, (subst \ s1)?, meth?, (subst \ s2)?, meth?, (subst \ s3)?, meth?, (subst \ s4)?, meth?)$

**method** *variousSubsts5* **methods** *meth* **uses** *s1 s2 s3 s4 s5* =  
 $(meth?, (subst \ s1)?, meth?, (subst \ s2)?, meth?, (subst \ s3)?, meth?, (subst \ s4)?, meth?, (subst \ s5)?, meth?)$

**method** *variousSubsts6* **methods** *meth* **uses** *s1 s2 s3 s4 s5 s6* =  
 $(meth?, (subst \ s1)?, meth?, (subst \ s2)?, meth?, (subst \ s3)?, meth?,$

*(subst s4)?, meth?, (subst s5)?, meth?, (subst s6)?, meth?*

# Chapter 2

## Syntax

### 2.1 Generic Syntax

We develop some generic (meta-)axioms for syntax and substitution. We only assume that the syntax of our logic has notions of variable, term and formula, which *include* subsets of "numeric" variables, terms and formulas, the latter being endowed with notions of free variables and substitution subject to some natural properties.

**locale** *Generic\_Syntax* =

**fixes**

*var* :: 'var set — numeric variables (i.e., variables ranging over numbers)  
**and** *trm* :: 'trm set — numeric trms, which include the numeric variables  
**and** *fmla* :: 'fmla set — numeric formulas  
**and** *Var* :: 'var  $\Rightarrow$  'trm — trms include at least the variables  
**and** *FvarsT* :: 'trm  $\Rightarrow$  'var set — free variables for trms  
**and** *substT* :: 'trm  $\Rightarrow$  'trm  $\Rightarrow$  'var  $\Rightarrow$  'trm — substitution for trms  
**and** *Fvars* :: 'fmla  $\Rightarrow$  'var set — free variables for formulas  
**and** *subst* :: 'fmla  $\Rightarrow$  'trm  $\Rightarrow$  'var  $\Rightarrow$  'fmla — substitution for formulas

**assumes**

*infinite\_var*: *infinite var* — the variables are assumed infinite  
**and** — Assumptions about the infrastructure (free vars, substitution and the embedding of variables into trms. NB: We need fewer assumptions for trm substitution than for formula substitution!

*Var[simp,intro!]*:  $\bigwedge x. x \in \text{var} \implies \text{Var } x \in \text{trm}$

**and**

*inj\_Var[simp]*:  $\bigwedge x y. x \in \text{var} \implies y \in \text{var} \implies (\text{Var } x = \text{Var } y \longleftrightarrow x = y)$

**and**

*finite\_FvarsT*:  $\bigwedge t. t \in \text{trm} \implies \text{finite } (\text{FvarsT } t)$

**and**

*FvarsT*:  $\bigwedge t. t \in \text{trm} \implies \text{FvarsT } t \subseteq \text{var}$

**and**

*substT[simp,intro]*:  $\bigwedge t1 t x. t1 \in \text{trm} \implies t \in \text{trm} \implies x \in \text{var} \implies \text{substT } t1 t x \in \text{trm}$

**and**

*FvarsT\_Var[simp]*:  $\bigwedge x. x \in \text{var} \implies \text{FvarsT } (\text{Var } x) = \{x\}$

**and**

*substT\_Var[simp]*:  $\bigwedge x t y. x \in \text{var} \implies y \in \text{var} \implies t \in \text{trm} \implies$

$\text{substT } (\text{Var } x) t y = (\text{if } x = y \text{ then } t \text{ else } \text{Var } x)$

**and**

*substT\_notIn[simp]*:

$\bigwedge t1 t2 x. x \in \text{var} \implies t1 \in \text{trm} \implies t2 \in \text{trm} \implies x \notin \text{FvarsT } t1 \implies \text{substT } t1 t2 x = t1$

**and**

— Assumptions about the infrastructure (free vars and substitution) on formulas

*finite\_Fvars*:  $\bigwedge \varphi. \varphi \in \text{fmla} \implies \text{finite } (\text{Fvars } \varphi)$

**and**



**Fvars**:  $\bigwedge \varphi. \varphi \in \text{fmla} \implies \text{Fvars } \varphi \subseteq \text{var}$   
**and**  
**subst[simp,intro]**:  $\bigwedge \varphi \ t \ x. \varphi \in \text{fmla} \implies t \in \text{trm} \implies x \in \text{var} \implies \text{subst } \varphi \ t \ x \in \text{fmla}$   
**and**  
**Fvars\_subst\_in**:  
 $\bigwedge \varphi \ t \ x. \varphi \in \text{fmla} \implies t \in \text{trm} \implies x \in \text{var} \implies x \in \text{Fvars } \varphi \implies$   
 $\text{Fvars } (\text{subst } \varphi \ t \ x) = \text{Fvars } \varphi - \{x\} \cup \text{FvarsT } t$   
**and**  
**subst\_compose\_eq\_or**:  
 $\bigwedge \varphi \ t1 \ t2 \ x1 \ x2. \varphi \in \text{fmla} \implies t1 \in \text{trm} \implies t2 \in \text{trm} \implies x1 \in \text{var} \implies x2 \in \text{var} \implies$   
 $x1 = x2 \vee x2 \notin \text{Fvars } \varphi \implies \text{subst } (\text{subst } \varphi \ t1 \ x1) \ t2 \ x2 = \text{subst } \varphi \ (\text{substT } t1 \ t2 \ x2) \ x1$   
**and**  
**subst\_compose\_diff**:  
 $\bigwedge \varphi \ t1 \ t2 \ x1 \ x2. \varphi \in \text{fmla} \implies t1 \in \text{trm} \implies t2 \in \text{trm} \implies x1 \in \text{var} \implies x2 \in \text{var} \implies$   
 $x1 \neq x2 \implies x1 \notin \text{FvarsT } t2 \implies$   
 $\text{subst } (\text{subst } \varphi \ t1 \ x1) \ t2 \ x2 = \text{subst } (\text{subst } \varphi \ t2 \ x2) \ (\text{substT } t1 \ t2 \ x2) \ x1$   
**and**  
**subst\_same\_Var[simp]**:  
 $\bigwedge \varphi \ x. \varphi \in \text{fmla} \implies x \in \text{var} \implies \text{subst } \varphi \ (\text{Var } x) \ x = \varphi$   
**and**  
**subst\_notIn[simp]**:  
 $\bigwedge x \ \varphi \ t. \varphi \in \text{fmla} \implies t \in \text{trm} \implies x \in \text{var} \implies x \notin \text{Fvars } \varphi \implies \text{subst } \varphi \ t \ x = \varphi$   
**begin**

**lemma var\_NE**:  $\text{var} \neq \{\}$   
 $\langle \text{proof} \rangle$

**lemma Var\_injD**:  $\text{Var } x = \text{Var } y \implies x \in \text{var} \implies y \in \text{var} \implies x = y$   
 $\langle \text{proof} \rangle$

**lemma FvarsT\_VarD**:  $x \in \text{FvarsT } (\text{Var } y) \implies y \in \text{var} \implies x = y$   
 $\langle \text{proof} \rangle$

**lemma FvarsT'**:  $t \in \text{trm} \implies x \in \text{FvarsT } t \implies x \in \text{var}$   
 $\langle \text{proof} \rangle$

**lemma Fvars'**:  $\varphi \in \text{fmla} \implies x \in \text{Fvars } \varphi \implies x \in \text{var}$   
 $\langle \text{proof} \rangle$

**lemma Fvars\_subst[simp]**:  
 $\varphi \in \text{fmla} \implies t \in \text{trm} \implies x \in \text{var} \implies$   
 $\text{Fvars } (\text{subst } \varphi \ t \ x) = (\text{Fvars } \varphi - \{x\}) \cup (\text{if } x \in \text{Fvars } \varphi \text{ then } \text{FvarsT } t \text{ else } \{\})$   
 $\langle \text{proof} \rangle$

**lemma in\_Fvars\_substD**:  
 $y \in \text{Fvars } (\text{subst } \varphi \ t \ x) \implies \varphi \in \text{fmla} \implies t \in \text{trm} \implies x \in \text{var}$   
 $\implies y \in (\text{Fvars } \varphi - \{x\}) \cup (\text{if } x \in \text{Fvars } \varphi \text{ then } \text{FvarsT } t \text{ else } \{\})$   
 $\langle \text{proof} \rangle$

**lemma inj\_on\_Var**:  $\text{inj\_on } \text{Var } \text{var}$   
 $\langle \text{proof} \rangle$

**lemma subst\_compose\_same**:  
 $\bigwedge \varphi \ t1 \ t2 \ x. \varphi \in \text{fmla} \implies t1 \in \text{trm} \implies t2 \in \text{trm} \implies x \in \text{var} \implies$   
 $\text{subst } (\text{subst } \varphi \ t1 \ x) \ t2 \ x = \text{subst } \varphi \ (\text{substT } t1 \ t2 \ x) \ x$   
 $\langle \text{proof} \rangle$

**lemma subst\_subst[simp]**:

**assumes**  $\varphi[simp]: \varphi \in fmla$  **and**  $t[simp]: t \in trm$  **and**  $x[simp]: x \in var$  **and**  $y[simp]: y \in var$   
**assumes**  $yy: x \neq y \ y \notin Fvars \ \varphi$   
**shows**  $subst (subst \ \varphi \ (Var \ y) \ x) \ t \ y = subst \ \varphi \ t \ x$   
 $\langle proof \rangle$

**lemma** *subst\_comp*:

$\bigwedge x \ y \ \varphi \ t. \ \varphi \in fmla \implies t \in trm \implies x \in var \implies y \in var \implies$   
 $x \neq y \implies y \notin FvarsT \ t \implies$   
 $subst (subst \ \varphi \ (Var \ x) \ y) \ t \ x = subst (subst \ \varphi \ t \ x) \ t \ y$   
 $\langle proof \rangle$

**lemma** *exists\_nat\_var*:

$\exists f::nat \Rightarrow 'var. \ inj \ f \ \wedge \ range \ f \ \subseteq \ var$   
 $\langle proof \rangle$

**definition** *Variable*  $:: \ nat \Rightarrow 'var$  **where**

$Variable = (SOME \ f. \ inj \ f \ \wedge \ range \ f \ \subseteq \ var)$

**lemma** *Variable\_inj\_var*:

$inj \ Variable \ \wedge \ range \ Variable \ \subseteq \ var$   
 $\langle proof \rangle$

**lemma** *inj\_Variable[simp]*:  $\bigwedge i \ j. \ Variable \ i = Variable \ j \longleftrightarrow i = j$

**and**  $Variable[simp,intro!]: \bigwedge i. \ Variable \ i \in var$   
 $\langle proof \rangle$

Convenient notations for some variables We reserve the first 10 indexes for any special variables we may wish to consider later.

**abbreviation** *xx* **where**  $xx \equiv Variable \ 10$

**abbreviation** *yy* **where**  $yy \equiv Variable \ 11$

**abbreviation** *zz* **where**  $zz \equiv Variable \ 12$

**abbreviation** *xx'* **where**  $xx' \equiv Variable \ 13$

**abbreviation** *yy'* **where**  $yy' \equiv Variable \ 14$

**abbreviation** *zz'* **where**  $zz' \equiv Variable \ 15$

**lemma** *xx*:  $xx \in var$

**and** *yy*:  $yy \in var$

**and** *zz*:  $zz \in var$

**and** *xx'*:  $xx' \in var$

**and** *yy'*:  $yy' \in var$

**and** *zz'*:  $zz' \in var$

$\langle proof \rangle$

**lemma** *vars\_distinct[simp]*:

$xx \neq yy \ yy \neq xx \ xx \neq zz \ zz \neq xx \ xx \neq xx' \ xx' \neq xx \ xx \neq yy' \ yy' \neq xx \ xx \neq zz' \ zz' \neq xx$

$yy \neq zz \ zz \neq yy \ yy \neq xx' \ xx' \neq yy \ yy \neq yy' \ yy' \neq yy \ yy \neq zz' \ zz' \neq yy$

$zz \neq xx' \ xx' \neq zz \ zz \neq yy' \ yy' \neq zz \ zz \neq zz' \ zz' \neq zz$

$xx' \neq yy' \ yy' \neq xx' \ xx' \neq zz' \ zz' \neq xx'$

$yy' \neq zz' \ zz' \neq yy'$

$\langle proof \rangle$

### 2.1.1 Instance Operator

**definition** *inst*  $:: \ fmla \Rightarrow 'trm \Rightarrow 'fmla$  **where**

$inst \ \varphi \ t = subst \ \varphi \ t \ xx$

**lemma** *inst[simp]*:  $\varphi \in fmla \implies t \in trm \implies inst \ \varphi \ t \in fmla$

$\langle \text{proof} \rangle$

**definition**  $\text{getFresh} :: 'var \text{ set} \Rightarrow 'var \text{ where}$   
 $\text{getFresh } V = (\text{SOME } x. x \in \text{var} \wedge x \notin V)$

**lemma**  $\text{getFresh}$ :  $\text{finite } V \Longrightarrow \text{getFresh } V \in \text{var} \wedge \text{getFresh } V \notin V$   
 $\langle \text{proof} \rangle$

**definition**  $\text{getFr} :: 'var \text{ list} \Rightarrow 'trm \text{ list} \Rightarrow 'fmla \text{ list} \Rightarrow 'var \text{ where}$   
 $\text{getFr } xs \ ts \ \varphi s =$   
 $\text{getFresh } (\text{set } xs \cup (\bigcup (\text{FvarsT } ' \text{ set } ts)) \cup (\bigcup (\text{Fvars } ' \text{ set } \varphi s)))$

**lemma**  $\text{getFr\_FvarsT\_Fvars}$ :  
**assumes**  $\text{set } xs \subseteq \text{var}$   $\text{set } ts \subseteq \text{trm}$  **and**  $\text{set } \varphi s \subseteq \text{fmla}$   
**shows**  $\text{getFr } xs \ ts \ \varphi s \in \text{var} \wedge$   
 $\text{getFr } xs \ ts \ \varphi s \notin \text{set } xs \wedge$   
 $(t \in \text{set } ts \longrightarrow \text{getFr } xs \ ts \ \varphi s \notin \text{FvarsT } t) \wedge$   
 $(\varphi \in \text{set } \varphi s \longrightarrow \text{getFr } xs \ ts \ \varphi s \notin \text{Fvars } \varphi)$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{getFr}[\text{simp}, \text{intro}]$ :  
**assumes**  $\text{set } xs \subseteq \text{var}$   $\text{set } ts \subseteq \text{trm}$  **and**  $\text{set } \varphi s \subseteq \text{fmla}$   
**shows**  $\text{getFr } xs \ ts \ \varphi s \in \text{var}$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{getFr\_var}$ :  
**assumes**  $\text{set } xs \subseteq \text{var}$   $\text{set } ts \subseteq \text{trm}$  **and**  $\text{set } \varphi s \subseteq \text{fmla}$  **and**  $t \in \text{set } ts$   
**shows**  $\text{getFr } xs \ ts \ \varphi s \notin \text{set } xs$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{getFr\_FvarsT}$ :  
**assumes**  $\text{set } xs \subseteq \text{var}$   $\text{set } ts \subseteq \text{trm}$  **and**  $\text{set } \varphi s \subseteq \text{fmla}$  **and**  $t \in \text{set } ts$   
**shows**  $\text{getFr } xs \ ts \ \varphi s \notin \text{FvarsT } t$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{getFr\_Fvars}$ :  
**assumes**  $\text{set } xs \subseteq \text{var}$   $\text{set } ts \subseteq \text{trm}$  **and**  $\text{set } \varphi s \subseteq \text{fmla}$  **and**  $\varphi \in \text{set } \varphi s$   
**shows**  $\text{getFr } xs \ ts \ \varphi s \notin \text{Fvars } \varphi$   
 $\langle \text{proof} \rangle$

## 2.1.2 Fresh Variables

**fun**  $\text{getFreshN} :: 'var \text{ set} \Rightarrow \text{nat} \Rightarrow 'var \text{ list} \text{ where}$   
 $\text{getFreshN } V \ 0 = []$   
 $| \text{getFreshN } V \ (\text{Suc } n) = (\text{let } u = \text{getFresh } V \text{ in } u \# \text{getFreshN } (\text{insert } u \ V) \ n)$

**lemma**  $\text{getFreshN}$ :  $\text{finite } V \Longrightarrow$   
 $\text{set } (\text{getFreshN } V \ n) \subseteq \text{var} \wedge \text{set } (\text{getFreshN } V \ n) \cap V = \{\} \wedge \text{length } (\text{getFreshN } V \ n) = n \wedge \text{distinct}$   
 $(\text{getFreshN } V \ n)$   
 $\langle \text{proof} \rangle$

**definition**  $\text{getFrN} :: 'var \text{ list} \Rightarrow 'trm \text{ list} \Rightarrow 'fmla \text{ list} \Rightarrow \text{nat} \Rightarrow 'var \text{ list} \text{ where}$   
 $\text{getFrN } xs \ ts \ \varphi s \ n =$   
 $\text{getFreshN } (\text{set } xs \cup (\bigcup (\text{FvarsT } ' \text{ set } ts)) \cup (\bigcup (\text{Fvars } ' \text{ set } \varphi s))) \ n$

**lemma**  $\text{getFrN\_FvarsT\_Fvars}$ :  
**assumes**  $\text{set } xs \subseteq \text{var}$   $\text{set } ts \subseteq \text{trm}$  **and**  $\text{set } \varphi s \subseteq \text{fmla}$   
**shows**  $\text{set } (\text{getFrN } xs \ ts \ \varphi s \ n) \subseteq \text{var} \wedge$

$set (getFrN\ xs\ ts\ \varphi\ n) \cap set\ xs = \{\}$   $\wedge$   
 $(t \in set\ ts \longrightarrow set (getFrN\ xs\ ts\ \varphi\ n) \cap FvarsT\ t = \{\}) \wedge$   
 $(\varphi \in set\ \varphi\ s \longrightarrow set (getFrN\ xs\ ts\ \varphi\ n) \cap Fvars\ \varphi = \{\}) \wedge$   
 $length (getFrN\ xs\ ts\ \varphi\ n) = n \wedge$   
 $distinct (getFrN\ xs\ ts\ \varphi\ n)$   
 <proof>

**lemma** *getFrN[simp,intro]*:  
**assumes**  $set\ xs \subseteq var\ set\ ts \subseteq trm$  **and**  $set\ \varphi\ s \subseteq fmla$   
**shows**  $set (getFrN\ xs\ ts\ \varphi\ n) \subseteq var$   
 <proof>

**lemma** *getFrN\_var*:  
**assumes**  $set\ xs \subseteq var\ set\ ts \subseteq trm$  **and**  $set\ \varphi\ s \subseteq fmla$  **and**  $t \in set\ ts$   
**shows**  $set (getFrN\ xs\ ts\ \varphi\ n) \cap set\ xs = \{\}$   
 <proof>

**lemma** *getFrN\_FvarsT*:  
**assumes**  $set\ xs \subseteq var\ set\ ts \subseteq trm$  **and**  $set\ \varphi\ s \subseteq fmla$  **and**  $t \in set\ ts$   
**shows**  $set (getFrN\ xs\ ts\ \varphi\ n) \cap FvarsT\ t = \{\}$   
 <proof>

**lemma** *getFrN\_Fvars*:  
**assumes**  $set\ xs \subseteq var\ set\ ts \subseteq trm$  **and**  $set\ \varphi\ s \subseteq fmla$  **and**  $\varphi \in set\ \varphi\ s$   
**shows**  $set (getFrN\ xs\ ts\ \varphi\ n) \cap Fvars\ \varphi = \{\}$   
 <proof>

**lemma** *getFrN\_length*:  
**assumes**  $set\ xs \subseteq var\ set\ ts \subseteq trm$  **and**  $set\ \varphi\ s \subseteq fmla$   
**shows**  $length (getFrN\ xs\ ts\ \varphi\ n) = n$   
 <proof>

**lemma** *getFrN\_distinct[simp,intro]*:  
**assumes**  $set\ xs \subseteq var\ set\ ts \subseteq trm$  **and**  $set\ \varphi\ s \subseteq fmla$   
**shows**  $distinct (getFrN\ xs\ ts\ \varphi\ n)$   
 <proof>

### 2.1.3 Parallel Term Substitution

**fun** *rawpsubstT* ::  $'trm \Rightarrow ('trm \times 'var)\ list \Rightarrow 'trm$  **where**  
 $rawpsubstT\ t\ [] = t$   
 $| rawpsubstT\ t\ ((t1,x1) \# txs) = rawpsubstT (substT\ t\ t1\ x1)\ txs$

**lemma** *rawpsubstT[simp]*:  
**assumes**  $t \in trm$  **and**  $snd\ ' (set\ txs) \subseteq var$  **and**  $fst\ ' (set\ txs) \subseteq trm$   
**shows**  $rawpsubstT\ t\ txs \in trm$   
 <proof>

**definition** *psubstT* ::  $'trm \Rightarrow ('trm \times 'var)\ list \Rightarrow 'trm$  **where**  
 $psubstT\ t\ txs =$   
 $(let\ xs = map\ snd\ txs; ts = map\ fst\ txs; us = getFrN\ xs\ (t \# ts) [] (length\ xs)\ in$   
 $rawpsubstT (rawpsubstT\ t (zip (map\ Var\ us)\ xs)) (zip\ ts\ us))$

The psubstT versions of the subst axioms.

**lemma** *psubstT[simp,intro]*:  
**assumes**  $t \in trm$  **and**  $snd\ ' (set\ txs) \subseteq var$  **and**  $fst\ ' (set\ txs) \subseteq trm$   
**shows**  $psubstT\ t\ txs \in trm$   
 <proof>

**lemma** *rawpsubstT\_Var\_not[simp]*:  
**assumes**  $x \in \text{var } \text{snd} \text{ ' (set } \text{txs}) \subseteq \text{var } \text{fst} \text{ ' (set } \text{txs}) \subseteq \text{trm}$   
**and**  $x \notin \text{snd} \text{ ' (set } \text{txs})$   
**shows**  $\text{rawpsubstT } (\text{Var } x) \text{ txs} = \text{Var } x$   
 $\langle \text{proof} \rangle$

**lemma** *psubstT\_Var\_not[simp]*:  
**assumes**  $x \in \text{var } \text{snd} \text{ ' (set } \text{txs}) \subseteq \text{var } \text{fst} \text{ ' (set } \text{txs}) \subseteq \text{trm}$   
**and**  $x \notin \text{snd} \text{ ' (set } \text{txs})$   
**shows**  $\text{psubstT } (\text{Var } x) \text{ txs} = \text{Var } x$   
 $\langle \text{proof} \rangle$

**lemma** *rawpsubstT\_notIn[simp]*:  
**assumes**  $x \in \text{var } \text{snd} \text{ ' (set } \text{txs}) \subseteq \text{var } \text{fst} \text{ ' (set } \text{txs}) \subseteq \text{trm } t \in \text{trm}$   
**and**  $\text{FvarsT } t \cap \text{snd} \text{ ' (set } \text{txs}) = \{\}$   
**shows**  $\text{rawpsubstT } t \text{ txs} = t$   
 $\langle \text{proof} \rangle$

**lemma** *psubstT\_notIn[simp]*:  
**assumes**  $x \in \text{var } \text{snd} \text{ ' (set } \text{txs}) \subseteq \text{var } \text{fst} \text{ ' (set } \text{txs}) \subseteq \text{trm } t \in \text{trm}$   
**and**  $\text{FvarsT } t \cap \text{snd} \text{ ' (set } \text{txs}) = \{\}$   
**shows**  $\text{psubstT } t \text{ txs} = t$   
 $\langle \text{proof} \rangle$

## 2.1.4 Parallel Formula Substitution

**fun** *rawpsubst* ::  $\text{'fmla} \Rightarrow (\text{'trm} \times \text{'var}) \text{ list} \Rightarrow \text{'fmla}$  **where**  
 $\text{rawpsubst } \varphi [] = \varphi$   
 $|\ \text{rawpsubst } \varphi ((t1, x1) \# \text{txs}) = \text{rawpsubst } (\text{subst } \varphi \ t1 \ x1) \ \text{txs}$

**lemma** *rawpsubst[simp]*:  
**assumes**  $\varphi \in \text{fmla}$  **and**  $\text{snd} \text{ ' (set } \text{txs}) \subseteq \text{var}$  **and**  $\text{fst} \text{ ' (set } \text{txs}) \subseteq \text{trm}$   
**shows**  $\text{rawpsubst } \varphi \ \text{txs} \in \text{fmla}$   
 $\langle \text{proof} \rangle$

**definition** *psubst* ::  $\text{'fmla} \Rightarrow (\text{'trm} \times \text{'var}) \text{ list} \Rightarrow \text{'fmla}$  **where**  
 $\text{psubst } \varphi \ \text{txs} =$   
 $(\text{let } \text{xs} = \text{map } \text{snd} \ \text{txs}; \text{ts} = \text{map } \text{fst} \ \text{txs}; \text{us} = \text{getFrN } \text{xs} \ \text{ts} \ [\varphi] \ (\text{length } \text{xs}) \ \text{in}$   
 $\text{rawpsubst } (\text{rawpsubst } \varphi \ (\text{zip } (\text{map } \text{Var } \ \text{us}) \ \text{xs})) \ (\text{zip } \ \text{ts} \ \text{us}))$

The psubst versions of the subst axioms.

**lemma** *psubst[simp,intro]*:  
**assumes**  $\varphi \in \text{fmla}$  **and**  $\text{snd} \text{ ' (set } \text{txs}) \subseteq \text{var}$  **and**  $\text{fst} \text{ ' (set } \text{txs}) \subseteq \text{trm}$   
**shows**  $\text{psubst } \varphi \ \text{txs} \in \text{fmla}$   
 $\langle \text{proof} \rangle$

**lemma** *Fvars\_rawpsubst\_su*:  
**assumes**  $\varphi \in \text{fmla}$  **and**  $\text{snd} \text{ ' (set } \text{txs}) \subseteq \text{var}$  **and**  $\text{fst} \text{ ' (set } \text{txs}) \subseteq \text{trm}$   
**shows**  $\text{Fvars } (\text{rawpsubst } \varphi \ \text{txs}) \subseteq$   
 $(\text{Fvars } \varphi - \text{snd} \text{ ' (set } \text{txs})) \cup (\bigcup \{ \text{FvarsT } t \mid t \ x . (t, x) \in \text{set } \text{txs} \})$   
 $\langle \text{proof} \rangle$

**lemma** *in\_Fvars\_rawpsubst\_imp*:  
**assumes**  $y \in \text{Fvars } (\text{rawpsubst } \varphi \ \text{txs})$   
**and**  $\varphi \in \text{fmla}$  **and**  $\text{snd} \text{ ' (set } \text{txs}) \subseteq \text{var}$  **and**  $\text{fst} \text{ ' (set } \text{txs}) \subseteq \text{trm}$   
**shows**  $(y \in \text{Fvars } \varphi - \text{snd} \text{ ' (set } \text{txs})) \vee$   
 $(y \in \bigcup \{ \text{FvarsT } t \mid t \ x . (t, x) \in \text{set } \text{txs} \})$

*<proof>*

**lemma** *Fvars\_rawsubst*:

**assumes**  $\varphi \in \text{fmla}$  **and**  $\text{snd} \text{ ' (set txs) } \subseteq \text{var}$  **and**  $\text{fst} \text{ ' (set txs) } \subseteq \text{trm}$   
**and**  $\text{distinct} (\text{map} \text{ snd txs})$  **and**  $\forall x \in \text{snd} \text{ ' (set txs)}. \forall t \in \text{fst} \text{ ' (set txs)}. x \notin \text{FvarsT } t$   
**shows**  $\text{Fvars} (\text{rawsubst } \varphi \text{ txs}) =$   
 $(\text{Fvars } \varphi - \text{snd} \text{ ' (set txs)}) \cup$   
 $(\bigcup \{ \text{if } x \in \text{Fvars } \varphi \text{ then } \text{FvarsT } t \text{ else } \{ \} \mid t \text{ x. } (t,x) \in \text{set txs} \})$   
*<proof>*

**lemma** *in\_Fvars\_rawsubstD*:

**assumes**  $y \in \text{Fvars} (\text{rawsubst } \varphi \text{ txs})$   
**and**  $\varphi \in \text{fmla}$  **and**  $\text{snd} \text{ ' (set txs) } \subseteq \text{var}$  **and**  $\text{fst} \text{ ' (set txs) } \subseteq \text{trm}$   
**and**  $\text{distinct} (\text{map} \text{ snd txs})$  **and**  $\forall x \in \text{snd} \text{ ' (set txs)}. \forall t \in \text{fst} \text{ ' (set txs)}. x \notin \text{FvarsT } t$   
**shows**  $(y \in \text{Fvars } \varphi - \text{snd} \text{ ' (set txs)}) \vee$   
 $(y \in \bigcup \{ \text{if } x \in \text{Fvars } \varphi \text{ then } \text{FvarsT } t \text{ else } \{ \} \mid t \text{ x. } (t,x) \in \text{set txs} \})$   
*<proof>*

**lemma** *Fvars\_psubst*:

**assumes**  $\varphi \in \text{fmla}$  **and**  $\text{snd} \text{ ' (set txs) } \subseteq \text{var}$  **and**  $\text{fst} \text{ ' (set txs) } \subseteq \text{trm}$   
**and**  $\text{distinct} (\text{map} \text{ snd txs})$   
**shows**  $\text{Fvars} (\text{psubst } \varphi \text{ txs}) =$   
 $(\text{Fvars } \varphi - \text{snd} \text{ ' (set txs)}) \cup$   
 $(\bigcup \{ \text{if } x \in \text{Fvars } \varphi \text{ then } \text{FvarsT } t \text{ else } \{ \} \mid t \text{ x. } (t,x) \in \text{set txs} \})$   
*<proof>*

**lemma** *in\_Fvars\_psubstD*:

**assumes**  $y \in \text{Fvars} (\text{psubst } \varphi \text{ txs})$   
**and**  $\varphi \in \text{fmla}$  **and**  $\text{snd} \text{ ' (set txs) } \subseteq \text{var}$  **and**  $\text{fst} \text{ ' (set txs) } \subseteq \text{trm}$   
**and**  $\text{distinct} (\text{map} \text{ snd txs})$   
**shows**  $y \in (\text{Fvars } \varphi - \text{snd} \text{ ' (set txs)}) \cup$   
 $(\bigcup \{ \text{if } x \in \text{Fvars } \varphi \text{ then } \text{FvarsT } t \text{ else } \{ \} \mid t \text{ x. } (t,x) \in \text{set txs} \})$   
*<proof>*

**lemma** *subst2\_fresh\_switch*:

**assumes**  $\varphi \in \text{fmla}$   $t \in \text{trm}$   $s \in \text{trm}$   $x \in \text{var}$   $y \in \text{var}$   
**and**  $x \neq y$   $x \notin \text{FvarsT } s$   $y \notin \text{FvarsT } t$   
**shows**  $\text{subst} (\text{subst } \varphi \text{ s } y) \text{ t } x = \text{subst} (\text{subst } \varphi \text{ t } x) \text{ s } y$  (**is** ?L = ?R)  
*<proof>*

**lemma** *rawsubst2\_fresh\_switch*:

**assumes**  $\varphi \in \text{fmla}$   $t \in \text{trm}$   $s \in \text{trm}$   $x \in \text{var}$   $y \in \text{var}$   
**and**  $x \neq y$   $x \notin \text{FvarsT } s$   $y \notin \text{FvarsT } t$   
**shows**  $\text{rawsubst } \varphi ((s,y),(t,x)) = \text{rawsubst } \varphi ((t,x),(s,y))$   
*<proof>*

**lemma** *rawsubst\_compose*:

**assumes**  $\varphi \in \text{fmla}$  **and**  $\text{snd} \text{ ' (set txs1) } \subseteq \text{var}$  **and**  $\text{fst} \text{ ' (set txs1) } \subseteq \text{trm}$   
**and**  $\text{snd} \text{ ' (set txs2) } \subseteq \text{var}$  **and**  $\text{fst} \text{ ' (set txs2) } \subseteq \text{trm}$   
**shows**  $\text{rawsubst} (\text{rawsubst } \varphi \text{ txs1}) \text{ txs2} = \text{rawsubst } \varphi (\text{txs1} @ \text{txs2})$   
*<proof>*

**lemma** *rawsubst\_subst\_fresh\_switch*:

**assumes**  $\varphi \in \text{fmla}$   $\text{snd} \text{ ' (set txs) } \subseteq \text{var}$  **and**  $\text{fst} \text{ ' (set txs) } \subseteq \text{trm}$   
**and**  $\forall x \in \text{snd} \text{ ' (set txs)}. x \notin \text{FvarsT } s$   
**and**  $\forall t \in \text{fst} \text{ ' (set txs)}. y \notin \text{FvarsT } t$   
**and**  $\text{distinct} (\text{map} \text{ snd txs})$   
**and**  $s \in \text{trm}$  **and**  $y \in \text{var}$   $y \notin \text{snd} \text{ ' (set txs)}$

**shows**  $\text{rawpsubst } (\text{subst } \varphi \ s \ y) \ txs = \text{rawpsubst } \varphi \ (txs \ @ \ [(s,y)])$   
 ⟨proof⟩

**lemma** *subst\_rawpsubst\_fresh\_switch*:

**assumes**  $\varphi \in \text{fmla}$   $\text{snd } '(\text{set } txs) \subseteq \text{var}$  **and**  $\text{fst } '(\text{set } txs) \subseteq \text{trm}$   
**and**  $\forall x \in \text{snd } '(\text{set } txs). x \notin \text{FvarsT } s$   
**and**  $\forall t \in \text{fst } '(\text{set } txs). y \notin \text{FvarsT } t$   
**and**  $\text{distinct } (\text{map } \text{snd } txs)$   
**and**  $s \in \text{trm}$  **and**  $y \in \text{var}$   $y \notin \text{snd } '(\text{set } txs)$   
**shows**  $\text{subst } (\text{rawpsubst } \varphi \ txs) \ s \ y = \text{rawpsubst } \varphi \ ((s,y) \# txs)$   
 ⟨proof⟩

**lemma** *rawpsubst\_compose\_freshVar*:

**assumes**  $\varphi \in \text{fmla}$   $\text{snd } '(\text{set } txs) \subseteq \text{var}$  **and**  $\text{fst } '(\text{set } txs) \subseteq \text{trm}$   
**and**  $\text{distinct } (\text{map } \text{snd } txs)$   
**and**  $\bigwedge i \ j. i < j \implies j < \text{length } txs \implies \text{snd } (txs!j) \notin \text{FvarsT } (\text{fst } (txs!i))$   
**and**  $us\_facts: \text{set } us \subseteq \text{var}$   
 $\text{set } us \cap \text{Fvars } \varphi = \{\}$   
 $\text{set } us \cap \bigcup (\text{FvarsT } '(\text{fst } '(\text{set } txs))) = \{\}$   
 $\text{set } us \cap \text{snd } '(\text{set } txs) = \{\}$   
 $\text{length } us = \text{length } txs$   
 $\text{distinct } us$   
**shows**  $\text{rawpsubst } (\text{rawpsubst } \varphi \ (\text{zip } (\text{map } \text{Var } us) (\text{map } \text{snd } txs))) \ (\text{zip } (\text{map } \text{fst } txs) \ us) = \text{rawpsubst } \varphi$   
 $txs$   
 ⟨proof⟩

**lemma** *rawpsubst\_compose\_freshVar2\_aux*:

**assumes**  $\varphi[\text{simp}]: \varphi \in \text{fmla}$   
**and**  $ts: \text{set } ts \subseteq \text{trm}$   
**and**  $xs: \text{set } xs \subseteq \text{var}$   $\text{distinct } xs$   
**and**  $us\_facts: \text{set } us \subseteq \text{var}$   $\text{distinct } us$   
 $\text{set } us \cap \text{Fvars } \varphi = \{\}$   
 $\text{set } us \cap \bigcup (\text{FvarsT } '(\text{set } ts)) = \{\}$   
 $\text{set } us \cap \text{set } xs = \{\}$   
**and**  $vs\_facts: \text{set } vs \subseteq \text{var}$   $\text{distinct } vs$   
 $\text{set } vs \cap \text{Fvars } \varphi = \{\}$   
 $\text{set } vs \cap \bigcup (\text{FvarsT } '(\text{set } ts)) = \{\}$   
 $\text{set } vs \cap \text{set } xs = \{\}$   
**and**  $l: \text{length } us = \text{length } xs$   $\text{length } vs = \text{length } xs$   $\text{length } ts = \text{length } xs$   
**and**  $d: \text{set } us \cap \text{set } vs = \{\}$   
**shows**  $\text{rawpsubst } (\text{rawpsubst } \varphi \ (\text{zip } (\text{map } \text{Var } us) \ xs)) \ (\text{zip } ts \ us) =$   
 $\text{rawpsubst } (\text{rawpsubst } \varphi \ (\text{zip } (\text{map } \text{Var } vs) \ xs)) \ (\text{zip } ts \ vs)$   
 ⟨proof⟩

... now getting rid of the disjointness hypothesis:

**lemma** *rawpsubst\_compose\_freshVar2*:

**assumes**  $\varphi[\text{simp}]: \varphi \in \text{fmla}$   
**and**  $ts: \text{set } ts \subseteq \text{trm}$   
**and**  $xs: \text{set } xs \subseteq \text{var}$   $\text{distinct } xs$   
**and**  $us\_facts: \text{set } us \subseteq \text{var}$   $\text{distinct } us$   
 $\text{set } us \cap \text{Fvars } \varphi = \{\}$   
 $\text{set } us \cap \bigcup (\text{FvarsT } '(\text{set } ts)) = \{\}$   
 $\text{set } us \cap \text{set } xs = \{\}$   
**and**  $vs\_facts: \text{set } vs \subseteq \text{var}$   $\text{distinct } vs$   
 $\text{set } vs \cap \text{Fvars } \varphi = \{\}$   
 $\text{set } vs \cap \bigcup (\text{FvarsT } '(\text{set } ts)) = \{\}$   
 $\text{set } vs \cap \text{set } xs = \{\}$   
**and**  $l: \text{length } us = \text{length } xs$   $\text{length } vs = \text{length } xs$   $\text{length } ts = \text{length } xs$

**shows**  $\text{rawpsubst} (\text{rawpsubst } \varphi (\text{zip} (\text{map } \text{Var } us) xs)) (\text{zip } ts us) =$   
 $\text{rawpsubst} (\text{rawpsubst } \varphi (\text{zip} (\text{map } \text{Var } vs) xs)) (\text{zip } ts vs)$  (**is**  $?L = ?R$ )  
 ⟨proof⟩

**lemma**  $\text{psubst\_subst\_fresh\_switch}$ :  
**assumes**  $\varphi \in \text{fmla}$   $\text{snd } ' \text{ set } txs \subseteq \text{var}$   $\text{fst } ' \text{ set } txs \subseteq \text{trm}$   
**and**  $\forall x \in \text{snd } ' \text{ set } txs. x \notin \text{FvarsT } s$   $\forall t \in \text{fst } ' \text{ set } txs. y \notin \text{FvarsT } t$   
**and**  $\text{distinct} (\text{map } \text{snd } txs)$   
**and**  $s \in \text{trm}$   $y \in \text{var}$   $y \notin \text{snd } ' \text{ set } txs$   
**shows**  $\text{psubst} (\text{subst } \varphi s y) txs = \text{subst} (\text{psubst } \varphi txs) s y$   
 ⟨proof⟩

For many cases, the simpler  $\text{rawpsubst}$  can replace  $\text{psubst}$ :

**lemma**  $\text{psubst\_eq\_rawpsubst}$ :  
**assumes**  $\varphi \in \text{fmla}$   $\text{snd } ' (\text{set } txs) \subseteq \text{var}$  **and**  $\text{fst } ' (\text{set } txs) \subseteq \text{trm}$   
**and**  $\text{distinct} (\text{map } \text{snd } txs)$   
**and**  $\bigwedge i j. i < j \implies j < \text{length } txs \implies \text{snd} (txs!j) \notin \text{FvarsT} (\text{fst} (txs!i))$   
**shows**  $\text{psubst } \varphi txs = \text{rawpsubst } \varphi txs$   
 ⟨proof⟩

Some particular cases:

**lemma**  $\text{psubst\_eq\_subst}$ :  
**assumes**  $\varphi \in \text{fmla}$   $x \in \text{var}$  **and**  $t \in \text{trm}$   
**shows**  $\text{psubst } \varphi [(t,x)] = \text{subst } \varphi t x$   
 ⟨proof⟩

**lemma**  $\text{psubst\_eq\_rawpsubst2}$ :  
**assumes**  $\varphi \in \text{fmla}$   $x1 \in \text{var}$   $x2 \in \text{var}$   $t1 \in \text{trm}$   $t2 \in \text{trm}$   
**and**  $x1 \neq x2$   $x2 \notin \text{FvarsT } t1$   
**shows**  $\text{psubst } \varphi [(t1,x1),(t2,x2)] = \text{rawpsubst } \varphi [(t1,x1),(t2,x2)]$   
 ⟨proof⟩

**lemma**  $\text{psubst\_eq\_rawpsubst3}$ :  
**assumes**  $\varphi \in \text{fmla}$   $x1 \in \text{var}$   $x2 \in \text{var}$   $x3 \in \text{var}$   $t1 \in \text{trm}$   $t2 \in \text{trm}$   $t3 \in \text{trm}$   
**and**  $x1 \neq x2$   $x1 \neq x3$   $x2 \neq x3$   
 $x2 \notin \text{FvarsT } t1$   $x3 \notin \text{FvarsT } t1$   $x3 \notin \text{FvarsT } t2$   
**shows**  $\text{psubst } \varphi [(t1,x1),(t2,x2),(t3,x3)] = \text{rawpsubst } \varphi [(t1,x1),(t2,x2),(t3,x3)]$   
 ⟨proof⟩

**lemma**  $\text{psubst\_eq\_rawpsubst4}$ :  
**assumes**  $\varphi \in \text{fmla}$   $x1 \in \text{var}$   $x2 \in \text{var}$   $x3 \in \text{var}$   $x4 \in \text{var}$   
 $t1 \in \text{trm}$   $t2 \in \text{trm}$   $t3 \in \text{trm}$   $t4 \in \text{trm}$   
**and**  $x1 \neq x2$   $x1 \neq x3$   $x2 \neq x3$   $x1 \neq x4$   $x2 \neq x4$   $x3 \neq x4$   
 $x2 \notin \text{FvarsT } t1$   $x3 \notin \text{FvarsT } t1$   $x3 \notin \text{FvarsT } t2$   $x4 \notin \text{FvarsT } t1$   $x4 \notin \text{FvarsT } t2$   $x4 \notin \text{FvarsT } t3$   
**shows**  $\text{psubst } \varphi [(t1,x1),(t2,x2),(t3,x3),(t4,x4)] = \text{rawpsubst } \varphi [(t1,x1),(t2,x2),(t3,x3),(t4,x4)]$   
 ⟨proof⟩

**lemma**  $\text{rawpsubst\_same\_Var[simp]}$ :  
**assumes**  $\varphi \in \text{fmla}$   $\text{set } xs \subseteq \text{var}$   
**shows**  $\text{rawpsubst } \varphi (\text{map } (\lambda x. (\text{Var } x,x)) xs) = \varphi$   
 ⟨proof⟩

**lemma**  $\text{psubst\_same\_Var[simp]}$ :  
**assumes**  $\varphi \in \text{fmla}$   $\text{set } xs \subseteq \text{var}$  **and**  $\text{distinct } xs$   
**shows**  $\text{psubst } \varphi (\text{map } (\lambda x. (\text{Var } x,x)) xs) = \varphi$   
 ⟨proof⟩



**lemma** *rawpsubst\_notIn[simp]*:  
**assumes**  $snd \text{ ' (set } txs) \subseteq var \text{ fst ' (set } txs) \subseteq trm \varphi \in fmla$   
**and**  $Fvars \varphi \cap snd \text{ ' (set } txs) = \{\}$   
**shows**  $rawpsubst \varphi txs = \varphi$   
 $\langle proof \rangle$

**lemma** *psubst\_notIn[simp]*:  
**assumes**  $x \in var \text{ snd ' (set } txs) \subseteq var \text{ fst ' (set } txs) \subseteq trm \varphi \in fmla$   
**and**  $Fvars \varphi \cap snd \text{ ' (set } txs) = \{\}$   
**shows**  $psubst \varphi txs = \varphi$   
 $\langle proof \rangle$

**end** — context *Generic\_Syntax*

## 2.2 Adding Numerals to the Generic Syntax

**locale** *Syntax\_with\_Numerals* =  
*Generic\_Syntax* *var trm fmla Var FvarsT substT Fvars subst*  
**for** *var* :: 'var set **and** *trm* :: 'trm set **and** *fmla* :: 'fmla set  
**and** *Var FvarsT substT Fvars subst*  
+  
**fixes**  
— Abstract notion of numerals, as a subset of the ground terms:  
*num* :: 'trm set  
**assumes**  
*numNE*:  $num \neq \{\}$   
**and**  
*num*:  $num \subseteq trm$   
**and**  
*FvarsT\_num[simp, intro!]*:  $\bigwedge n. n \in num \implies FvarsT n = \{\}$   
**begin**

**lemma** *substT\_num1[simp]*:  $t \in trm \implies y \in var \implies n \in num \implies substT n t y = n$   
 $\langle proof \rangle$

**lemma** *in\_num[simp]*:  $n \in num \implies n \in trm$   $\langle proof \rangle$

**lemma** *subst\_comp\_num*:  
**assumes**  $\varphi \in fmla \ x \in var \ y \in var \ n \in num$   
**shows**  $x \neq y \implies subst (subst \varphi (Var x) y) n x = subst (subst \varphi n x) n y$   
 $\langle proof \rangle$

**lemma** *rawpsubstT\_num*:  
**assumes**  $snd \text{ ' (set } txs) \subseteq var \text{ fst ' (set } txs) \subseteq trm \ n \in num$   
**shows**  $rawpsubstT n txs = n$   
 $\langle proof \rangle$

**lemma** *psubstT\_num[simp]*:  
**assumes**  $snd \text{ ' (set } txs) \subseteq var \text{ fst ' (set } txs) \subseteq trm \ n \in num$   
**shows**  $psubstT n txs = n$   
 $\langle proof \rangle$

**end** — context *Syntax\_with\_Numerals*

## 2.3 Adding Connectives and Quantifiers

**locale** *Syntax\_with\_Connectives* =

*Generic\_Syntax* var trm fmla Var FvarsT substT Fvars subst  
**for**  
var :: 'var set **and** trm :: 'trm set **and** fmla :: 'fmla set  
**and** Var FvarsT substT Fvars subst  
+  
**fixes**  
— Logical connectives  
eql :: 'trm  $\Rightarrow$  'trm  $\Rightarrow$  'fmla  
**and**  
cnj :: 'fmla  $\Rightarrow$  'fmla  $\Rightarrow$  'fmla  
**and**  
imp :: 'fmla  $\Rightarrow$  'fmla  $\Rightarrow$  'fmla  
**and**  
all :: 'var  $\Rightarrow$  'fmla  $\Rightarrow$  'fmla  
**and**  
exi :: 'var  $\Rightarrow$  'fmla  $\Rightarrow$  'fmla  
**assumes**  
eql[simp,intro]:  $\bigwedge t1 t2. t1 \in trm \Longrightarrow t2 \in trm \Longrightarrow eql\ t1\ t2 \in fmla$   
**and**  
cnj[simp,intro]:  $\bigwedge \varphi1\ \varphi2. \varphi1 \in fmla \Longrightarrow \varphi2 \in fmla \Longrightarrow cnj\ \varphi1\ \varphi2 \in fmla$   
**and**  
imp[simp,intro]:  $\bigwedge \varphi1\ \varphi2. \varphi1 \in fmla \Longrightarrow \varphi2 \in fmla \Longrightarrow imp\ \varphi1\ \varphi2 \in fmla$   
**and**  
all[simp,intro]:  $\bigwedge x\ \varphi. x \in var \Longrightarrow \varphi \in fmla \Longrightarrow all\ x\ \varphi \in fmla$   
**and**  
exi[simp,intro]:  $\bigwedge x\ \varphi. x \in var \Longrightarrow \varphi \in fmla \Longrightarrow exi\ x\ \varphi \in fmla$   
**and**  
Fvars\_eql[simp]:  
 $\bigwedge t1\ t2. t1 \in trm \Longrightarrow t2 \in trm \Longrightarrow Fvars\ (eql\ t1\ t2) = FvarsT\ t1 \cup FvarsT\ t2$   
**and**  
Fvars\_cnj[simp]:  
 $\bigwedge \varphi\ \chi. \varphi \in fmla \Longrightarrow \chi \in fmla \Longrightarrow Fvars\ (cnj\ \varphi\ \chi) = Fvars\ \varphi \cup Fvars\ \chi$   
**and**  
Fvars\_imp[simp]:  
 $\bigwedge \varphi\ \chi. \varphi \in fmla \Longrightarrow \chi \in fmla \Longrightarrow Fvars\ (imp\ \varphi\ \chi) = Fvars\ \varphi \cup Fvars\ \chi$   
**and**  
Fvars\_all[simp]:  
 $\bigwedge x\ \varphi. x \in var \Longrightarrow \varphi \in fmla \Longrightarrow Fvars\ (all\ x\ \varphi) = Fvars\ \varphi - \{x\}$   
**and**  
Fvars\_exi[simp]:  
 $\bigwedge x\ \varphi. x \in var \Longrightarrow \varphi \in fmla \Longrightarrow Fvars\ (exi\ x\ \varphi) = Fvars\ \varphi - \{x\}$   
**and**  
— Assumed properties of substitution  
subst\_cnj[simp]:  
 $\bigwedge x\ \varphi\ \chi\ t. \varphi \in fmla \Longrightarrow \chi \in fmla \Longrightarrow t \in trm \Longrightarrow x \in var \Longrightarrow$   
 $subst\ (cnj\ \varphi\ \chi)\ t\ x = cnj\ (subst\ \varphi\ t\ x)\ (subst\ \chi\ t\ x)$   
**and**  
subst\_imp[simp]:  
 $\bigwedge x\ \varphi\ \chi\ t. \varphi \in fmla \Longrightarrow \chi \in fmla \Longrightarrow t \in trm \Longrightarrow x \in var \Longrightarrow$   
 $subst\ (imp\ \varphi\ \chi)\ t\ x = imp\ (subst\ \varphi\ t\ x)\ (subst\ \chi\ t\ x)$   
**and**  
subst\_all[simp]:  
 $\bigwedge x\ y\ \varphi\ t. \varphi \in fmla \Longrightarrow t \in trm \Longrightarrow x \in var \Longrightarrow y \in var \Longrightarrow$   
 $x \neq y \Longrightarrow x \notin FvarsT\ t \Longrightarrow subst\ (all\ x\ \varphi)\ t\ y = all\ x\ (subst\ \varphi\ t\ y)$   
**and**  
subst\_exi[simp]:  
 $\bigwedge x\ y\ \varphi\ t. \varphi \in fmla \Longrightarrow t \in trm \Longrightarrow x \in var \Longrightarrow y \in var \Longrightarrow$   
 $x \neq y \Longrightarrow x \notin FvarsT\ t \Longrightarrow subst\ (exi\ x\ \varphi)\ t\ y = exi\ x\ (subst\ \varphi\ t\ y)$

**and**  
*subst\_eql[simp]*:  
 $\bigwedge t1\ t2\ t\ x. t \in trm \implies t1 \in trm \implies t2 \in trm \implies x \in var \implies$   
 $subst\ (eq\ t1\ t2)\ t\ x = eq\ (substT\ t1\ t\ x)\ (substT\ t2\ t\ x)$

**begin**

Formula equivalence,  $\longleftrightarrow$ , a derived connective

**definition** *eqv* :: 'fmla  $\Rightarrow$  'fmla  $\Rightarrow$  'fmla **where**  
 $eqv\ \varphi\ \chi = cnj\ (imp\ \varphi\ \chi)\ (imp\ \chi\ \varphi)$

**lemma**

*eqv[simp]*:  $\bigwedge \varphi\ \chi. \varphi \in fmla \implies \chi \in fmla \implies eqv\ \varphi\ \chi \in fmla$

**and**

*Fvars\_eqv[simp]*:  $\bigwedge \varphi\ \chi. \varphi \in fmla \implies \chi \in fmla \implies$   
 $Fvars\ (eqv\ \varphi\ \chi) = Fvars\ \varphi \cup Fvars\ \chi$

**and**

*subst\_eqv[simp]*:  
 $\bigwedge \varphi\ \chi\ t\ x. \varphi \in fmla \implies \chi \in fmla \implies t \in trm \implies x \in var \implies$   
 $subst\ (eqv\ \varphi\ \chi)\ t\ x = eqv\ (subst\ \varphi\ t\ x)\ (subst\ \chi\ t\ x)$   
 <proof>

**lemma** *subst\_all\_idle[simp]*:

**assumes** [simp]:  $x \in var\ \varphi \in fmla\ t \in trm$

**shows**  $subst\ (all\ x\ \varphi)\ t\ x = all\ x\ \varphi$

<proof>

**lemma** *subst\_exi\_idle[simp]*:

**assumes** [simp]:  $x \in var\ \varphi \in fmla\ t \in trm$

**shows**  $subst\ (exi\ x\ \varphi)\ t\ x = exi\ x\ \varphi$

<proof>

Parallel substitution versus connectives and quantifiers.

**lemma** *rawpsubst\_cnj*:

**assumes**  $\varphi1 \in fmla\ \varphi2 \in fmla$

**and**  $snd\ ' (set\ txs) \subseteq var\ fst\ ' (set\ txs) \subseteq trm$

**shows**  $rawpsubst\ (cnj\ \varphi1\ \varphi2)\ txs = cnj\ (rawpsubst\ \varphi1\ txs)\ (rawpsubst\ \varphi2\ txs)$

<proof>

**lemma** *psubst\_cnj[simp]*:

**assumes**  $\varphi1 \in fmla\ \varphi2 \in fmla$

**and**  $snd\ ' (set\ txs) \subseteq var\ fst\ ' (set\ txs) \subseteq trm$

**and** *distinct* (map snd txs)

**shows**  $psubst\ (cnj\ \varphi1\ \varphi2)\ txs = cnj\ (psubst\ \varphi1\ txs)\ (psubst\ \varphi2\ txs)$

<proof>

**lemma** *rawpsubst\_imp*:

**assumes**  $\varphi1 \in fmla\ \varphi2 \in fmla$

**and**  $snd\ ' (set\ txs) \subseteq var\ fst\ ' (set\ txs) \subseteq trm$

**shows**  $rawpsubst\ (imp\ \varphi1\ \varphi2)\ txs = imp\ (rawpsubst\ \varphi1\ txs)\ (rawpsubst\ \varphi2\ txs)$

<proof>

**lemma** *psubst\_imp[simp]*:

**assumes**  $\varphi1 \in fmla\ \varphi2 \in fmla$

**and**  $snd\ ' (set\ txs) \subseteq var\ fst\ ' (set\ txs) \subseteq trm$

**and** *distinct* (map snd txs)

**shows**  $psubst\ (imp\ \varphi1\ \varphi2)\ txs = imp\ (psubst\ \varphi1\ txs)\ (psubst\ \varphi2\ txs)$

<proof>

**lemma** *rawpsubst\_eqv*:  
**assumes**  $\varphi 1 \in \text{fmla}$   $\varphi 2 \in \text{fmla}$   
**and**  $\text{snd} \text{ ' (set txs) } \subseteq \text{var fst} \text{ ' (set txs) } \subseteq \text{trm}$   
**shows**  $\text{rawpsubst (eqv } \varphi 1 \varphi 2) \text{ txs} = \text{eqv (rawpsubst } \varphi 1 \text{ txs) (rawpsubst } \varphi 2 \text{ txs)}$   
 $\langle \text{proof} \rangle$

**lemma** *psubst\_eqv[simp]*:  
**assumes**  $\varphi 1 \in \text{fmla}$   $\varphi 2 \in \text{fmla}$   
**and**  $\text{snd} \text{ ' (set txs) } \subseteq \text{var fst} \text{ ' (set txs) } \subseteq \text{trm}$   
**and**  $\text{distinct (map snd txs)}$   
**shows**  $\text{psubst (eqv } \varphi 1 \varphi 2) \text{ txs} = \text{eqv (psubst } \varphi 1 \text{ txs) (psubst } \varphi 2 \text{ txs)}$   
 $\langle \text{proof} \rangle$

**lemma** *rawpsubst\_all*:  
**assumes**  $\varphi \in \text{fmla}$   $y \in \text{var}$   
**and**  $\text{snd} \text{ ' (set txs) } \subseteq \text{var fst} \text{ ' (set txs) } \subseteq \text{trm}$   
**and**  $y \notin \text{snd} \text{ ' (set txs) } \cup (FvarsT \text{ ' fst} \text{ ' (set txs))}$   
**shows**  $\text{rawpsubst (all } y \varphi) \text{ txs} = \text{all } y \text{ (rawpsubst } \varphi \text{ txs)}$   
 $\langle \text{proof} \rangle$

**lemma** *psubst\_all[simp]*:  
**assumes**  $\varphi \in \text{fmla}$   $y \in \text{var}$   
**and**  $\text{snd} \text{ ' (set txs) } \subseteq \text{var fst} \text{ ' (set txs) } \subseteq \text{trm}$   
**and**  $y \notin \text{snd} \text{ ' (set txs) } \cup (FvarsT \text{ ' fst} \text{ ' (set txs))}$   
**and**  $\text{distinct (map snd txs)}$   
**shows**  $\text{psubst (all } y \varphi) \text{ txs} = \text{all } y \text{ (psubst } \varphi \text{ txs)}$   
 $\langle \text{proof} \rangle$

**lemma** *rawpsubst\_exi*:  
**assumes**  $\varphi \in \text{fmla}$   $y \in \text{var}$   
**and**  $\text{snd} \text{ ' (set txs) } \subseteq \text{var fst} \text{ ' (set txs) } \subseteq \text{trm}$   
**and**  $y \notin \text{snd} \text{ ' (set txs) } \cup (FvarsT \text{ ' fst} \text{ ' (set txs))}$   
**shows**  $\text{rawpsubst (exi } y \varphi) \text{ txs} = \text{exi } y \text{ (rawpsubst } \varphi \text{ txs)}$   
 $\langle \text{proof} \rangle$

**lemma** *psubst\_exi[simp]*:  
**assumes**  $\varphi \in \text{fmla}$   $y \in \text{var}$   
**and**  $\text{snd} \text{ ' (set txs) } \subseteq \text{var fst} \text{ ' (set txs) } \subseteq \text{trm}$   
**and**  $y \notin \text{snd} \text{ ' (set txs) } \cup (FvarsT \text{ ' fst} \text{ ' (set txs))}$   
**and**  $\text{distinct (map snd txs)}$   
**shows**  $\text{psubst (exi } y \varphi) \text{ txs} = \text{exi } y \text{ (psubst } \varphi \text{ txs)}$   
 $\langle \text{proof} \rangle$

**end** — context *Syntax\_with\_Connectives*

**locale** *Syntax\_with\_Numerals\_and\_Connectives* =  
*Syntax\_with\_Numerals*  
*var trm fmla Var FvarsT substT Fvars subst*  
*num*  
+  
*Syntax\_with\_Connectives*  
*var trm fmla Var FvarsT substT Fvars subst*  
*eqI cnj imp all exi*  
**for**  
*var* :: 'var set **and** *trm* :: 'trm set **and** *fmla* :: 'fmla set  
**and** *Var FvarsT substT Fvars subst*  
**and** *num*

**and** *eql cnj imp all exi*  
**begin**

**lemma** *subst\_all\_num[simp]*:  
**assumes**  $\varphi \in \text{fmla } x \in \text{var } y \in \text{var } n \in \text{num}$   
**shows**  $x \neq y \implies \text{subst } (\text{all } x \varphi) n y = \text{all } x (\text{subst } \varphi n y)$   
*<proof>*

**lemma** *subst\_exi\_num[simp]*:  
**assumes**  $\varphi \in \text{fmla } x \in \text{var } y \in \text{var } n \in \text{num}$   
**shows**  $x \neq y \implies \text{subst } (\text{exi } x \varphi) n y = \text{exi } x (\text{subst } \varphi n y)$   
*<proof>*

The "soft substitution" function:

**definition** *softSubst* ::  $'\text{fmla} \Rightarrow '\text{trm} \Rightarrow '\text{var} \Rightarrow '\text{fmla}$  **where**  
*softSubst*  $\varphi t x = \text{exi } x (\text{cnj } (\text{eql } (\text{Var } x) t) \varphi)$

**lemma** *softSubst[simp,intro]*:  $\varphi \in \text{fmla} \implies t \in \text{trm} \implies x \in \text{var} \implies \text{softSubst } \varphi t x \in \text{fmla}$   
*<proof>*

**lemma** *Fvars\_softSubst[simp]*:  
 $\varphi \in \text{fmla} \implies t \in \text{trm} \implies x \in \text{var} \implies$   
 $\text{Fvars } (\text{softSubst } \varphi t x) = (\text{Fvars } \varphi \cup \text{FvarsT } t - \{x\})$   
*<proof>*

**lemma** *Fvars\_softSubst\_subst\_in*:  
 $\varphi \in \text{fmla} \implies t \in \text{trm} \implies x \in \text{var} \implies x \notin \text{FvarsT } t \implies x \in \text{Fvars } \varphi \implies$   
 $\text{Fvars } (\text{softSubst } \varphi t x) = \text{Fvars } (\text{subst } \varphi t x)$   
*<proof>*

**lemma** *Fvars\_softSubst\_subst\_notIn*:  
 $\varphi \in \text{fmla} \implies t \in \text{trm} \implies x \in \text{var} \implies x \notin \text{FvarsT } t \implies x \notin \text{Fvars } \varphi \implies$   
 $\text{Fvars } (\text{softSubst } \varphi t x) = \text{Fvars } (\text{subst } \varphi t x) \cup \text{FvarsT } t$   
*<proof>*

**end** — context *Syntax\_with\_Connectives*

The addition of False among logical connectives

**locale** *Syntax\_with\_Connectives\_False* =  
*Syntax\_with\_Connectives*  
*var trm fmla Var FvarsT substT Fvars subst*  
*eql cnj imp all exi*  
**for**  
*var* ::  $'\text{var set}$  **and** *trm* ::  $'\text{trm set}$  **and** *fmla* ::  $'\text{fmla set}$   
**and** *Var FvarsT substT Fvars subst*  
**and** *eql cnj imp all exi*  
 +  
**fixes** *fls*:: $'\text{fmla}$   
**assumes**  
*fls[simp,intro!]*:  $\text{fls} \in \text{fmla}$   
**and**  
*Fvars\_fl[simp,intro!]*:  $\text{Fvars } \text{fls} = \{\}$   
**and**  
*subst\_fl[simp]*:  
 $\bigwedge t x. t \in \text{trm} \implies x \in \text{var} \implies \text{subst } \text{fls } t x = \text{fls}$   
**begin**

Negation as a derived connective:

**definition**  $neg :: 'fmla \Rightarrow 'fmla$  **where**  
 $neg \varphi = imp \varphi fls$

**lemma**

$neg[simp]: \bigwedge \varphi. \varphi \in fmla \Longrightarrow neg \varphi \in fmla$   
**and**  
 $Fvars\_neg[simp]: \bigwedge \varphi. \varphi \in fmla \Longrightarrow Fvars (neg \varphi) = Fvars \varphi$   
**and**  
 $subst\_neg[simp]:$   
 $\bigwedge \varphi t x. \varphi \in fmla \Longrightarrow t \in trm \Longrightarrow x \in var \Longrightarrow$   
 $subst (neg \varphi) t x = neg (subst \varphi t x)$   
 $\langle proof \rangle$

True as a derived connective:

**definition**  $tru$  **where**  $tru = neg fls$

**lemma**

$tru[simp,intro!]: tru \in fmla$   
**and**  
 $Fvars\_tru[simp]: Fvars tru = \{\}$   
**and**  
 $subst\_tru[simp]: \bigwedge t x. t \in trm \Longrightarrow x \in var \Longrightarrow subst tru t x = tru$   
 $\langle proof \rangle$

### 2.3.1 Iterated conjunction

First we define list-based conjunction:

**fun**  $lcnj :: 'fmla list \Rightarrow 'fmla$  **where**  
 $lcnj [] = tru$   
 $| lcnj (\varphi \# \varphi s) = cnj \varphi (lcnj \varphi s)$

**lemma**  $lcnj[simp,intro!]: set \varphi s \subseteq fmla \Longrightarrow lcnj \varphi s \in fmla$   
 $\langle proof \rangle$

**lemma**  $Fvars\_lcnj[simp]:$   
 $set \varphi s \subseteq fmla \Longrightarrow finite F \Longrightarrow Fvars (lcnj \varphi s) = \bigcup (set (map Fvars \varphi s))$   
 $\langle proof \rangle$

**lemma**  $subst\_lcnj[simp]:$   
 $set \varphi s \subseteq fmla \Longrightarrow t \in trm \Longrightarrow x \in var \Longrightarrow$   
 $subst (lcnj \varphi s) t x = lcnj (map (\lambda \varphi. subst \varphi t x) \varphi s)$   
 $\langle proof \rangle$

Then we define (finite-)set-based conjunction:

**definition**  $scnj :: 'fmla set \Rightarrow 'fmla$  **where**  
 $scnj F = lcnj (asList F)$

**lemma**  $scnj[simp,intro!]: F \subseteq fmla \Longrightarrow finite F \Longrightarrow scnj F \in fmla$   
 $\langle proof \rangle$

**lemma**  $Fvars\_scnj[simp]:$   
 $F \subseteq fmla \Longrightarrow finite F \Longrightarrow Fvars (scnj F) = \bigcup (Fvars ` F)$   
 $\langle proof \rangle$

### 2.3.2 Parallel substitution versus the new connectives

**lemma**  $rawpsubst\_fls:$

$snd \text{ ' (set txs) } \subseteq var \implies fst \text{ ' (set txs) } \subseteq trm \implies rawpsubst \text{ fls txs} = fls$   
 ⟨proof⟩

**lemma** *psubst\_fl[simp]*:  
**assumes**  $snd \text{ ' (set txs) } \subseteq var$  **and**  $fst \text{ ' (set txs) } \subseteq trm$   
**shows**  $psubst \text{ fls txs} = fls$   
 ⟨proof⟩

**lemma** *psubst\_neg[simp]*:  
**assumes**  $\varphi \in fmla$   
**and**  $snd \text{ ' (set txs) } \subseteq var$   $fst \text{ ' (set txs) } \subseteq trm$   
**and**  $distinct \text{ (map snd txs)}$   
**shows**  $psubst \text{ (neg } \varphi) \text{ txs} = neg \text{ (psubst } \varphi \text{ txs)}$   
 ⟨proof⟩

**lemma** *psubst\_tru[simp]*:  
**assumes**  $snd \text{ ' (set txs) } \subseteq var$  **and**  $fst \text{ ' (set txs) } \subseteq trm$   
**and**  $distinct \text{ (map snd txs)}$   
**shows**  $psubst \text{ tru txs} = tru$   
 ⟨proof⟩

**lemma** *psubst\_lcnj[simp]*:  
 $set \varphi s \subseteq fmla \implies snd \text{ ' (set txs) } \subseteq var \implies fst \text{ ' (set txs) } \subseteq trm \implies$   
 $distinct \text{ (map snd txs)} \implies$   
 $psubst \text{ (lcnj } \varphi s) \text{ txs} = lcnj \text{ (map } (\lambda \varphi. psubst \varphi \text{ txs)} \varphi s)$   
 ⟨proof⟩

**end** — context *Syntax\_with\_Connectives\_False*

## 2.4 Adding Disjunction

NB: In intuitionistic logic, disjunction is not definable from the other connectives.

**locale** *Syntax\_with\_Connectives\_False\_Disj* =  
*Syntax\_with\_Connectives\_False*  
 $var \text{ trm fmla Var FvarsT substT Fvars subst}$   
 $eql \text{ cnj imp all exi}$   
 $fls$   
**for**  
 $var :: \text{'var set and trm :: 'trm set and fmla :: 'fmla set}$   
**and**  $Var \text{ FvarsT substT Fvars subst}$   
**and**  $eql \text{ cnj imp all exi}$   
**and**  $fls$   
 +  
**fixes**  $dsj :: \text{'fmla } \Rightarrow \text{'fmla } \Rightarrow \text{'fmla}$   
**assumes**  
 $dsj[simp]: \bigwedge \varphi \chi. \varphi \in fmla \implies \chi \in fmla \implies dsj \varphi \chi \in fmla$   
**and**  
 $Fvars\_dsj[simp]: \bigwedge \varphi \chi. \varphi \in fmla \implies \chi \in fmla \implies$   
 $Fvars \text{ (dsj } \varphi \chi) = Fvars \varphi \cup Fvars \chi$   
**and**  
 $subst\_dsj[simp]:$   
 $\bigwedge x \varphi \chi t. \varphi \in fmla \implies \chi \in fmla \implies t \in trm \implies x \in var \implies$   
 $subst \text{ (dsj } \varphi \chi) \text{ t } x = dsj \text{ (subst } \varphi \text{ t } x) \text{ (subst } \chi \text{ t } x)$   
**begin**

### 2.4.1 Iterated disjunction

First we define list-based disjunction:

```
fun ldsj :: 'fmla list  $\Rightarrow$  'fmla where
  ldsj [] = fls
| ldsj ( $\varphi \# \varphi s$ ) = dsj  $\varphi$  (ldsj  $\varphi s$ )
```

```
lemma ldsj[simp,intro]: set  $\varphi s \subseteq fmla \implies ldsj \varphi s \in fmla$ 
  <proof>
```

```
lemma Fvars_ldsj[simp]:
  set  $\varphi s \subseteq fmla \implies Fvars (ldsj \varphi s) = \bigcup (set (map Fvars \varphi s))$ 
  <proof>
```

```
lemma subst_ldsj[simp]:
  set  $\varphi s \subseteq fmla \implies t \in trm \implies x \in var \implies$ 
  subst (ldsj  $\varphi s$ ) t x = ldsj (map ( $\lambda\varphi. subst \varphi t x$ )  $\varphi s$ )
  <proof>
```

Then we define (finite-)set-based disjunction:

```
definition sdsj :: 'fmla set  $\Rightarrow$  'fmla where
  sdsj F = ldsj (asList F)
```

```
lemma sdsj[simp,intro]: F  $\subseteq fmla \implies finite F \implies sdsj F \in fmla$ 
  <proof>
```

```
lemma Fvars_sdsj[simp]:
  F  $\subseteq fmla \implies finite F \implies Fvars (sdsj F) = \bigcup (Fvars ` F)$ 
  <proof>
```

### 2.4.2 Parallel substitution versus the new connectives

```
lemma rawpsubst_dsj:
  assumes  $\varphi 1 \in fmla \varphi 2 \in fmla$ 
  and  $snd ` (set txs) \subseteq var fst ` (set txs) \subseteq trm$ 
  shows rawpsubst (dsj  $\varphi 1 \varphi 2$ ) txs = dsj (rawpsubst  $\varphi 1 txs$ ) (rawpsubst  $\varphi 2 txs$ )
  <proof>
```

```
lemma psubst_dsj[simp]:
  assumes  $\varphi 1 \in fmla \varphi 2 \in fmla$ 
  and  $snd ` (set txs) \subseteq var fst ` (set txs) \subseteq trm$ 
  and distinct (map snd txs)
  shows psubst (dsj  $\varphi 1 \varphi 2$ ) txs = dsj (psubst  $\varphi 1 txs$ ) (psubst  $\varphi 2 txs$ )
  <proof>
```

```
lemma psubst_ldsj[simp]:
  set  $\varphi s \subseteq fmla \implies snd ` (set txs) \subseteq var \implies fst ` (set txs) \subseteq trm \implies$ 
  distinct (map snd txs)  $\implies$ 
  psubst (ldsj  $\varphi s$ ) txs = ldsj (map ( $\lambda\varphi. psubst \varphi txs$ )  $\varphi s$ )
  <proof>
```

**end** — context *Syntax\_with\_Connectives\_False\_Disj*

## 2.5 Adding an Ordering-Like Formula

```
locale Syntax_with_Numerals_and_Connectives_False_Disj =
  Syntax_with_Connectives_False_Disj
```



```

var trm fmla Var FvarsT substT Fvars subst
eql cnj imp all exi
fls
dsj
+
Syntax_with_Numerals_and_Connectives
var trm fmla Var FvarsT substT Fvars subst
num
eql cnj imp all exi
for
  var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
  and Var FvarsT substT Fvars subst
  and eql cnj imp all exi
  and fls
  and dsj
  and num

```

... and in addition a formula expressing order (think: less than or equal to)

```

locale Syntax_PseudoOrder =
  Syntax_with_Numerals_and_Connectives_False_Disj
  var trm fmla Var FvarsT substT Fvars subst
  eql cnj imp all exi
  fls
  dsj
  num
  for
    var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
    and Var FvarsT substT Fvars subst
    and eql cnj imp all exi
    and fls
    and dsj
    and num
  +
  fixes
    — Lq is a formula with free variables xx yy:
    Lq :: 'fmla
  assumes
    Lq[simp,intro!]: Lq ∈ fmla
  and
    Fvars_Lq[simp]: Fvars Lq = {zz,yy}
begin

definition LLq where LLq t1 t2 = psubst Lq [(t1,zz), (t2,yy)]

lemma LLq_def2: t1 ∈ trm ⇒ t2 ∈ trm ⇒ yy ∉ FvarsT t1 ⇒
  LLq t1 t2 = subst (subst Lq t1 zz) t2 yy
  ⟨proof⟩

lemma LLq[simp,intro]:
  assumes t1 ∈ trm t2 ∈ trm
  shows LLq t1 t2 ∈ fmla
  ⟨proof⟩

lemma LLq2[simp,intro!]:
  n ∈ num ⇒ LLq n (Var yy') ∈ fmla
  ⟨proof⟩

lemma Fvars_LLq[simp]: t1 ∈ trm ⇒ t2 ∈ trm ⇒ yy ∉ FvarsT t1 ⇒

```

$Fvars (LLq\ t1\ t2) = FvarsT\ t1 \cup FvarsT\ t2$   
 ⟨proof⟩

**lemma**  $LLq\_simps[simp]$ :

$m \in num \implies n \in num \implies subst\ (LLq\ m\ (Var\ yy))\ n\ yy = LLq\ m\ n$   
 $m \in num \implies n \in num \implies subst\ (LLq\ m\ (Var\ yy'))\ n\ yy = LLq\ m\ (Var\ yy')$   
 $m \in num \implies subst\ (LLq\ m\ (Var\ yy'))\ (Var\ yy)\ yy' = LLq\ m\ (Var\ yy)$   
 $n \in num \implies subst\ (LLq\ (Var\ xx)\ (Var\ yy))\ n\ xx = LLq\ n\ (Var\ yy)$   
 $n \in num \implies subst\ (LLq\ (Var\ zz)\ (Var\ yy))\ n\ yy = LLq\ (Var\ zz)\ n$   
 $m \in num \implies subst\ (LLq\ (Var\ zz)\ (Var\ yy))\ m\ zz = LLq\ m\ (Var\ yy)$   
 $m \in num \implies n \in num \implies subst\ (LLq\ (Var\ zz)\ n)\ m\ xx = LLq\ (Var\ zz)\ n$   
 ⟨proof⟩

**end** — context  $Syntax\_PseudoOrder$

## 2.6 Allowing the Renaming of Quantified Variables

So far, we did not need any renaming axiom for the quantifiers. However, our axioms for substitution implicitly assume the irrelevance of the bound names; in other words, their usual instances would have this property; and since this assumption greatly simplifies the formal development, we make it at this point.

**locale**  $Syntax\_with\_Connectives\_Rename =$   
 $Syntax\_with\_Connectives$   
*var trm fmla Var FvarsT substT Fvars subst*  
*eql cnj imp all exi*

**for**

*var* :: 'var set **and** *trm* :: 'trm set **and** *fmla* :: 'fmla set

**and** *Var FvarsT substT Fvars subst*

**and** *eql cnj imp all exi*

+

**assumes** *all\_rename*:

$\bigwedge \varphi\ x\ y. \varphi \in fmla \implies x \in var \implies y \in var \implies y \notin Fvars\ \varphi \implies$   
 $all\ x\ \varphi = all\ y\ (subst\ \varphi\ (Var\ y)\ x)$

**and** *exi\_rename*:

$\bigwedge \varphi\ x\ y. \varphi \in fmla \implies x \in var \implies y \in var \implies y \notin Fvars\ \varphi \implies$   
 $exi\ x\ \varphi = exi\ y\ (subst\ \varphi\ (Var\ y)\ x)$

**begin**

**lemma** *all\_rename2*:

$\varphi \in fmla \implies x \in var \implies y \in var \implies (y = x \vee y \notin Fvars\ \varphi) \implies$   
 $all\ x\ \varphi = all\ y\ (subst\ \varphi\ (Var\ y)\ x)$

⟨proof⟩

**lemma** *exi\_rename2*:

$\varphi \in fmla \implies x \in var \implies y \in var \implies (y = x \vee y \notin Fvars\ \varphi) \implies$   
 $exi\ x\ \varphi = exi\ y\ (subst\ \varphi\ (Var\ y)\ x)$

⟨proof⟩

## 2.7 The Exists-Unique Quantifier

It is phrased in such a way as to avoid substitution:

**definition** *exu* :: 'var  $\Rightarrow$  'fmla  $\Rightarrow$  'fmla **where**

*exu*  $x\ \varphi \equiv let\ y = getFr\ [x]\ []\ [\varphi]\ in$

*cnj\ (exi\ x\ \varphi)\ (exi\ y\ (all\ x\ (imp\ \varphi\ (eql\ (Var\ x)\ (Var\ y))))))*

**lemma** *exu[simp,intro]*:

$x \in \text{var} \implies \varphi \in \text{fmla} \implies \text{exu } x \varphi \in \text{fmla}$   
(*proof*)

**lemma** *Fvars\_exu[simp]*:

$x \in \text{var} \implies \varphi \in \text{fmla} \implies \text{Fvars } (\text{exu } x \varphi) = \text{Fvars } \varphi - \{x\}$   
(*proof*)

**lemma** *exu\_def\_var*:

**assumes** [*simp*]:  $x \in \text{var } y \in \text{var } y \neq x \ y \notin \text{Fvars } \varphi \ \varphi \in \text{fmla}$   
**shows**

$\text{exu } x \varphi = \text{cnj } (\text{exi } x \varphi) (\text{exi } y (\text{all } x (\text{imp } \varphi (\text{eql } (\text{Var } x) (\text{Var } y))))))$   
(*proof*)

**lemma** *subst\_exu[simp]*:

**assumes** [*simp*]:  $\varphi \in \text{fmla } t \in \text{trm } x \in \text{var } y \in \text{var } x \neq y \ x \notin \text{FvarsT } t$   
**shows**  $\text{subst } (\text{exu } x \varphi) t y = \text{exu } x (\text{subst } \varphi t y)$   
(*proof*)

**lemma** *subst\_exu\_idle[simp]*:

**assumes** [*simp*]:  $x \in \text{var } \varphi \in \text{fmla } t \in \text{trm}$   
**shows**  $\text{subst } (\text{exu } x \varphi) t x = \text{exu } x \varphi$   
(*proof*)

**lemma** *exu\_rename*:

**assumes** [*simp*]:  $\varphi \in \text{fmla } x \in \text{var } y \in \text{var } y \notin \text{Fvars } \varphi$   
**shows**  $\text{exu } x \varphi = \text{exu } y (\text{subst } \varphi (\text{Var } y) x)$   
(*proof*)

**lemma** *exu\_rename2*:

$\varphi \in \text{fmla} \implies x \in \text{var} \implies y \in \text{var} \implies (y = x \vee y \notin \text{Fvars } \varphi) \implies$   
 $\text{exu } x \varphi = \text{exu } y (\text{subst } \varphi (\text{Var } y) x)$   
(*proof*)

**end** — context *Syntax\_with\_Connectives\_Rename*

# Chapter 3

## Deduction

We formalize deduction in a logical system that (shallowly) embeds intuitionistic logic connectives and quantifiers over a signature containing the numerals.

### 3.1 Positive Logic Deduction

**locale** *Deduct* =

*Syntax\_with\_Numerals\_and\_Connectives*

*var trm fmla Var FvarsT substT Fvars subst*

*num*

*eqL cnj imp all exi*

**for**

*var* :: '*var set* **and** *trm* :: '*trm set* **and** *fmla* :: '*fmla set*

**and** *Var FvarsT substT Fvars subst*

**and** *num*

**and** *eqL cnj imp all exi*

+

**fixes**

— Provability of numeric formulas:

*prv* :: '*fmla*  $\Rightarrow$  *bool*

— Hilbert-style system for intuitionistic logic over  $\rightarrow, \wedge, \forall, \exists$ . ( $\perp$ ,  $\neg$  and  $\vee$  will be included later.) Hilbert-style is preferred since it requires the least amount of infrastructure. (Later, natural deduction rules will also be defined.)

**assumes**

— Propositional rules and axioms. There is a single propositional rule, modus ponens.

— The modus ponens rule:

*prv\_imp\_mp*:

$\bigwedge \varphi \chi. \varphi \in \text{fmla} \Rightarrow \chi \in \text{fmla} \Rightarrow$

$\text{prv} (\text{imp } \varphi \chi) \Rightarrow \text{prv } \varphi \Rightarrow \text{prv } \chi$

**and**

— The propositional intuitionistic axioms:

*prv\_imp\_imp\_triv*:

$\bigwedge \varphi \chi. \varphi \in \text{fmla} \Rightarrow \chi \in \text{fmla} \Rightarrow$

$\text{prv} (\text{imp } \varphi (\text{imp } \chi \varphi))$

**and**

*prv\_imp\_trans*:

$\bigwedge \varphi \chi \psi. \varphi \in \text{fmla} \Rightarrow \chi \in \text{fmla} \Rightarrow \psi \in \text{fmla} \Rightarrow$

$\text{prv} (\text{imp} (\text{imp } \varphi (\text{imp } \chi \psi))$

$(\text{imp} (\text{imp } \varphi \chi) (\text{imp } \varphi \psi)))$

**and**

*prv\_imp\_cnjL*:

$\bigwedge \varphi \chi. \varphi \in \text{fmla} \Rightarrow \chi \in \text{fmla} \Rightarrow$

```

    prv (imp (cnj  $\varphi$   $\chi$ )  $\varphi$ )
and
prv_imp_cnjR:
 $\bigwedge \varphi \chi. \varphi \in \text{fmla} \implies \chi \in \text{fmla} \implies$ 
    prv (imp (cnj  $\varphi$   $\chi$ )  $\chi$ )
and
prv_imp_cnjI:
 $\bigwedge \varphi \chi. \varphi \in \text{fmla} \implies \chi \in \text{fmla} \implies$ 
    prv (imp  $\varphi$  (imp  $\chi$  (cnj  $\varphi$   $\chi$ )))
and
— Predicate calculus (quantifier) rules and axioms
— The rules of universal and existential generalization:
prv_all_imp_gen:
 $\bigwedge x \varphi \chi. x \notin \text{Fvars } \varphi \implies \text{prv } (\text{imp } \varphi \chi) \implies \text{prv } (\text{imp } \varphi (\text{all } x \chi))$ 
and
prv_exi_imp_gen:
 $\bigwedge x \varphi \chi. x \in \text{var} \implies \varphi \in \text{fmla} \implies \chi \in \text{fmla} \implies$ 
 $x \notin \text{Fvars } \chi \implies \text{prv } (\text{imp } \varphi \chi) \implies \text{prv } (\text{imp } (\text{exi } x \varphi) \chi)$ 
and
— Two quantifier instantiation axioms:
prv_all_inst:
 $\bigwedge x \varphi t.$ 
 $x \in \text{var} \implies \varphi \in \text{fmla} \implies t \in \text{trm} \implies$ 
    prv (imp (all  $x \varphi$ ) (subst  $\varphi$   $t$   $x$ ))
and
prv_exi_inst:
 $\bigwedge x \varphi t.$ 
 $x \in \text{var} \implies \varphi \in \text{fmla} \implies t \in \text{trm} \implies$ 
    prv (imp (subst  $\varphi$   $t$   $x$ ) (exi  $x \varphi$ ))
and
— The equality axioms:
prv_eql_refl:
 $\bigwedge x. x \in \text{var} \implies$ 
    prv (eql (Var  $x$ ) (Var  $x$ ))
and
prv_eql_subst:
 $\bigwedge \varphi x y.$ 
 $x \in \text{var} \implies y \in \text{var} \implies \varphi \in \text{fmla} \implies$ 
    prv ((imp (eql (Var  $x$ ) (Var  $y$ )))
        (imp  $\varphi$  (subst  $\varphi$  (Var  $y$ )  $x$ )))
begin

```

### 3.1.1 Properties of the propositional fragment

```

lemma prv_imp_triv:
assumes  $\text{phi}: \varphi \in \text{fmla}$  and  $\text{psi}: \psi \in \text{fmla}$ 
shows  $\text{prv } \psi \implies \text{prv } (\text{imp } \varphi \psi)$ 
    <proof>

```

```

lemma prv_imp_refl:
assumes  $\text{phi}: \varphi \in \text{fmla}$ 
shows  $\text{prv } (\text{imp } \varphi \varphi)$ 
    <proof>

```

```

lemma prv_imp_refl2:  $\varphi \in \text{fmla} \implies \psi \in \text{fmla} \implies \varphi = \psi \implies \text{prv } (\text{imp } \varphi \psi)$ 
    <proof>

```

**lemma** *prv\_cnjI*:

**assumes** *phi*:  $\varphi \in \text{fmla}$  **and** *chi*:  $\chi \in \text{fmla}$   
**shows**  $\text{prv } \varphi \implies \text{prv } \chi \implies \text{prv } (\text{cnj } \varphi \ \chi)$   
*<proof>*

**lemma** *prv\_cnjEL*:

**assumes** *phi*:  $\varphi \in \text{fmla}$  **and** *chi*:  $\chi \in \text{fmla}$   
**shows**  $\text{prv } (\text{cnj } \varphi \ \chi) \implies \text{prv } \varphi$   
*<proof>*

**lemma** *prv\_cnjER*:

**assumes** *phi*:  $\varphi \in \text{fmla}$  **and** *chi*:  $\chi \in \text{fmla}$   
**shows**  $\text{prv } (\text{cnj } \varphi \ \chi) \implies \text{prv } \chi$   
*<proof>*

**lemma** *prv\_prv\_imp\_trans*:

**assumes** *phi*:  $\varphi \in \text{fmla}$  **and** *chi*:  $\chi \in \text{fmla}$  **and** *psi*:  $\psi \in \text{fmla}$   
**assumes** *1*:  $\text{prv } (\text{imp } \varphi \ \chi)$  **and** *2*:  $\text{prv } (\text{imp } \chi \ \psi)$   
**shows**  $\text{prv } (\text{imp } \varphi \ \psi)$   
*<proof>*

**lemma** *prv\_imp\_trans1*:

**assumes** *phi*:  $\varphi \in \text{fmla}$  **and** *chi*:  $\chi \in \text{fmla}$  **and** *psi*:  $\psi \in \text{fmla}$   
**shows**  $\text{prv } (\text{imp } (\text{imp } \chi \ \psi) (\text{imp } (\text{imp } \varphi \ \chi) (\text{imp } \varphi \ \psi)))$   
*<proof>*

**lemma** *prv\_imp\_com*:

**assumes** *phi*:  $\varphi \in \text{fmla}$  **and** *chi*:  $\chi \in \text{fmla}$  **and** *psi*:  $\psi \in \text{fmla}$   
**assumes**  $\text{prv } (\text{imp } \varphi (\text{imp } \chi \ \psi))$   
**shows**  $\text{prv } (\text{imp } \chi (\text{imp } \varphi \ \psi))$   
*<proof>*

**lemma** *prv\_imp\_trans2*:

**assumes** *phi*:  $\varphi \in \text{fmla}$  **and** *chi*:  $\chi \in \text{fmla}$  **and** *psi*:  $\psi \in \text{fmla}$   
**shows**  $\text{prv } (\text{imp } (\text{imp } \varphi \ \chi) (\text{imp } (\text{imp } \chi \ \psi) (\text{imp } \varphi \ \psi)))$   
*<proof>*

**lemma** *prv\_imp\_cnj*:

**assumes**  $\varphi \in \text{fmla}$  **and**  $\chi \in \text{fmla}$  **and**  $\psi \in \text{fmla}$   
**shows**  $\text{prv } (\text{imp } \varphi \ \psi) \implies \text{prv } (\text{imp } \varphi \ \chi) \implies \text{prv } (\text{imp } \varphi (\text{cnj } \psi \ \chi))$   
*<proof>*

**lemma** *prv\_imp\_imp\_com*:

**assumes**  $\varphi \in \text{fmla}$  **and**  $\chi \in \text{fmla}$  **and**  $\psi \in \text{fmla}$   
**shows**  
 $\text{prv } (\text{imp } (\text{imp } \varphi (\text{imp } \chi \ \psi))$   
 $\quad (\text{imp } \chi (\text{imp } \varphi \ \psi)))$   
*<proof>*

**lemma** *prv\_cnj\_imp\_monoR2*:

**assumes**  $\varphi \in \text{fmla}$  **and**  $\chi \in \text{fmla}$  **and**  $\psi \in \text{fmla}$   
**assumes**  $\text{prv } (\text{imp } \varphi (\text{imp } \chi \ \psi))$   
**shows**  $\text{prv } (\text{imp } (\text{cnj } \varphi \ \chi) \ \psi)$   
*<proof>*

**lemma** *prv\_imp\_imp\_imp\_cnj*:

**assumes**  $\varphi \in \text{fmla}$  **and**  $\chi \in \text{fmla}$  **and**  $\psi \in \text{fmla}$   
**shows**

*prv* (*imp* (*imp*  $\varphi$  (*imp*  $\chi$   $\psi$ ))  
           (*imp* (*cnj*  $\varphi$   $\chi$ )  $\psi$ ))  
 ⟨*proof*⟩

**lemma** *prv\_imp\_cnj\_imp*:  
**assumes**  $\varphi \in \text{fmla}$  **and**  $\chi \in \text{fmla}$  **and**  $\psi \in \text{fmla}$   
**shows**  
*prv* (*imp* (*imp* (*cnj*  $\varphi$   $\chi$ )  $\psi$ )  
           (*imp*  $\varphi$  (*imp*  $\chi$   $\psi$ )))  
 ⟨*proof*⟩

**lemma** *prv\_cnj\_imp*:  
**assumes**  $\varphi \in \text{fmla}$  **and**  $\chi \in \text{fmla}$  **and**  $\psi \in \text{fmla}$   
**assumes** *prv* (*imp* (*cnj*  $\varphi$   $\chi$ )  $\psi$ )  
**shows** *prv* (*imp*  $\varphi$  (*imp*  $\chi$   $\psi$ ))  
 ⟨*proof*⟩

Monotonicity of conjunction w.r.t. implication:

**lemma** *prv\_cnj\_imp\_monoR*:  
**assumes**  $\varphi \in \text{fmla}$  **and**  $\chi \in \text{fmla}$  **and**  $\psi \in \text{fmla}$   
**shows** *prv* (*imp* (*imp*  $\varphi$   $\chi$ ) (*imp* (*imp*  $\varphi$   $\psi$ ) (*imp*  $\varphi$  (*cnj*  $\chi$   $\psi$ ))))  
 ⟨*proof*⟩

**lemma** *prv\_imp\_cnj3L*:  
**assumes**  $\varphi1 \in \text{fmla}$  **and**  $\varphi2 \in \text{fmla}$  **and**  $\chi \in \text{fmla}$   
**shows** *prv* (*imp* (*imp*  $\varphi1$   $\chi$ ) (*imp* (*cnj*  $\varphi1$   $\varphi2$ )  $\chi$ ))  
 ⟨*proof*⟩

**lemma** *prv\_imp\_cnj3R*:  
**assumes**  $\varphi1 \in \text{fmla}$  **and**  $\varphi2 \in \text{fmla}$  **and**  $\chi \in \text{fmla}$   
**shows** *prv* (*imp* (*imp*  $\varphi2$   $\chi$ ) (*imp* (*cnj*  $\varphi1$   $\varphi2$ )  $\chi$ ))  
 ⟨*proof*⟩

**lemma** *prv\_cnj\_imp\_mono*:  
**assumes**  $\varphi1 \in \text{fmla}$  **and**  $\varphi2 \in \text{fmla}$  **and**  $\chi1 \in \text{fmla}$  **and**  $\chi2 \in \text{fmla}$   
**shows** *prv* (*imp* (*imp*  $\varphi1$   $\chi1$ ) (*imp* (*imp*  $\varphi2$   $\chi2$ ) (*imp* (*cnj*  $\varphi1$   $\varphi2$ ) (*cnj*  $\chi1$   $\chi2$ ))))  
 ⟨*proof*⟩

**lemma** *prv\_cnj\_mono*:  
**assumes**  $\varphi1 \in \text{fmla}$  **and**  $\varphi2 \in \text{fmla}$  **and**  $\chi1 \in \text{fmla}$  **and**  $\chi2 \in \text{fmla}$   
**assumes** *prv* (*imp*  $\varphi1$   $\chi1$ ) **and** *prv* (*imp*  $\varphi2$   $\chi2$ )  
**shows** *prv* (*imp* (*cnj*  $\varphi1$   $\varphi2$ ) (*cnj*  $\chi1$   $\chi2$ ))  
 ⟨*proof*⟩

**lemma** *prv\_cnj\_imp\_monoR4*:  
**assumes**  $\varphi \in \text{fmla}$  **and**  $\chi \in \text{fmla}$  **and**  $\psi1 \in \text{fmla}$  **and**  $\psi2 \in \text{fmla}$   
**shows**  
*prv* (*imp* (*imp*  $\varphi$  (*imp*  $\chi$   $\psi1$ ))  
           (*imp* (*imp*  $\varphi$  (*imp*  $\chi$   $\psi2$ )) (*imp*  $\varphi$  (*imp*  $\chi$  (*cnj*  $\psi1$   $\psi2$ ))))))  
 ⟨*proof*⟩

**lemma** *prv\_imp\_cnj4*:  
**assumes**  $\varphi \in \text{fmla}$  **and**  $\chi \in \text{fmla}$  **and**  $\psi1 \in \text{fmla}$  **and**  $\psi2 \in \text{fmla}$   
**shows** *prv* (*imp*  $\varphi$  (*imp*  $\chi$   $\psi1$ ))  $\implies$  *prv* (*imp*  $\varphi$  (*imp*  $\chi$   $\psi2$ ))  $\implies$  *prv* (*imp*  $\varphi$  (*imp*  $\chi$  (*cnj*  $\psi1$   $\psi2$ )))  
 ⟨*proof*⟩

**lemma** *prv\_cnj\_imp\_monoR5*:  
**assumes**  $\varphi1 \in \text{fmla}$  **and**  $\varphi2 \in \text{fmla}$  **and**  $\chi1 \in \text{fmla}$  **and**  $\chi2 \in \text{fmla}$

**shows**  $prv (imp (imp \varphi 1 \chi 1) (imp (imp \varphi 2 \chi 2) (imp \varphi 1 (imp \varphi 2 (cnj \chi 1 \chi 2))))))$   
(*proof*)

**lemma** *prv\_imp\_cnj5*:

**assumes**  $\varphi 1 \in fmla$  **and**  $\varphi 2 \in fmla$  **and**  $\chi 1 \in fmla$  **and**  $\chi 2 \in fmla$

**assumes**  $prv (imp \varphi 1 \chi 1)$  **and**  $prv (imp \varphi 2 \chi 2)$

**shows**  $prv (imp \varphi 1 (imp \varphi 2 (cnj \chi 1 \chi 2)))$

(*proof*)

Properties of formula equivalence:

**lemma** *prv\_eqv\_imp*:

**assumes**  $\varphi \in fmla$  **and**  $\chi \in fmla$

**shows**  $prv (imp (eqv \varphi \chi) (eqv \chi \varphi))$

(*proof*)

**lemma** *prv\_eqv\_eqv*:

**assumes**  $\varphi \in fmla$  **and**  $\chi \in fmla$

**shows**  $prv (eqv (eqv \varphi \chi) (eqv \chi \varphi))$

(*proof*)

**lemma** *prv\_imp\_eqvEL*:

$\varphi 1 \in fmla \implies \varphi 2 \in fmla \implies prv (eqv \varphi 1 \varphi 2) \implies prv (imp \varphi 1 \varphi 2)$

(*proof*)

**lemma** *prv\_imp\_eqvER*:

$\varphi 1 \in fmla \implies \varphi 2 \in fmla \implies prv (eqv \varphi 1 \varphi 2) \implies prv (imp \varphi 2 \varphi 1)$

(*proof*)

**lemma** *prv\_eqv\_imp\_trans*:

**assumes**  $\varphi \in fmla$  **and**  $\chi \in fmla$  **and**  $\psi \in fmla$

**shows**  $prv (imp (eqv \varphi \chi) (imp (eqv \chi \psi) (eqv \varphi \psi)))$

(*proof*)

**lemma** *prv\_eqv\_cnj\_trans*:

**assumes**  $\varphi \in fmla$  **and**  $\chi \in fmla$  **and**  $\psi \in fmla$

**shows**  $prv (imp (cnj (eqv \varphi \chi) (eqv \chi \psi)) (eqv \varphi \psi))$

(*proof*)

**lemma** *prv\_eqvI*:

**assumes**  $\varphi \in fmla$  **and**  $\chi \in fmla$

**assumes**  $prv (imp \varphi \chi)$  **and**  $prv (imp \chi \varphi)$

**shows**  $prv (eqv \varphi \chi)$

(*proof*)

Formula equivalence is a congruence (i.e., an equivalence that is compatible with the other connectives):

**lemma** *prv\_eqv\_refl*:  $\varphi \in fmla \implies prv (eqv \varphi \varphi)$

(*proof*)

**lemma** *prv\_eqv\_sym*:

**assumes**  $\varphi \in fmla$  **and**  $\chi \in fmla$

**shows**  $prv (eqv \varphi \chi) \implies prv (eqv \chi \varphi)$

(*proof*)

**lemma** *prv\_eqv\_trans*:

**assumes**  $\varphi \in fmla$  **and**  $\chi \in fmla$  **and**  $\psi \in fmla$

**shows**  $prv (eqv \varphi \chi) \implies prv (eqv \chi \psi) \implies prv (eqv \varphi \psi)$

(*proof*)



**lemma** *imp\_imp\_compat\_eqvL*:  
**assumes**  $\varphi 1 \in \text{fm}la$  **and**  $\varphi 2 \in \text{fm}la$  **and**  $\chi \in \text{fm}la$   
**shows** *prv* (*imp* (*eqv*  $\varphi 1$   $\varphi 2$ ) (*eqv* (*imp*  $\varphi 1$   $\chi$ ) (*imp*  $\varphi 2$   $\chi$ )))  
*<proof>*

**lemma** *imp\_imp\_compat\_eqvR*:  
**assumes**  $\varphi \in \text{fm}la$  **and**  $\chi 1 \in \text{fm}la$  **and**  $\chi 2 \in \text{fm}la$   
**shows** *prv* (*imp* (*eqv*  $\chi 1$   $\chi 2$ ) (*eqv* (*imp*  $\varphi$   $\chi 1$ ) (*imp*  $\varphi$   $\chi 2$ )))  
*<proof>*

**lemma** *imp\_imp\_compat\_eqv*:  
**assumes**  $\varphi 1 \in \text{fm}la$  **and**  $\varphi 2 \in \text{fm}la$  **and**  $\chi 1 \in \text{fm}la$  **and**  $\chi 2 \in \text{fm}la$   
**shows** *prv* (*imp* (*eqv*  $\varphi 1$   $\varphi 2$ ) (*imp* (*eqv*  $\chi 1$   $\chi 2$ ) (*eqv* (*imp*  $\varphi 1$   $\chi 1$ ) (*imp*  $\varphi 2$   $\chi 2$ ))))  
*<proof>*

**lemma** *imp\_compat\_eqvL*:  
**assumes**  $\varphi 1 \in \text{fm}la$  **and**  $\varphi 2 \in \text{fm}la$  **and**  $\chi \in \text{fm}la$   
**assumes** *prv* (*eqv*  $\varphi 1$   $\varphi 2$ )  
**shows** *prv* (*eqv* (*imp*  $\varphi 1$   $\chi$ ) (*imp*  $\varphi 2$   $\chi$ ))  
*<proof>*

**lemma** *imp\_compat\_eqvR*:  
**assumes**  $\varphi \in \text{fm}la$  **and**  $\chi 1 \in \text{fm}la$  **and**  $\chi 2 \in \text{fm}la$   
**assumes** *prv* (*eqv*  $\chi 1$   $\chi 2$ )  
**shows** *prv* (*eqv* (*imp*  $\varphi$   $\chi 1$ ) (*imp*  $\varphi$   $\chi 2$ ))  
*<proof>*

**lemma** *imp\_compat\_eqv*:  
**assumes**  $\varphi 1 \in \text{fm}la$  **and**  $\varphi 2 \in \text{fm}la$  **and**  $\chi 1 \in \text{fm}la$  **and**  $\chi 2 \in \text{fm}la$   
**assumes** *prv* (*eqv*  $\varphi 1$   $\varphi 2$ ) **and** *prv* (*eqv*  $\chi 1$   $\chi 2$ )  
**shows** *prv* (*eqv* (*imp*  $\varphi 1$   $\chi 1$ ) (*imp*  $\varphi 2$   $\chi 2$ ))  
*<proof>*

**lemma** *imp\_cnj\_compat\_eqvL*:  
**assumes**  $\varphi 1 \in \text{fm}la$  **and**  $\varphi 2 \in \text{fm}la$  **and**  $\chi \in \text{fm}la$   
**shows** *prv* (*imp* (*eqv*  $\varphi 1$   $\varphi 2$ ) (*eqv* (*cnj*  $\varphi 1$   $\chi$ ) (*cnj*  $\varphi 2$   $\chi$ )))  
*<proof>*

**lemma** *imp\_cnj\_compat\_eqvR*:  
**assumes**  $\varphi \in \text{fm}la$  **and**  $\chi 1 \in \text{fm}la$  **and**  $\chi 2 \in \text{fm}la$   
**shows** *prv* (*imp* (*eqv*  $\chi 1$   $\chi 2$ ) (*eqv* (*cnj*  $\varphi$   $\chi 1$ ) (*cnj*  $\varphi$   $\chi 2$ )))  
*<proof>*

**lemma** *imp\_cnj\_compat\_eqv*:  
**assumes**  $\varphi 1 \in \text{fm}la$  **and**  $\varphi 2 \in \text{fm}la$  **and**  $\chi 1 \in \text{fm}la$  **and**  $\chi 2 \in \text{fm}la$   
**shows** *prv* (*imp* (*eqv*  $\varphi 1$   $\varphi 2$ ) (*imp* (*eqv*  $\chi 1$   $\chi 2$ ) (*eqv* (*cnj*  $\varphi 1$   $\chi 1$ ) (*cnj*  $\varphi 2$   $\chi 2$ ))))  
*<proof>*

**lemma** *cnj\_compat\_eqvL*:  
**assumes**  $\varphi 1 \in \text{fm}la$  **and**  $\varphi 2 \in \text{fm}la$  **and**  $\chi \in \text{fm}la$   
**assumes** *prv* (*eqv*  $\varphi 1$   $\varphi 2$ )  
**shows** *prv* (*eqv* (*cnj*  $\varphi 1$   $\chi$ ) (*cnj*  $\varphi 2$   $\chi$ ))  
*<proof>*

**lemma** *cnj\_compat\_eqvR*:

**assumes**  $\varphi \in \text{fmla}$  **and**  $\chi1 \in \text{fmla}$  **and**  $\chi2 \in \text{fmla}$   
**assumes**  $\text{prv } (\text{eqv } \chi1 \ \chi2)$   
**shows**  $\text{prv } (\text{eqv } (\text{cnj } \varphi \ \chi1) \ (\text{cnj } \varphi \ \chi2))$   
 <proof>

**lemma** *cnj\_compat\_eqv*:  
**assumes**  $\varphi1 \in \text{fmla}$  **and**  $\varphi2 \in \text{fmla}$  **and**  $\chi1 \in \text{fmla}$  **and**  $\chi2 \in \text{fmla}$   
**assumes**  $\text{prv } (\text{eqv } \varphi1 \ \varphi2)$  **and**  $\text{prv } (\text{eqv } \chi1 \ \chi2)$   
**shows**  $\text{prv } (\text{eqv } (\text{cnj } \varphi1 \ \chi1) \ (\text{cnj } \varphi2 \ \chi2))$   
 <proof>

**lemma** *prv\_eqv\_prv*:  
**assumes**  $\varphi \in \text{fmla}$  **and**  $\chi \in \text{fmla}$   
**assumes**  $\text{prv } \varphi$  **and**  $\text{prv } (\text{eqv } \varphi \ \chi)$   
**shows**  $\text{prv } \chi$   
 <proof>

**lemma** *prv\_eqv\_prv\_rev*:  
**assumes**  $\varphi \in \text{fmla}$  **and**  $\chi \in \text{fmla}$   
**assumes**  $\text{prv } \varphi$  **and**  $\text{prv } (\text{eqv } \chi \ \varphi)$   
**shows**  $\text{prv } \chi$   
 <proof>

**lemma** *prv\_imp\_eqv\_transi*:  
**assumes**  $\varphi \in \text{fmla}$  **and**  $\chi1 \in \text{fmla}$  **and**  $\chi2 \in \text{fmla}$   
**assumes**  $\text{prv } (\text{imp } \varphi \ \chi1)$  **and**  $\text{prv } (\text{eqv } \chi1 \ \chi2)$   
**shows**  $\text{prv } (\text{imp } \varphi \ \chi2)$   
 <proof>

**lemma** *prv\_imp\_eqv\_transi\_rev*:  
**assumes**  $\varphi \in \text{fmla}$  **and**  $\chi1 \in \text{fmla}$  **and**  $\chi2 \in \text{fmla}$   
**assumes**  $\text{prv } (\text{imp } \varphi \ \chi2)$  **and**  $\text{prv } (\text{eqv } \chi1 \ \chi2)$   
**shows**  $\text{prv } (\text{imp } \varphi \ \chi1)$   
 <proof>

**lemma** *prv\_eqv\_imp\_transi*:  
**assumes**  $\varphi1 \in \text{fmla}$  **and**  $\varphi2 \in \text{fmla}$  **and**  $\chi \in \text{fmla}$   
**assumes**  $\text{prv } (\text{eqv } \varphi1 \ \varphi2)$  **and**  $\text{prv } (\text{imp } \varphi2 \ \chi)$   
**shows**  $\text{prv } (\text{imp } \varphi1 \ \chi)$   
 <proof>

**lemma** *prv\_eqv\_imp\_transi\_rev*:  
**assumes**  $\varphi1 \in \text{fmla}$  **and**  $\varphi2 \in \text{fmla}$  **and**  $\chi \in \text{fmla}$   
**assumes**  $\text{prv } (\text{eqv } \varphi1 \ \varphi2)$  **and**  $\text{prv } (\text{imp } \varphi1 \ \chi)$   
**shows**  $\text{prv } (\text{imp } \varphi2 \ \chi)$   
 <proof>

**lemma** *prv\_imp\_monoL*:  $\varphi \in \text{fmla} \implies \chi \in \text{fmla} \implies \psi \in \text{fmla} \implies$   
 $\text{prv } (\text{imp } \chi \ \psi) \implies \text{prv } (\text{imp } (\text{imp } \varphi \ \chi) \ (\text{imp } \varphi \ \psi))$   
 <proof>

**lemma** *prv\_imp\_monoR*:  $\varphi \in \text{fmla} \implies \chi \in \text{fmla} \implies \psi \in \text{fmla} \implies$   
 $\text{prv } (\text{imp } \psi \ \chi) \implies \text{prv } (\text{imp } (\text{imp } \chi \ \varphi) \ (\text{imp } \psi \ \varphi))$   
 <proof>

More properties involving conjunction:

**lemma** *prv\_cnj\_com\_imp*:  
**assumes**  $\varphi[\text{simp}]$ :  $\varphi \in \text{fmla}$  **and**  $\chi[\text{simp}]$ :  $\chi \in \text{fmla}$

**shows**  $prv (imp (cnj \varphi \chi) (cnj \chi \varphi))$   
*<proof>*

**lemma** *prv\_cnj\_com*:  
**assumes**  $\varphi[simp]: \varphi \in fmla$  **and**  $\chi[simp]: \chi \in fmla$   
**shows**  $prv (equiv (cnj \varphi \chi) (cnj \chi \varphi))$   
*<proof>*

**lemma** *prv\_cnj\_assoc\_imp1*:  
**assumes**  $\varphi[simp]: \varphi \in fmla$  **and**  $\chi[simp]: \chi \in fmla$  **and**  $\psi[simp]: \psi \in fmla$   
**shows**  $prv (imp (cnj \varphi (cnj \chi \psi)) (cnj (cnj \varphi \chi) \psi))$   
*<proof>*

**lemma** *prv\_cnj\_assoc\_imp2*:  
**assumes**  $\varphi[simp]: \varphi \in fmla$  **and**  $\chi[simp]: \chi \in fmla$  **and**  $\psi[simp]: \psi \in fmla$   
**shows**  $prv (imp (cnj (cnj \varphi \chi) \psi) (cnj \varphi (cnj \chi \psi)))$   
*<proof>*

**lemma** *prv\_cnj\_assoc*:  
**assumes**  $\varphi[simp]: \varphi \in fmla$  **and**  $\chi[simp]: \chi \in fmla$  **and**  $\psi[simp]: \psi \in fmla$   
**shows**  $prv (equiv (cnj \varphi (cnj \chi \psi)) (cnj (cnj \varphi \chi) \psi))$   
*<proof>*

**lemma** *prv\_cnj\_com\_imp3*:  
**assumes**  $\varphi1 \in fmla$   $\varphi2 \in fmla$   $\varphi3 \in fmla$   
**shows**  $prv (imp (cnj \varphi1 (cnj \varphi2 \varphi3))$   
 $(cnj \varphi2 (cnj \varphi1 \varphi3)))$   
*<proof>*

### 3.1.2 Properties involving quantifiers

Fundamental properties:

**lemma** *prv\_allE*:  
**assumes**  $x \in var$  **and**  $\varphi \in fmla$  **and**  $t \in trm$   
**shows**  $prv (all x \varphi) \implies prv (subst \varphi t x)$   
*<proof>*

**lemma** *prv\_exiI*:  
**assumes**  $x \in var$  **and**  $\varphi \in fmla$  **and**  $t \in trm$   
**shows**  $prv (subst \varphi t x) \implies prv (exi x \varphi)$   
*<proof>*

**lemma** *prv\_imp\_imp\_exi*:  
**assumes**  $x \in var$  **and**  $\varphi \in fmla$  **and**  $\chi \in fmla$   
**assumes**  $x \notin Fvars \varphi$   
**shows**  $prv (imp (exi x (imp \varphi \chi)) (imp \varphi (exi x \chi)))$   
*<proof>*

**lemma** *prv\_imp\_exi*:  
**assumes**  $x \in var$  **and**  $\varphi \in fmla$  **and**  $\chi \in fmla$   
**shows**  $x \notin Fvars \varphi \implies prv (exi x (imp \varphi \chi)) \implies prv (imp \varphi (exi x \chi))$   
*<proof>*

**lemma** *prv\_exi\_imp*:  
**assumes**  $x: x \in var$  **and**  $\varphi \in fmla$  **and**  $\chi \in fmla$   
**assumes**  $x \notin Fvars \chi$  **and**  $d: prv (all x (imp \varphi \chi))$   
**shows**  $prv (imp (exi x \varphi) \chi)$   
*<proof>*

**lemma** *prv\_all\_imp*:  
**assumes**  $x \in \text{var}$  **and**  $\varphi \in \text{fmla}$  **and**  $\chi \in \text{fmla}$   
**assumes**  $x \notin \text{Fvars } \varphi$  **and**  $\text{prv } (\text{all } x \ (\text{imp } \varphi \ \chi))$   
**shows**  $\text{prv } (\text{imp } \varphi \ (\text{all } x \ \chi))$   
 $\langle \text{proof} \rangle$

**lemma** *prv\_exi\_inst\_same*:  
**assumes**  $\varphi \in \text{fmla}$   $\chi \in \text{fmla}$   $x \in \text{var}$   
**shows**  $\text{prv } (\text{imp } \varphi \ (\text{exi } x \ \varphi))$   
 $\langle \text{proof} \rangle$

**lemma** *prv\_exi\_cong*:  
**assumes**  $\varphi \in \text{fmla}$   $\chi \in \text{fmla}$   $x \in \text{var}$   
**and**  $\text{prv } (\text{imp } \varphi \ \chi)$   
**shows**  $\text{prv } (\text{imp } (\text{exi } x \ \varphi) \ (\text{exi } x \ \chi))$   
 $\langle \text{proof} \rangle$

**lemma** *prv\_exi\_congW*:  
**assumes**  $\varphi \in \text{fmla}$   $\chi \in \text{fmla}$   $x \in \text{var}$   
**and**  $\text{prv } (\text{imp } \varphi \ \chi)$   $\text{prv } (\text{exi } x \ \varphi)$   
**shows**  $\text{prv } (\text{exi } x \ \chi)$   
 $\langle \text{proof} \rangle$

**lemma** *prv\_all\_cong*:  
**assumes**  $\varphi \in \text{fmla}$   $\chi \in \text{fmla}$   $x \in \text{var}$   
**and**  $\text{prv } (\text{imp } \varphi \ \chi)$   
**shows**  $\text{prv } (\text{imp } (\text{all } x \ \varphi) \ (\text{all } x \ \chi))$   
 $\langle \text{proof} \rangle$

**lemma** *prv\_all\_congW*:  
**assumes**  $\varphi \in \text{fmla}$   $\chi \in \text{fmla}$   $x \in \text{var}$   
**and**  $\text{prv } (\text{imp } \varphi \ \chi)$   $\text{prv } (\text{all } x \ \varphi)$   
**shows**  $\text{prv } (\text{all } x \ \chi)$   
 $\langle \text{proof} \rangle$

Quantifiers versus free variables and substitution:

**lemma** *exists\_no\_Fvars*:  $\exists \varphi. \varphi \in \text{fmla} \wedge \text{prv } \varphi \wedge \text{Fvars } \varphi = \{\}$   
 $\langle \text{proof} \rangle$

**lemma** *prv\_all\_gen*:  
**assumes**  $x \in \text{var}$  **and**  $\varphi \in \text{fmla}$   
**assumes**  $\text{prv } \varphi$  **shows**  $\text{prv } (\text{all } x \ \varphi)$   
 $\langle \text{proof} \rangle$

**lemma** *all\_subst\_rename\_prv*:  
 $\varphi \in \text{fmla} \implies x \in \text{var} \implies y \in \text{var} \implies$   
 $y \notin \text{Fvars } \varphi \implies \text{prv } (\text{all } x \ \varphi) \implies \text{prv } (\text{all } y \ (\text{subst } \varphi \ (\text{Var } y) \ x))$   
 $\langle \text{proof} \rangle$

**lemma** *allE\_id*:  
**assumes**  $y \in \text{var}$  **and**  $\varphi \in \text{fmla}$   
**assumes**  $\text{prv } (\text{all } y \ \varphi)$   
**shows**  $\text{prv } \varphi$   
 $\langle \text{proof} \rangle$

**lemma** *prv\_subst*:  
**assumes**  $x \in \text{var}$  **and**  $\varphi \in \text{fmla}$  **and**  $t \in \text{trm}$

**shows**  $prv \varphi \implies prv (subst \varphi t x)$   
(*proof*)

**lemma** *prv\_rawsubst*:

**assumes**  $\varphi \in fmla$  **and**  $snd \text{ ' (set txs) } \subseteq var$  **and**  $fst \text{ ' (set txs) } \subseteq trm$   
**and**  $prv \varphi$   
**shows**  $prv (rawpsubst \varphi txs)$   
(*proof*)

**lemma** *prv\_psubst*:

**assumes**  $\varphi \in fmla$  **and**  $snd \text{ ' (set txs) } \subseteq var$  **and**  $fst \text{ ' (set txs) } \subseteq trm$   
**and**  $prv \varphi$   
**shows**  $prv (psubst \varphi txs)$   
(*proof*)

**lemma** *prv\_eqv\_rawsubst*:

$\varphi \in fmla \implies \psi \in fmla \implies snd \text{ ' set txs } \subseteq var \implies fst \text{ ' set txs } \subseteq trm \implies prv (eqv \varphi \psi) \implies$   
 $prv (eqv (rawpsubst \varphi txs) (rawpsubst \psi txs))$   
(*proof*)

**lemma** *prv\_eqv\_psubst*:

$\varphi \in fmla \implies \psi \in fmla \implies snd \text{ ' set txs } \subseteq var \implies fst \text{ ' set txs } \subseteq trm \implies prv (eqv \varphi \psi) \implies$   
 $distinct (map snd txs) \implies$   
 $prv (eqv (psubst \varphi txs) (psubst \psi txs))$   
(*proof*)

**lemma** *prv\_all\_imp\_trans*:

**assumes**  $x \in var$  **and**  $\varphi \in fmla$  **and**  $\chi \in fmla$  **and**  $\psi \in fmla$   
**shows**  $prv (all x (imp \varphi \chi)) \implies prv (all x (imp \chi \psi)) \implies prv (all x (imp \varphi \psi))$   
(*proof*)

**lemma** *prv\_all\_imp\_cnj*:

**assumes**  $x \in var$  **and**  $\varphi \in fmla$  **and**  $\chi \in fmla$  **and**  $\psi \in fmla$   
**shows**  $prv (all x (imp \varphi (imp \psi \chi))) \implies prv (all x (imp (cnj \psi \varphi) \chi))$   
(*proof*)

**lemma** *prv\_all\_imp\_cnj\_rev*:

**assumes**  $x \in var$  **and**  $\varphi \in fmla$  **and**  $\chi \in fmla$  **and**  $\psi \in fmla$   
**shows**  $prv (all x (imp (cnj \varphi \psi) \chi)) \implies prv (all x (imp \varphi (imp \psi \chi)))$   
(*proof*)

### 3.1.3 Properties concerning equality

**lemma** *prv\_eql\_reflT*:

**assumes**  $t: t \in trm$   
**shows**  $prv (eql t t)$   
(*proof*)

**lemma** *prv\_eql\_subst\_trm*:

**assumes**  $xx: x \in var$  **and**  $\varphi: \varphi \in fmla$  **and**  $t1 \in trm$  **and**  $t2 \in trm$   
**shows**  $prv ((imp (eql t1 t2)$   
 $(imp (subst \varphi t1 x) (subst \varphi t2 x))))$   
(*proof*)

**lemma** *prv\_eql\_subst\_trm2*:

**assumes**  $x \in var$  **and**  $\varphi \in fmla$  **and**  $t1 \in trm$  **and**  $t2 \in trm$   
**assumes**  $prv (eql t1 t2)$   
**shows**  $prv (imp (subst \varphi t1 x) (subst \varphi t2 x))$

*<proof>*

**lemma** *prv\_eql\_sym:*

**assumes** [*simp*]:  $t1 \in trm$   $t2 \in trm$   
**shows**  $prv (imp (eq1 t1 t2) (eq1 t2 t1))$   
*<proof>*

**lemma** *prv\_prv\_eql\_sym:*  $t1 \in trm \implies t2 \in trm \implies prv (eq1 t1 t2) \implies prv (eq1 t2 t1)$   
*<proof>*

**lemma** *prv\_all\_eq1:*

**assumes**  $x \in var$  **and**  $y \in var$  **and**  $\varphi \in fmla$  **and**  $t1 \in trm$  **and**  $t2 \in trm$   
**shows**  $prv (all x ((imp (eq1 t1 t2) (imp (subst \varphi t2 y) (subst \varphi t1 y))))))$   
*<proof>*

**lemma** *prv\_eq1\_subst\_trm\_rev:*

**assumes**  $t1 \in trm$  **and**  $t2 \in trm$  **and**  $\varphi \in fmla$  **and**  $y \in var$   
**shows**  
 $prv ((imp (eq1 t1 t2) (imp (subst \varphi t2 y) (subst \varphi t1 y))))$   
*<proof>*

**lemma** *prv\_eq1\_subst\_trm\_rev2:*

**assumes**  $x \in var$  **and**  $\varphi \in fmla$  **and**  $t1 \in trm$  **and**  $t2 \in trm$   
**assumes**  $prv (eq1 t1 t2)$   
**shows**  $prv (imp (subst \varphi t2 x) (subst \varphi t1 x))$   
*<proof>*

**lemma** *prv\_eq1\_subst\_trm\_eqv:*

**assumes**  $x \in var$  **and**  $\varphi \in fmla$  **and**  $t1 \in trm$  **and**  $t2 \in trm$   
**assumes**  $prv (eq1 t1 t2)$   
**shows**  $prv (eqv (subst \varphi t1 x) (subst \varphi t2 x))$   
*<proof>*

**lemma** *prv\_eq1\_subst\_trm\_id:*

**assumes**  $y \in var$   $\varphi \in fmla$  **and**  $n \in num$   
**shows**  $prv (imp (eq1 (Var y) n) (imp \varphi (subst \varphi n y)))$   
*<proof>*

**lemma** *prv\_eq1\_subst\_trm\_id\_back:*

**assumes**  $y \in var$   $\varphi \in fmla$  **and**  $n \in num$   
**shows**  $prv (imp (eq1 (Var y) n) (imp (subst \varphi n y) \varphi))$   
*<proof>*

**lemma** *prv\_eq1\_subst\_trm\_id\_rev:*

**assumes**  $y \in var$   $\varphi \in fmla$  **and**  $n \in num$   
**shows**  $prv (imp (eq1 n (Var y)) (imp \varphi (subst \varphi n y)))$   
*<proof>*

**lemma** *prv\_eq1\_subst\_trm\_id\_back\_rev:*

**assumes**  $y \in var$   $\varphi \in fmla$  **and**  $n \in num$   
**shows**  $prv (imp (eq1 n (Var y)) (imp (subst \varphi n y) \varphi))$   
*<proof>*

**lemma** *prv\_eq1\_imp\_trans\_rev:*

**assumes**  $t1[simp]: t1 \in trm$  **and**  $t2[simp]: t2 \in trm$  **and**  $t3[simp]: t3 \in trm$   
**shows**  $prv (imp (eq1 t1 t2) (imp (eq1 t1 t3) (eq1 t2 t3)))$

*<proof>*

**lemma** *prv\_eql\_imp\_trans*:

**assumes**  $t1[simp]: t1 \in trm$  **and**  $t2[simp]: t2 \in trm$  **and**  $t3[simp]: t3 \in trm$   
**shows**  $prv (imp (eq1 t1 t2) (imp (eq1 t2 t3) (eq1 t1 t3)))$   
*<proof>*

**lemma** *prv\_eq1\_trans*:

**assumes**  $t1[simp]: t1 \in trm$  **and**  $t2[simp]: t2 \in trm$  **and**  $t3[simp]: t3 \in trm$   
**and**  $prv (eq1 t1 t2)$  **and**  $prv (eq1 t2 t3)$   
**shows**  $prv (eq1 t1 t3)$   
*<proof>*

### 3.1.4 The equivalence between soft substitution and substitution

**lemma** *prv\_subst\_imp\_softSubst*:

**assumes**  $[simp,intro!]: x \in var \ t \in trm \ \varphi \in fmla \ x \notin FvarsT \ t$   
**shows**  $prv (imp (subst \varphi \ t \ x) (softSubst \varphi \ t \ x))$   
*<proof>*

**lemma** *prv\_subst\_implies\_softSubst*:

**assumes**  $x \in var \ t \in trm \ \varphi \in fmla$   
**and**  $x \notin FvarsT \ t$   
**and**  $prv (subst \varphi \ t \ x)$   
**shows**  $prv (softSubst \varphi \ t \ x)$   
*<proof>*

**lemma** *prv\_softSubst\_imp\_subst*:

**assumes**  $[simp,intro!]: x \in var \ t \in trm \ \varphi \in fmla \ x \notin FvarsT \ t$   
**shows**  $prv (imp (softSubst \varphi \ t \ x) (subst \varphi \ t \ x))$   
*<proof>*

**lemma** *prv\_softSubst\_implies\_subst*:

**assumes**  $x \in var \ t \in trm \ \varphi \in fmla$   
**and**  $x \notin FvarsT \ t$   
**and**  $prv (softSubst \varphi \ t \ x)$   
**shows**  $prv (subst \varphi \ t \ x)$   
*<proof>*

**lemma** *prv\_softSubst\_eqv\_subst*:

**assumes**  $[simp,intro!]: x \in var \ t \in trm \ \varphi \in fmla \ x \notin FvarsT \ t$   
**shows**  $prv (eqv (softSubst \varphi \ t \ x) (subst \varphi \ t \ x))$   
*<proof>*

**end** — context *Deduct*

## 3.2 Deduction Considering False

**locale** *Deduct\_with\_False* =

*Syntax\_with\_Connectives\_False*  
*var trm fmla Var FvarsT substT Fvars subst*  
*eq1 cnj imp all exi*  
*fls*  
*+*  
*Deduct*  
*var trm fmla Var FvarsT substT Fvars subst*  
*num*  
*eq1 cnj imp all exi*

```

prv
for
  var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
  and Var FvarsT substT Fvars subst
  and eql cnj imp all exi
  and fls
  and num
  and prv
+
assumes
  prv_fl[simp,intro]:  $\wedge \varphi. prv (imp fls \varphi)$ 
begin

```

### 3.2.1 Basic properties of False (fls)

**lemma** *prv\_expl*:

**assumes**  $\varphi \in fmla$

**assumes** *prv fls*

**shows** *prv  $\varphi$*

*<proof>*

**lemma** *prv\_cnjR\_fls*:  $\varphi \in fmla \implies prv (equiv (cnj fls \varphi) fls)$

*<proof>*

**lemma** *prv\_cnjL\_fls*:  $\varphi \in fmla \implies prv (equiv (cnj \varphi fls) fls)$

*<proof>*

### 3.2.2 Properties involving negation

Recall that negation has been defined from implication and False.

**lemma** *prv\_imp\_neg\_fls*:

**assumes**  $\varphi \in fmla$

**shows** *prv (imp  $\varphi (imp (neg \varphi) fls)$ )*

*<proof>*

**lemma** *prv\_neg\_fls*:

**assumes**  $\varphi \in fmla$

**assumes** *prv  $\varphi$*  and *prv (neg  $\varphi$ )*

**shows** *prv fls*

*<proof>*

**lemma** *prv\_imp\_neg\_neg*:

**assumes**  $\varphi \in fmla$

**shows** *prv (imp  $\varphi (neg (neg \varphi))$ )*

*<proof>*

**lemma** *prv\_neg\_neg*:

**assumes**  $\varphi \in fmla$

**assumes** *prv  $\varphi$*

**shows** *prv (neg (neg  $\varphi$ ))*

*<proof>*

**lemma** *prv\_imp\_imp\_neg\_rev*:

**assumes**  $\varphi \in fmla$  and  $\chi \in fmla$

**shows** *prv (imp (imp  $\varphi \chi$ ) (imp (neg  $\chi$ ) (neg  $\varphi$ )))*

*<proof>*



**lemma** *prv\_imp\_neg\_rev*:  
**assumes**  $\varphi \in \text{fmla}$  **and**  $\chi \in \text{fmla}$   
**assumes** *prv* (*imp*  $\varphi$   $\chi$ )  
**shows** *prv* (*imp* (*neg*  $\chi$ ) (*neg*  $\varphi$ ))  
*<proof>*

**lemma** *prv\_eqv\_neg\_prv\_flts*:  
 $\varphi \in \text{fmla} \implies$   
*prv* (*eqv*  $\varphi$  (*neg*  $\varphi$ ))  $\implies$  *prv fls*  
*<proof>*

**lemma** *prv\_eqv\_eqv\_neg\_prv\_flts*:  
 $\varphi \in \text{fmla} \implies \chi \in \text{fmla} \implies$   
*prv* (*eqv*  $\varphi$   $\chi$ )  $\implies$  *prv* (*eqv*  $\varphi$  (*neg*  $\chi$ ))  $\implies$  *prv fls*  
*<proof>*

**lemma** *prv\_eqv\_eqv\_neg\_prv\_flts2*:  
 $\varphi \in \text{fmla} \implies \chi \in \text{fmla} \implies$   
*prv* (*eqv*  $\varphi$   $\chi$ )  $\implies$  *prv* (*eqv*  $\chi$  (*neg*  $\varphi$ ))  $\implies$  *prv fls*  
*<proof>*

**lemma** *prv\_neg\_imp\_imp\_trans*:  
**assumes**  $\varphi \in \text{fmla}$   $\chi \in \text{fmla}$   $\psi \in \text{fmla}$   
**and** *prv* (*neg*  $\varphi$ )  
**and** *prv* (*imp*  $\chi$  (*imp*  $\psi$   $\varphi$ ))  
**shows** *prv* (*imp*  $\chi$  (*neg*  $\psi$ ))  
*<proof>*

**lemma** *prv\_imp\_neg\_imp\_neg\_imp*:  
**assumes**  $\varphi \in \text{fmla}$   $\chi \in \text{fmla}$   
**and** *prv* (*neg*  $\varphi$ )  
**shows** *prv* ((*imp*  $\chi$  (*neg* (*imp*  $\chi$   $\varphi$ ))))  
*<proof>*

**lemma** *prv\_prv\_neg\_imp\_neg*:  
**assumes**  $\varphi \in \text{fmla}$   $\chi \in \text{fmla}$   
**and** *prv*  $\varphi$  **and** *prv*  $\chi$   
**shows** *prv* (*neg* (*imp*  $\varphi$  (*neg*  $\chi$ )))  
*<proof>*

**lemma** *prv\_imp\_neg\_imp\_cnjL*:  
**assumes**  $\varphi \in \text{fmla}$   $\varphi1 \in \text{fmla}$   $\varphi2 \in \text{fmla}$   
**and** *prv* (*imp*  $\varphi$  (*neg*  $\varphi1$ ))  
**shows** *prv* (*imp*  $\varphi$  (*neg* (*cnj*  $\varphi1$   $\varphi2$ )))  
*<proof>*

**lemma** *prv\_imp\_neg\_imp\_cnjR*:  
**assumes**  $\varphi \in \text{fmla}$   $\varphi1 \in \text{fmla}$   $\varphi2 \in \text{fmla}$   
**and** *prv* (*imp*  $\varphi$  (*neg*  $\varphi2$ ))  
**shows** *prv* (*imp*  $\varphi$  (*neg* (*cnj*  $\varphi1$   $\varphi2$ )))  
*<proof>*

Negation versus quantifiers:

**lemma** *prv\_all\_neg\_imp\_neg\_exi*:  
**assumes**  $x: x \in \text{var}$  **and**  $\varphi: \varphi \in \text{fmla}$   
**shows** *prv* (*imp* (*all*  $x$  (*neg*  $\varphi$ )) (*neg* (*exi*  $x$   $\varphi$ )))  
*<proof>*

**lemma** *prv\_neg\_exi\_imp\_all\_neg*:  
**assumes**  $x: x \in \text{var}$  **and**  $\varphi: \varphi \in \text{fmla}$   
**shows**  $\text{prv } (\text{imp } (\text{neg } (\text{exi } x \ \varphi)) \ (\text{all } x \ (\text{neg } \varphi)))$   
 $\langle \text{proof} \rangle$

**lemma** *prv\_neg\_exi\_eqv\_all\_neg*:  
**assumes**  $x: x \in \text{var}$  **and**  $\varphi: \varphi \in \text{fmla}$   
**shows**  $\text{prv } (\text{eqv } (\text{neg } (\text{exi } x \ \varphi)) \ (\text{all } x \ (\text{neg } \varphi)))$   
 $\langle \text{proof} \rangle$

**lemma** *prv\_neg\_exi\_implies\_all\_neg*:  
**assumes**  $x: x \in \text{var}$  **and**  $\varphi: \varphi \in \text{fmla}$  **and**  $\text{prv } (\text{neg } (\text{exi } x \ \varphi))$   
**shows**  $\text{prv } (\text{all } x \ (\text{neg } \varphi))$   
 $\langle \text{proof} \rangle$

**lemma** *prv\_neg\_neg\_exi*:  
**assumes**  $x \in \text{var}$   $\varphi \in \text{fmla}$   
**and**  $\text{prv } (\text{neg } \varphi)$   
**shows**  $\text{prv } (\text{neg } (\text{exi } x \ \varphi))$   
 $\langle \text{proof} \rangle$

**lemma** *prv\_exi\_imp\_exiI*:  
**assumes**  $[\text{simp}]: x \in \text{var}$   $y \in \text{var}$   $\varphi \in \text{fmla}$  **and**  $yx: y = x \vee y \notin \text{Fvars } \varphi$   
**shows**  $\text{prv } (\text{imp } (\text{exi } x \ \varphi) \ (\text{exi } y \ (\text{subst } \varphi \ (\text{Var } y) \ x)))$   
 $\langle \text{proof} \rangle$

**lemma** *prv\_imp\_neg\_allI*:  
**assumes**  $\varphi \in \text{fmla}$   $\chi \in \text{fmla}$   $t \in \text{trm}$   $x \in \text{var}$   
**and**  $\text{prv } (\text{imp } \varphi \ (\text{neg } (\text{subst } \chi \ t \ x)))$   
**shows**  $\text{prv } (\text{imp } \varphi \ (\text{neg } (\text{all } x \ \chi)))$   
 $\langle \text{proof} \rangle$

**lemma** *prv\_imp\_neg\_allWI*:  
**assumes**  $\chi \in \text{fmla}$   $t \in \text{trm}$   $x \in \text{var}$   
**and**  $\text{prv } (\text{neg } (\text{subst } \chi \ t \ x))$   
**shows**  $\text{prv } (\text{neg } (\text{all } x \ \chi))$   
 $\langle \text{proof} \rangle$

### 3.2.3 Properties involving True (tru)

**lemma** *prv\_imp\_tru*:  $\varphi \in \text{fmla} \implies \text{prv } (\text{imp } \varphi \ \text{tru})$   
 $\langle \text{proof} \rangle$

**lemma** *prv\_tru\_imp*:  $\varphi \in \text{fmla} \implies \text{prv } (\text{eqv } (\text{imp } \text{tru } \varphi) \ \varphi)$   
 $\langle \text{proof} \rangle$

**lemma** *prv\_fls\_neg\_tru*:  $\varphi \in \text{fmla} \implies \text{prv } (\text{eqv } \text{fls} \ (\text{neg } \text{tru}))$   
 $\langle \text{proof} \rangle$

**lemma** *prv\_tru\_neg\_fls*:  $\varphi \in \text{fmla} \implies \text{prv } (\text{eqv } \text{tru} \ (\text{neg } \text{fls}))$   
 $\langle \text{proof} \rangle$

**lemma** *prv\_cnjR\_tru*:  $\varphi \in \text{fmla} \implies \text{prv } (\text{eqv } (\text{cnj } \text{tru } \varphi) \ \varphi)$   
 $\langle \text{proof} \rangle$

**lemma** *prv\_cnjL\_tru*:  $\varphi \in \text{fmla} \implies \text{prv } (\text{eqv } (\text{cnj } \varphi \ \text{tru}) \ \varphi)$   
 $\langle \text{proof} \rangle$

### 3.2.4 Property of set-based conjunctions

These are based on properties of the auxiliary list conjunctions.

**lemma** *prv\_lcnj\_imp\_in*:  
**assumes**  $set\ \varphi s \subseteq fmla$   
**and**  $\varphi \in set\ \varphi s$   
**shows**  $prv\ (imp\ (lcnj\ \varphi s)\ \varphi)$   
*<proof>*

**lemma** *prv\_lcnj\_imp*:  
**assumes**  $\chi \in fmla$  **and**  $set\ \varphi s \subseteq fmla$   
**and**  $\varphi \in set\ \varphi s$  **and**  $prv\ (imp\ \varphi\ \chi)$   
**shows**  $prv\ (imp\ (lcnj\ \varphi s)\ \chi)$   
*<proof>*

**lemma** *prv\_imp\_lcnj*:  
**assumes**  $\chi \in fmla$  **and**  $set\ \varphi s \subseteq fmla$   
**and**  $\bigwedge \varphi. \varphi \in set\ \varphi s \implies prv\ (imp\ \chi\ \varphi)$   
**shows**  $prv\ (imp\ \chi\ (lcnj\ \varphi s))$   
*<proof>*

**lemma** *prv\_lcnj\_imp\_inner*:  
**assumes**  $\varphi \in fmla$   $set\ \varphi 1s \subseteq fmla$   $set\ \varphi 2s \subseteq fmla$   
**shows**  $prv\ (imp\ (cnj\ \varphi\ (lcnj\ (\varphi 1s\ @\ \varphi 2s)))\ (lcnj\ (\varphi 1s\ @\ \varphi\ \#\ \varphi 2s)))$   
*<proof>*

**lemma** *prv\_lcnj\_imp\_remdups*:  
**assumes**  $set\ \varphi s \subseteq fmla$   
**shows**  $prv\ (imp\ (lcnj\ (remdups\ \varphi s))\ (lcnj\ \varphi s))$   
*<proof>*

**lemma** *prv\_lcnj\_mono*:  
**assumes**  $\varphi 1s: set\ \varphi 1s \subseteq fmla$  **and**  $set\ \varphi 2s \subseteq set\ \varphi 1s$   
**shows**  $prv\ (imp\ (lcnj\ \varphi 1s)\ (lcnj\ \varphi 2s))$   
*<proof>*

**lemma** *prv\_lcnj\_eqv*:  
**assumes**  $set\ \varphi 1s \subseteq fmla$  **and**  $set\ \varphi 2s = set\ \varphi 1s$   
**shows**  $prv\ (eqv\ (lcnj\ \varphi 1s)\ (lcnj\ \varphi 2s))$   
*<proof>*

**lemma** *prv\_lcnj\_mono\_imp*:  
**assumes**  $set\ \varphi 1s \subseteq fmla$   $set\ \varphi 2s \subseteq fmla$  **and**  $\forall\ \varphi 2 \in set\ \varphi 2s. \exists\ \varphi 1 \in set\ \varphi 1s. prv\ (imp\ \varphi 1\ \varphi 2)$   
**shows**  $prv\ (imp\ (lcnj\ \varphi 1s)\ (lcnj\ \varphi 2s))$   
*<proof>*

Set-based conjunction commutes with substitution only up to provably equivalence:

**lemma** *prv\_subst\_scnj*:  
**assumes**  $F \subseteq fmla$  *finite*  $F\ t \in trm\ x \in var$   
**shows**  $prv\ (eqv\ (subst\ (scnj\ F)\ t\ x)\ (scnj\ ((\lambda \varphi. subst\ \varphi\ t\ x)\ 'F)))$   
*<proof>*

**lemma** *prv\_imp\_subst\_scnj*:  
**assumes**  $F \subseteq fmla$  *finite*  $F\ t \in trm\ x \in var$   
**shows**  $prv\ (imp\ (subst\ (scnj\ F)\ t\ x)\ (scnj\ ((\lambda \varphi. subst\ \varphi\ t\ x)\ 'F)))$   
*<proof>*

**lemma** *prv\_subst\_scnj\_imp*:

**assumes**  $F \subseteq \text{fmla}$  *finite*  $F$   $t \in \text{trm}$   $x \in \text{var}$   
**shows**  $\text{prv} (\text{imp} (\text{scnj} ((\lambda\varphi. \text{subst } \varphi \ t \ x) ' F)) (\text{subst} (\text{scnj } F) \ t \ x))$   
 $\langle \text{proof} \rangle$

**lemma** *prv\_scnj\_imp\_in*:  
**assumes**  $F \subseteq \text{fmla}$  *finite*  $F$   
**and**  $\varphi \in F$   
**shows**  $\text{prv} (\text{imp} (\text{scnj } F) \ \varphi)$   
 $\langle \text{proof} \rangle$

**lemma** *prv\_scnj\_imp*:  
**assumes**  $\chi \in \text{fmla}$  **and**  $F \subseteq \text{fmla}$  *finite*  $F$   
**and**  $\varphi \in F$  **and**  $\text{prv} (\text{imp } \varphi \ \chi)$   
**shows**  $\text{prv} (\text{imp} (\text{scnj } F) \ \chi)$   
 $\langle \text{proof} \rangle$

**lemma** *prv\_imp\_scnj*:  
**assumes**  $\chi \in \text{fmla}$  **and**  $F \subseteq \text{fmla}$  *finite*  $F$   
**and**  $\bigwedge \varphi. \varphi \in F \implies \text{prv} (\text{imp } \chi \ \varphi)$   
**shows**  $\text{prv} (\text{imp } \chi (\text{scnj } F))$   
 $\langle \text{proof} \rangle$

**lemma** *prv\_scnj\_mono*:  
**assumes**  $F1 \subseteq \text{fmla}$  **and**  $F2 \subseteq F1$  **and** *finite*  $F1$   
**shows**  $\text{prv} (\text{imp} (\text{scnj } F1) (\text{scnj } F2))$   
 $\langle \text{proof} \rangle$

**lemma** *prv\_scnj\_mono\_imp*:  
**assumes**  $F1 \subseteq \text{fmla}$   $F2 \subseteq \text{fmla}$  *finite*  $F1$  *finite*  $F2$   
**and**  $\forall \varphi2 \in F2. \exists \varphi1 \in F1. \text{prv} (\text{imp } \varphi1 \ \varphi2)$   
**shows**  $\text{prv} (\text{imp} (\text{scnj } F1) (\text{scnj } F2))$   
 $\langle \text{proof} \rangle$

Commutation with parallel substitution:

**lemma** *prv\_imp\_scnj\_insert*:  
**assumes**  $F \subseteq \text{fmla}$  **and** *finite*  $F$  **and**  $\varphi \in \text{fmla}$   
**shows**  $\text{prv} (\text{imp} (\text{scnj} (\text{insert } \varphi \ F)) (\text{cnj } \varphi (\text{scnj } F)))$   
 $\langle \text{proof} \rangle$

**lemma** *prv\_implies\_scnj\_insert*:  
**assumes**  $F \subseteq \text{fmla}$  **and** *finite*  $F$  **and**  $\varphi \in \text{fmla}$   
**and**  $\text{prv} (\text{scnj} (\text{insert } \varphi \ F))$   
**shows**  $\text{prv} (\text{cnj } \varphi (\text{scnj } F))$   
 $\langle \text{proof} \rangle$

**lemma** *prv\_imp\_cnj\_scnj*:  
**assumes**  $F \subseteq \text{fmla}$  **and** *finite*  $F$  **and**  $\varphi \in \text{fmla}$   
**shows**  $\text{prv} (\text{imp} (\text{cnj } \varphi (\text{scnj } F)) (\text{scnj} (\text{insert } \varphi \ F)))$   
 $\langle \text{proof} \rangle$

**lemma** *prv\_implies\_cnj\_scnj*:  
**assumes**  $F \subseteq \text{fmla}$  **and** *finite*  $F$  **and**  $\varphi \in \text{fmla}$   
**and**  $\text{prv} (\text{cnj } \varphi (\text{scnj } F))$   
**shows**  $\text{prv} (\text{scnj} (\text{insert } \varphi \ F))$   
 $\langle \text{proof} \rangle$

**lemma** *prv\_eqv\_scnj\_insert*:  
**assumes**  $F \subseteq \text{fmla}$  **and** *finite*  $F$  **and**  $\varphi \in \text{fmla}$

**shows**  $prv (eqv (scnj (insert \varphi F)) (cnj \varphi (scnj F)))$   
 $\langle proof \rangle$

**lemma** *prv\_scnj1\_imp*:  
 $\varphi \in fmla \implies prv (imp (scnj \{\varphi\}) \varphi)$   
 $\langle proof \rangle$

**lemma** *prv\_imp\_scnj1*:  
 $\varphi \in fmla \implies prv (imp \varphi (scnj \{\varphi\}))$   
 $\langle proof \rangle$

**lemma** *prv\_scnj2\_imp\_cnj*:  
 $\varphi \in fmla \implies \psi \in fmla \implies prv (imp (scnj \{\varphi, \psi\}) (cnj \varphi \psi))$   
 $\langle proof \rangle$

**lemma** *prv\_cnj\_imp\_scnj2*:  
 $\varphi \in fmla \implies \psi \in fmla \implies prv (imp (cnj \varphi \psi) (scnj \{\varphi, \psi\}))$   
 $\langle proof \rangle$

**lemma** *prv\_imp\_imp\_scnj2*:  
 $\varphi \in fmla \implies \psi \in fmla \implies prv (imp \varphi (imp \psi (scnj \{\varphi, \psi\})))$   
 $\langle proof \rangle$

**lemma** *prv\_rawpsubst\_scnj*:  
**assumes**  $F \subseteq fmla$  *finite*  $F$   
**and**  $snd \text{ ' (set } txs) \subseteq var \text{ fst ' (set } txs) \subseteq trm$   
**shows**  $prv (eqv (rawpsubst (scnj F) txs) (scnj ((\lambda \varphi. rawpsubst \varphi txs) \text{ ' } F)))$   
 $\langle proof \rangle$

**lemma** *prv\_psubst\_scnj*:  
**assumes**  $F \subseteq fmla$  *finite*  $F$   
**and**  $snd \text{ ' (set } txs) \subseteq var \text{ fst ' (set } txs) \subseteq trm$   
**and**  $distinct (map snd txs)$   
**shows**  $prv (eqv (psubst (scnj F) txs) (scnj ((\lambda \varphi. psubst \varphi txs) \text{ ' } F)))$   
 $\langle proof \rangle$

**lemma** *prv\_imp\_psubst\_scnj*:  
**assumes**  $F \subseteq fmla$  *finite*  $F$   $snd \text{ ' set } txs \subseteq var \text{ fst ' set } txs \subseteq trm$   
**and**  $distinct (map snd txs)$   
**shows**  $prv (imp (psubst (scnj F) txs) (scnj ((\lambda \varphi. psubst \varphi txs) \text{ ' } F)))$   
 $\langle proof \rangle$

**lemma** *prv\_psubst\_scnj\_imp*:  
**assumes**  $F \subseteq fmla$  *finite*  $F$   $snd \text{ ' set } txs \subseteq var \text{ fst ' set } txs \subseteq trm$   
**and**  $distinct (map snd txs)$   
**shows**  $prv (imp (scnj ((\lambda \varphi. psubst \varphi txs) \text{ ' } F)) (psubst (scnj F) txs))$   
 $\langle proof \rangle$

### 3.2.5 Consistency and $\omega$ -consistency

**definition** *consistent* :: *bool* **where**  
 $consistent \equiv \neg prv \text{ fls}$

**lemma** *consistent\_def2*:  $consistent \longleftrightarrow (\exists \varphi \in fmla. \neg prv \varphi)$   
 $\langle proof \rangle$

**lemma** *consistent\_def3*:  $consistent \longleftrightarrow (\forall \varphi \in fmla. \neg (prv \varphi \wedge prv (neg \varphi)))$   
 ⟨proof⟩

**definition**  $\omega consistent :: bool$  **where**

$\omega consistent \equiv$   
 $\forall \varphi \in fmla. \forall x \in var. Fvars \varphi = \{x\} \longrightarrow$   
 $(\forall n \in num. prv (neg (subst \varphi n x)))$   
 $\longrightarrow$   
 $\neg prv (neg (neg (exi x \varphi)))$

The above particularly strong version of  $\omega consistent$  is used for the sake of working without assuming classical logic. It of course implies the more standard formulations for classical logic:

**definition**  $\omega consistentStd1 :: bool$  **where**

$\omega consistentStd1 \equiv$   
 $\forall \varphi \in fmla. \forall x \in var. Fvars \varphi = \{x\} \longrightarrow$   
 $(\forall n \in num. prv (neg (subst \varphi n x))) \longrightarrow \neg prv (exi x \varphi)$

**definition**  $\omega consistentStd2 :: bool$  **where**

$\omega consistentStd2 \equiv$   
 $\forall \varphi \in fmla. \forall x \in var. Fvars \varphi = \{x\} \longrightarrow$   
 $(\forall n \in num. prv (subst \varphi n x)) \longrightarrow \neg prv (exi x (neg \varphi))$

**lemma**  $\omega consistent\_impliesStd1$ :

$\omega consistent \implies$   
 $\omega consistentStd1$   
 ⟨proof⟩

**lemma**  $\omega consistent\_impliesStd2$ :

$\omega consistent \implies$   
 $\omega consistentStd2$   
 ⟨proof⟩

In the presence of classical logic deduction, the stronger condition is equivalent to the standard ones:

**lemma**  $\omega consistent\_iffStd1$ :

**assumes**  $\bigwedge \varphi. \varphi \in fmla \implies prv (imp (neg (neg \varphi)) \varphi)$   
**shows**  $\omega consistent \longleftrightarrow \omega consistentStd1$   
 ⟨proof⟩

**lemma**  $\omega consistent\_iffStd2$ :

**assumes**  $\bigwedge \varphi. \varphi \in fmla \implies prv (imp (neg (neg \varphi)) \varphi)$   
**shows**  $\omega consistent \longleftrightarrow \omega consistentStd2$   
 ⟨proof⟩

$\omega$ -consistency implies consistency:

**lemma**  $\omega consistentStd1\_implies\_consistent$ :

**assumes**  $\omega consistentStd1$   
**shows**  $consistent$   
 ⟨proof⟩

**lemma**  $\omega consistentStd2\_implies\_consistent$ :

**assumes**  $\omega consistentStd2$   
**shows**  $consistent$   
 ⟨proof⟩

**corollary**  $\omega consistent\_implies\_consistent$ :

**assumes**  $\omega consistent$

**shows** *consistent*  
 ⟨*proof*⟩

**end** — context *Deduct\_with\_False*

### 3.3 Deduction Considering False and Disjunction

```

locale Deduct_with_False_Disj =
  Syntax_with_Connectives_False_Disj
  var trm fmla Var FvarsT substT Fvars subst
  egl cnj imp all exi
  fls
  dsj
  +
  Deduct_with_False
  var trm fmla Var FvarsT substT Fvars subst
  egl cnj imp all exi
  fls
  num
  prv
for
  var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
  and Var FvarsT substT Fvars subst
  and egl cnj imp all exi
  and fls
  and dsj
  and num
  and prv
  +
assumes
  prv_dsj_impL:
   $\bigwedge \varphi \chi. \varphi \in \text{fmla} \implies \chi \in \text{fmla} \implies$ 
  prv (imp  $\varphi$  (dsj  $\varphi$   $\chi$ ))
  and
  prv_dsj_impR:
   $\bigwedge \varphi \chi. \varphi \in \text{fmla} \implies \chi \in \text{fmla} \implies$ 
  prv (imp  $\chi$  (dsj  $\varphi$   $\chi$ ))
  and
  prv_imp_dsjE:
   $\bigwedge \varphi \chi \psi. \varphi \in \text{fmla} \implies \chi \in \text{fmla} \implies \psi \in \text{fmla} \implies$ 
  prv (imp (imp  $\varphi$   $\psi$ ) (imp (imp  $\chi$   $\psi$ ) (imp (dsj  $\varphi$   $\chi$ )  $\psi$ )))
begin

lemma prv_imp_dsjEE:
  assumes  $\varphi[\text{simp}]: \varphi \in \text{fmla}$  and  $\chi[\text{simp}]: \chi \in \text{fmla}$  and  $\psi[\text{simp}]: \psi \in \text{fmla}$ 
  assumes prv (imp  $\varphi$   $\psi$ ) and prv (imp  $\chi$   $\psi$ )
  shows prv (imp (dsj  $\varphi$   $\chi$ )  $\psi$ )
  ⟨proof⟩

lemma prv_dsj_cases:
  assumes  $\varphi 1 \in \text{fmla}$   $\varphi 2 \in \text{fmla}$   $\chi \in \text{fmla}$ 
  and prv (dsj  $\varphi 1$   $\varphi 2$ ) and prv (imp  $\varphi 1$   $\chi$ ) and prv (imp  $\varphi 2$   $\chi$ )
  shows prv  $\chi$ 
  ⟨proof⟩

```

#### 3.3.1 Disjunction vs. disjunction

**lemma** *prv\_dsj\_com\_imp:*

**assumes**  $\varphi[simp]: \varphi \in fmla$  **and**  $\chi[simp]: \chi \in fmla$   
**shows**  $prv (imp (dsj \varphi \chi) (dsj \chi \varphi))$   
 $\langle proof \rangle$

**lemma** *prv\_dsj\_com*:  
**assumes**  $\varphi[simp]: \varphi \in fmla$  **and**  $\chi[simp]: \chi \in fmla$   
**shows**  $prv (eqv (dsj \varphi \chi) (dsj \chi \varphi))$   
 $\langle proof \rangle$

**lemma** *prv\_dsj\_assoc\_imp1*:  
**assumes**  $\varphi[simp]: \varphi \in fmla$  **and**  $\chi[simp]: \chi \in fmla$  **and**  $\psi[simp]: \psi \in fmla$   
**shows**  $prv (imp (dsj \varphi (dsj \chi \psi)) (dsj (dsj \varphi \chi) \psi))$   
 $\langle proof \rangle$

**lemma** *prv\_dsj\_assoc\_imp2*:  
**assumes**  $\varphi[simp]: \varphi \in fmla$  **and**  $\chi[simp]: \chi \in fmla$  **and**  $\psi[simp]: \psi \in fmla$   
**shows**  $prv (imp (dsj (dsj \varphi \chi) \psi) (dsj \varphi (dsj \chi \psi)))$   
 $\langle proof \rangle$

**lemma** *prv\_dsj\_assoc*:  
**assumes**  $\varphi[simp]: \varphi \in fmla$  **and**  $\chi[simp]: \chi \in fmla$  **and**  $\psi[simp]: \psi \in fmla$   
**shows**  $prv (eqv (dsj \varphi (dsj \chi \psi)) (dsj (dsj \varphi \chi) \psi))$   
 $\langle proof \rangle$

**lemma** *prv\_dsj\_com\_imp3*:  
**assumes**  $\varphi1 \in fmla$   $\varphi2 \in fmla$   $\varphi3 \in fmla$   
**shows**  $prv (imp (dsj \varphi1 (dsj \varphi2 \varphi3))$   
 $(dsj \varphi2 (dsj \varphi1 \varphi3)))$   
 $\langle proof \rangle$

**lemma** *prv\_dsj\_mono*:  
 $\varphi1 \in fmla \implies \varphi2 \in fmla \implies \chi1 \in fmla \implies \chi2 \in fmla \implies$   
 $prv (imp \varphi1 \chi1) \implies prv (imp \varphi2 \chi2) \implies prv (imp (dsj \varphi1 \varphi2) (dsj \chi1 \chi2))$   
 $\langle proof \rangle$

### 3.3.2 Disjunction vs. conjunction

**lemma** *prv\_cnj\_dsj\_distrib1*:  
**assumes**  $\varphi[simp]: \varphi \in fmla$  **and**  $\chi1[simp]: \chi1 \in fmla$  **and**  $\chi2[simp]: \chi2 \in fmla$   
**shows**  $prv (imp (cnj \varphi (dsj \chi1 \chi2)) (dsj (cnj \varphi \chi1) (cnj \varphi \chi2)))$   
 $\langle proof \rangle$

**lemma** *prv\_cnj\_dsj\_distrib2*:  
**assumes**  $\varphi[simp]: \varphi \in fmla$  **and**  $\chi1[simp]: \chi1 \in fmla$  **and**  $\chi2[simp]: \chi2 \in fmla$   
**shows**  $prv (imp (dsj (cnj \varphi \chi1) (cnj \varphi \chi2)) (cnj \varphi (dsj \chi1 \chi2)))$   
 $\langle proof \rangle$

**lemma** *prv\_cnj\_dsj\_distrib*:  
**assumes**  $\varphi[simp]: \varphi \in fmla$  **and**  $\chi1[simp]: \chi1 \in fmla$  **and**  $\chi2[simp]: \chi2 \in fmla$   
**shows**  $prv (eqv (cnj \varphi (dsj \chi1 \chi2)) (dsj (cnj \varphi \chi1) (cnj \varphi \chi2)))$   
 $\langle proof \rangle$

**lemma** *prv\_dsj\_cnj\_distrib1*:  
**assumes**  $\varphi[simp]: \varphi \in fmla$  **and**  $\chi1[simp]: \chi1 \in fmla$  **and**  $\chi2[simp]: \chi2 \in fmla$   
**shows**  $prv (imp (dsj \varphi (cnj \chi1 \chi2)) (cnj (dsj \varphi \chi1) (dsj \varphi \chi2)))$   
 $\langle proof \rangle$

**lemma** *prv\_dsj\_cnj\_distrib2*:



**assumes**  $\varphi[simp]: \varphi \in fmla$  **and**  $\chi1[simp]: \chi1 \in fmla$  **and**  $\chi2[simp]: \chi2 \in fmla$   
**shows**  $prv (imp (cnj (dsj \varphi \chi1) (dsj \varphi \chi2)) (dsj \varphi (cnj \chi1 \chi2)))$   
 $\langle proof \rangle$

**lemma**  $prv\_dsj\_cnj\_distrib$ :

**assumes**  $\varphi[simp]: \varphi \in fmla$  **and**  $\chi1[simp]: \chi1 \in fmla$  **and**  $\chi2[simp]: \chi2 \in fmla$   
**shows**  $prv (equiv (dsj \varphi (cnj \chi1 \chi2)) (cnj (dsj \varphi \chi1) (dsj \varphi \chi2)))$   
 $\langle proof \rangle$

### 3.3.3 Disjunction vs. True and False

**lemma**  $prv\_dsjR\_fls: \varphi \in fmla \implies prv (equiv (dsj fls \varphi) \varphi)$   
 $\langle proof \rangle$

**lemma**  $prv\_dsjL\_fls: \varphi \in fmla \implies prv (equiv (dsj \varphi fls) \varphi)$   
 $\langle proof \rangle$

**lemma**  $prv\_dsjR\_tru: \varphi \in fmla \implies prv (equiv (dsj tru \varphi) tru)$   
 $\langle proof \rangle$

**lemma**  $prv\_dsjL\_tru: \varphi \in fmla \implies prv (equiv (dsj \varphi tru) tru)$   
 $\langle proof \rangle$

### 3.3.4 Set-based disjunctions

Just like for conjunctions, these are based on properties of the auxiliary list disjunctions.

**lemma**  $prv\_imp\_lds\_in$ :

**assumes**  $set \varphi s \subseteq fmla$   
**and**  $\varphi \in set \varphi s$   
**shows**  $prv (imp \varphi (lds \varphi s))$

$\langle proof \rangle$

**lemma**  $prv\_imp\_lds$ :

**assumes**  $\chi \in fmla$  **and**  $set \varphi s \subseteq fmla$   
**and**  $\varphi \in set \varphi s$  **and**  $prv (imp \chi \varphi)$   
**shows**  $prv (imp \chi (lds \varphi s))$

$\langle proof \rangle$

**lemma**  $prv\_lds\_imp$ :

**assumes**  $\chi \in fmla$  **and**  $set \varphi s \subseteq fmla$   
**and**  $\bigwedge \varphi. \varphi \in set \varphi s \implies prv (imp \varphi \chi)$   
**shows**  $prv (imp (lds \varphi s) \chi)$

$\langle proof \rangle$

**lemma**  $prv\_lds\_imp\_inner$ :

**assumes**  $\varphi \in fmla$   $set \varphi1s \subseteq fmla$   $set \varphi2s \subseteq fmla$   
**shows**  $prv (imp (lds (\varphi1s @ \varphi \# \varphi2s)) (dsj \varphi (lds (\varphi1s @ \varphi2s))))$

$\langle proof \rangle$

**lemma**  $prv\_lds\_imp\_remdups$ :

**assumes**  $set \varphi s \subseteq fmla$   
**shows**  $prv (imp (lds \varphi s) (lds (remdups \varphi s)))$

$\langle proof \rangle$

**lemma**  $prv\_lds\_mono$ :

**assumes**  $\varphi2s: set \varphi2s \subseteq fmla$  **and**  $set \varphi1s \subseteq set \varphi2s$   
**shows**  $prv (imp (lds \varphi1s) (lds \varphi2s))$

$\langle proof \rangle$

**lemma** *prv\_ldsj\_eqv*:  
**assumes**  $set\ \varphi 1s \subseteq fmla$  **and**  $set\ \varphi 2s = set\ \varphi 1s$   
**shows**  $prv\ (eqv\ (ldsj\ \varphi 1s)\ (ldsj\ \varphi 2s))$   
*<proof>*

**lemma** *prv\_ldsj\_mono\_imp*:  
**assumes**  $set\ \varphi 1s \subseteq fmla$   $set\ \varphi 2s \subseteq fmla$  **and**  $\forall\ \varphi 1 \in set\ \varphi 1s. \exists\ \varphi 2 \in set\ \varphi 2s. prv\ (imp\ \varphi 1\ \varphi 2)$   
**shows**  $prv\ (imp\ (ldsj\ \varphi 1s)\ (ldsj\ \varphi 2s))$   
*<proof>*

Just like set-based conjunction, set-based disjunction commutes with substitution only up to provably equivalence:

**lemma** *prv\_subst\_sdsj*:  
 $F \subseteq fmla \implies finite\ F \implies t \in trm \implies x \in var \implies$   
 $prv\ (eqv\ (subst\ (sdsj\ F)\ t\ x)\ (sdsj\ ((\lambda\varphi. subst\ \varphi\ t\ x)\ 'F)))$   
*<proof>*

**lemma** *prv\_imp\_sdsj\_in*:  
**assumes**  $\varphi \in fmla$  **and**  $F \subseteq fmla$  *finite*  $F$   
**and**  $\varphi \in F$   
**shows**  $prv\ (imp\ \varphi\ (sdsj\ F))$   
*<proof>*

**lemma** *prv\_imp\_sdsj*:  
**assumes**  $\chi \in fmla$  **and**  $F \subseteq fmla$  *finite*  $F$   
**and**  $\varphi \in F$  **and**  $prv\ (imp\ \chi\ \varphi)$   
**shows**  $prv\ (imp\ \chi\ (sdsj\ F))$   
*<proof>*

**lemma** *prv\_sdsj\_imp*:  
**assumes**  $\chi \in fmla$  **and**  $F \subseteq fmla$  *finite*  $F$   
**and**  $\bigwedge\varphi. \varphi \in F \implies prv\ (imp\ \varphi\ \chi)$   
**shows**  $prv\ (imp\ (sdsj\ F)\ \chi)$   
*<proof>*

**lemma** *prv\_sdsj\_mono*:  
**assumes**  $F2 \subseteq fmla$  **and**  $F1 \subseteq F2$  **and** *finite*  $F2$   
**shows**  $prv\ (imp\ (sdsj\ F1)\ (sdsj\ F2))$   
*<proof>*

**lemma** *prv\_sdsj\_mono\_imp*:  
**assumes**  $F1 \subseteq fmla$   $F2 \subseteq fmla$  *finite*  $F1$  *finite*  $F2$   
**and**  $\forall\ \varphi 1 \in F1. \exists\ \varphi 2 \in F2. prv\ (imp\ \varphi 1\ \varphi 2)$   
**shows**  $prv\ (imp\ (sdsj\ F1)\ (sdsj\ F2))$   
*<proof>*

**lemma** *prv\_sdsj\_cases*:  
**assumes**  $F \subseteq fmla$  *finite*  $F$   $\psi \in fmla$   
**and**  $prv\ (sdsj\ F)$  **and**  $\bigwedge\varphi. \varphi \in F \implies prv\ (imp\ \varphi\ \psi)$   
**shows**  $prv\ \psi$   
*<proof>*

**lemma** *prv\_sdsj1\_imp*:  
 $\varphi \in fmla \implies prv\ (imp\ (sdsj\ \{\varphi\})\ \varphi)$   
*<proof>*

**lemma** *prv\_imp\_sdsj1*:

$\varphi \in \text{fmla} \implies \text{prv} (\text{imp } \varphi (\text{sdsj } \{\varphi\}))$   
 <proof>

**lemma** *prv\_sdsj2\_imp\_dsj*:  
 $\varphi \in \text{fmla} \implies \psi \in \text{fmla} \implies \text{prv} (\text{imp} (\text{sdsj } \{\varphi, \psi\}) (\text{dsj } \varphi \psi))$   
 <proof>

**lemma** *prv\_dsj\_imp\_sdsj2*:  
 $\varphi \in \text{fmla} \implies \psi \in \text{fmla} \implies \text{prv} (\text{imp} (\text{dsj } \varphi \psi) (\text{sdsj } \{\varphi, \psi\}))$   
 <proof>

Commutation with parallel substitution:

**lemma** *prv\_rawpsubst\_sdsj*:  
**assumes**  $F \subseteq \text{fmla}$  *finite*  $F$   
**and**  $\text{snd } \langle \text{set } \text{txs} \rangle \subseteq \text{var } \text{fst } \langle \text{set } \text{txs} \rangle \subseteq \text{trm}$   
**shows**  $\text{prv} (\text{eqv} (\text{rawpsubst} (\text{sdsj } F) \text{txs}) (\text{sdsj} ((\lambda \varphi. \text{rawpsubst } \varphi \text{txs}) \langle F \rangle)))$   
 <proof>

**lemma** *prv\_psubst\_sdsj*:  
**assumes**  $F \subseteq \text{fmla}$  *finite*  $F$   
**and**  $\text{snd } \langle \text{set } \text{txs} \rangle \subseteq \text{var } \text{fst } \langle \text{set } \text{txs} \rangle \subseteq \text{trm}$   
**and** *distinct*  $(\text{map } \text{snd } \text{txs})$   
**shows**  $\text{prv} (\text{eqv} (\text{psubst} (\text{sdsj } F) \text{txs}) (\text{sdsj} ((\lambda \varphi. \text{psubst } \varphi \text{txs}) \langle F \rangle)))$   
 <proof>

**end** — context *Deduct\_with\_False\_Disj*

### 3.4 Deduction with Quantified Variable Renaming Included

**locale** *Deduct\_with\_False\_Disj\_Rename* =  
*Deduct\_with\_False\_Disj*  
*var trm fmla Var FvarsT substT Fvars subst*  
*eql cnj imp all exi*  
*fls*  
*dsj*  
*num*  
*prv*  
 +  
*Syntax\_with\_Connectives\_Rename*  
*var trm fmla Var FvarsT substT Fvars subst*  
*eql cnj imp all exi*  
**for**  
*var :: 'var set and trm :: 'trm set and fmla :: 'fmla set*  
**and** *Var FvarsT substT Fvars subst*  
**and** *eql cnj imp all exi*  
**and** *fls*  
**and** *dsj*  
**and** *num*  
**and** *prv*

### 3.5 Deduction with PseudoOrder Axioms Included

We assume a two-variable formula Lq that satisfies two axioms resembling the properties of the strict or nonstrict ordering on naturals. The choice of these axioms is motivated by an abstract account of Rosser’s trick to improve on Gödel’s First Incompleteness Theorem, reported in our CADE 2019 paper [1].

```

locale Deduct_with_PseudoOrder =
  Deduct_with_False_Disj
    var trm fmla Var FvarsT substT Fvars subst
    eql cnj imp all exi
    fls
    dsj
    num
    prv
  +
  Syntax_PseudoOrder
    var trm fmla Var FvarsT substT Fvars subst
    eql cnj imp all exi
    fls
    dsj
    num
    Lq
for
  var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
and Var FvarsT substT Fvars subst
and eql cnj imp all exi
and fls
and dsj
and num
and prv
and Lq
  +
assumes
  Lq_num:
  let LLq = ( $\lambda$  t1 t2. psubst Lq [(t1,zz), (t2,yy)]) in
   $\forall \varphi \in \text{fmla}. \forall q \in \text{num}. \text{Fvars } \varphi = \{\text{zz}\} \wedge (\forall p \in \text{num}. \text{prv } (\text{subst } \varphi \text{ } p \text{ } \text{zz}))$ 
   $\longrightarrow \text{prv } (\text{all } \text{zz } (\text{imp } (\text{LLq } (\text{Var } \text{zz}) \text{ } q) \varphi))$ 
and
  Lq_num2:
  let LLq = ( $\lambda$  t1 t2. psubst Lq [(t1,zz), (t2,yy)]) in
   $\forall p \in \text{num}. \exists P \subseteq \text{num}. \text{finite } P \wedge \text{prv } (\text{dsj } (\text{sdsj } \{\text{eql } (\text{Var } \text{yy}) \text{ } r \mid r. r \in P\}) (\text{LLq } p \text{ } (\text{Var } \text{yy})))$ 
begin

  lemma LLq_num:
  assumes  $\varphi \in \text{fmla } q \in \text{num } \text{Fvars } \varphi = \{\text{zz}\} \forall p \in \text{num}. \text{prv } (\text{subst } \varphi \text{ } p \text{ } \text{zz})$ 
  shows  $\text{prv } (\text{all } \text{zz } (\text{imp } (\text{LLq } (\text{Var } \text{zz}) \text{ } q) \varphi))$ 
  <proof>

  lemma LLq_num2:
  assumes  $p \in \text{num}$ 
  shows  $\exists P \subseteq \text{num}. \text{finite } P \wedge \text{prv } (\text{dsj } (\text{sdsj } \{\text{eql } (\text{Var } \text{yy}) \text{ } r \mid r. r \in P\}) (\text{LLq } p \text{ } (\text{Var } \text{yy})))$ 
  <proof>

end — context Deduct_with_PseudoOrder

```

# Chapter 4

## Natural Deduction

We develop a natural deduction system based on the Hilbert system.

```
context Deduct_with_False_Disj
begin
```

### 4.1 Natural Deduction from the Hilbert System

```
definition nprv :: 'fmla set  $\Rightarrow$  'fmla  $\Rightarrow$  bool where
nprv F  $\varphi \equiv$  prv (imp (scnj F)  $\varphi$ )
```

```
lemma nprv_hyp[simp,intro]:
 $\varphi \in F \Longrightarrow F \subseteq \text{fmla} \Longrightarrow \text{finite } F \Longrightarrow \text{nprv } F \ \varphi$ 
<proof>
```

### 4.2 Structural Rules for the Natural Deduction Relation

```
lemma prv_nprv0I: prv  $\varphi \Longrightarrow \varphi \in \text{fmla} \Longrightarrow \text{nprv } \{\} \ \varphi$ 
<proof>
```

```
lemma prv_nprv_emp:  $\varphi \in \text{fmla} \Longrightarrow \text{prv } \varphi \longleftrightarrow \text{nprv } \{\} \ \varphi$ 
<proof>
```

```
lemma nprv_mono:
assumes nprv G  $\varphi$ 
and  $F \subseteq \text{fmla}$  finite F  $G \subseteq F$   $\varphi \in \text{fmla}$ 
shows nprv F  $\varphi$ 
<proof>
```

```
lemma nprv_cut:
assumes nprv F  $\varphi$  and nprv (insert  $\varphi$  F)  $\psi$ 
and  $F \subseteq \text{fmla}$  finite F  $\varphi \in \text{fmla}$   $\psi \in \text{fmla}$ 
shows nprv F  $\psi$ 
<proof>
```

```
lemma nprv_strong_cut2:
nprv F  $\varphi1 \Longrightarrow \text{nprv}$  (insert  $\varphi1$  F)  $\varphi2 \Longrightarrow \text{nprv}$  (insert  $\varphi2$  (insert  $\varphi1$  F))  $\psi \Longrightarrow$ 
 $F \subseteq \text{fmla} \Longrightarrow \text{finite } F \Longrightarrow \varphi1 \in \text{fmla} \Longrightarrow \varphi2 \in \text{fmla} \Longrightarrow \psi \in \text{fmla} \Longrightarrow$ 
nprv F  $\psi$ 
<proof>
```

```
lemma nprv_cut2:
```

$nprv\ F\ \varphi1 \implies nprv\ F\ \varphi2 \implies$   
 $F \subseteq fmla \implies finite\ F \implies \varphi1 \in fmla \implies \varphi2 \in fmla \implies \psi \in fmla \implies$   
 $nprv\ (insert\ \varphi2\ (insert\ \varphi1\ F))\ \psi \implies nprv\ F\ \psi$   
 <proof>

Useful for fine control of the eigenformula:

**lemma** *nprv\_insertShiftI*:  
 $nprv\ (insert\ \varphi1\ (insert\ \varphi2\ F))\ \psi \implies nprv\ (insert\ \varphi2\ (insert\ \varphi1\ F))\ \psi$   
 <proof>

**lemma** *nprv\_insertShift2I*:  
 $nprv\ (insert\ \varphi3\ (insert\ \varphi1\ (insert\ \varphi2\ F)))\ \psi \implies nprv\ (insert\ \varphi1\ (insert\ \varphi2\ (insert\ \varphi3\ F)))\ \psi$   
 <proof>

### 4.3 Back and Forth between Hilbert and Natural Deduction

This is now easy, thanks to the large number of facts we have proved for Hilbert-style deduction

**lemma** *prv\_nprvI*:  $prv\ \varphi \implies \varphi \in fmla \implies F \subseteq fmla \implies finite\ F \implies nprv\ F\ \varphi$   
 <proof>

**thm** *prv\_nprv0I*

**lemma** *prv\_nprv1I*:  
**assumes**  $\varphi \in fmla\ \psi \in fmla$  **and**  $prv\ (imp\ \varphi\ \psi)$   
**shows**  $nprv\ \{\varphi\}\ \psi$   
 <proof>

**lemma** *prv\_nprv2I*:  
**assumes**  $prv\ (imp\ \varphi1\ (imp\ \varphi2\ \psi))\ \varphi1 \in fmla\ \varphi2 \in fmla\ \psi \in fmla$   
**shows**  $nprv\ \{\varphi1, \varphi2\}\ \psi$   
 <proof>

**lemma** *nprv\_prvI*:  $nprv\ \{\}\ \varphi \implies \varphi \in fmla \implies prv\ \varphi$   
 <proof>

### 4.4 More Structural Properties

**lemma** *nprv\_clear*:  $nprv\ \{\}\ \varphi \implies F \subseteq fmla \implies finite\ F \implies \varphi \in fmla \implies nprv\ F\ \varphi$   
 <proof>

**lemma** *nprv\_cut\_set*:  
**assumes**  $F: finite\ F\ F \subseteq fmla$  **and**  $G: finite\ G\ G \subseteq fmla\ \chi \in fmla$   
**and**  $n1: \bigwedge \psi. \psi \in G \implies nprv\ F\ \psi$  **and**  $n2: nprv\ (G \cup F)\ \chi$   
**shows**  $nprv\ F\ \chi$   
 <proof>

**lemma** *nprv\_clear2\_1*:  
 $nprv\ \{\varphi2\}\ \psi \implies \varphi1 \in fmla \implies \varphi2 \in fmla \implies \psi \in fmla \implies$   
 $nprv\ \{\varphi1, \varphi2\}\ \psi$   
 <proof>

**lemma** *nprv\_clear2\_2*:  
 $nprv\ \{\varphi1\}\ \psi \implies \varphi1 \in fmla \implies \varphi2 \in fmla \implies \psi \in fmla \implies$   
 $nprv\ \{\varphi1, \varphi2\}\ \psi$   
 <proof>



**lemma** *nprv\_clear5\_5*:  
*nprv*  $\{\varphi 1, \varphi 2, \varphi 3, \varphi 4\} \psi \implies \varphi 1 \in \text{fmla} \implies \varphi 2 \in \text{fmla} \implies \varphi 3 \in \text{fmla} \implies \varphi 4 \in \text{fmla} \implies \varphi 5 \in \text{fmla}$   
 $\implies \psi \in \text{fmla} \implies$   
*nprv*  $\{\varphi 1, \varphi 2, \varphi 3, \varphi 4, \varphi 5\} \psi$   
 $\langle \text{proof} \rangle$

## 4.5 Properties Involving Substitution

**lemma** *nprv\_subst*:  
**assumes**  $x \in \text{var } t \in \text{trm } \psi \in \text{fmla } \text{finite } F \ F \subseteq \text{fmla}$   
**and**  $1: \text{nprv } F \ \psi$   
**shows** *nprv*  $((\lambda \varphi. \text{subst } \varphi \ t \ x) \ ' F) (\text{subst } \psi \ t \ x)$   
 $\langle \text{proof} \rangle$

**lemma** *nprv\_subst\_fresh*:  
**assumes**  $0: x \in \text{var } t \in \text{trm } \psi \in \text{fmla } \text{finite } F \ F \subseteq \text{fmla}$   
*nprv*  $F \ \psi$  **and**  $1: x \notin \bigcup (F\text{vars } \ ' F)$   
**shows** *nprv*  $F (\text{subst } \psi \ t \ x)$   
 $\langle \text{proof} \rangle$

**lemma** *nprv\_subst\_rev*:  
**assumes**  $0: x \in \text{var } y \in \text{var } \psi \in \text{fmla } \text{finite } F \ F \subseteq \text{fmla}$   
**and**  $f: y = x \vee (y \notin F\text{vars } \psi \wedge y \notin \bigcup (F\text{vars } \ ' F))$   
**and**  $1: \text{nprv } ((\lambda \varphi. \text{subst } \varphi \ (\text{Var } y) \ x) \ ' F) (\text{subst } \psi \ (\text{Var } y) \ x)$   
**shows** *nprv*  $F \ \psi$   
 $\langle \text{proof} \rangle$

**lemma** *nprv\_psubst*:  
**assumes**  $0: \text{snd } \ ' \text{set } \text{txs} \subseteq \text{var } \text{fst } \ ' \text{set } \text{txs} \subseteq \text{trm } \psi \in \text{fmla } \text{finite } F \ F \subseteq \text{fmla}$   
 $\text{distinct } (\text{map } \text{snd } \text{txs})$   
**and**  $1: \text{nprv } F \ \psi$   
**shows** *nprv*  $((\lambda \varphi. \text{psubst } \varphi \ \text{txs}) \ ' F) (\text{psubst } \psi \ \text{txs})$   
 $\langle \text{proof} \rangle$

## 4.6 Introduction and Elimination Rules

We systematically leave the side-conditions at the end, to simplify reasoning.

**lemma** *nprv\_impI*:  
*nprv*  $(\text{insert } \varphi \ F) \ \psi \implies$   
 $F \subseteq \text{fmla} \implies \text{finite } F \implies \varphi \in \text{fmla} \implies \psi \in \text{fmla} \implies$   
*nprv*  $F (\text{imp } \varphi \ \psi)$   
 $\langle \text{proof} \rangle$

**lemma** *nprv\_impI\_rev*:  
**assumes** *nprv*  $F (\text{imp } \varphi \ \psi)$   
**and**  $F \subseteq \text{fmla}$  **and**  $\text{finite } F$  **and**  $\varphi \in \text{fmla}$  **and**  $\psi \in \text{fmla}$   
**shows** *nprv*  $(\text{insert } \varphi \ F) \ \psi$   
 $\langle \text{proof} \rangle$

**lemma** *nprv\_impI\_rev2*:  
**assumes** *nprv*  $F (\text{imp } \varphi \ \psi)$  **and**  $G: \text{insert } \varphi \ F \subseteq G$   
**and**  $G \subseteq \text{fmla}$  **and**  $\text{finite } G$  **and**  $\varphi \in \text{fmla}$  **and**  $\psi \in \text{fmla}$   
**shows** *nprv*  $G \ \psi$   
 $\langle \text{proof} \rangle$



**lemma** *nprv\_mp*:

$nprv\ F\ (imp\ \varphi\ \psi) \implies nprv\ F\ \varphi \implies$   
 $F \subseteq fmla \implies finite\ F \implies \varphi \in fmla \implies \psi \in fmla \implies$   
 $nprv\ F\ \psi$   
(proof)

**lemma** *nprv\_impE*:

$nprv\ F\ (imp\ \varphi\ \psi) \implies nprv\ F\ \varphi \implies nprv\ (insert\ \psi\ F)\ \chi \implies$   
 $F \subseteq fmla \implies finite\ F \implies \varphi \in fmla \implies \psi \in fmla \implies \chi \in fmla \implies$   
 $nprv\ F\ \chi$   
(proof)

**lemmas** *nprv\_impE0* = *nprv\_impE*[*OF nprv\_hyp \_ \_*, *simped*]

**lemmas** *nprv\_impE1* = *nprv\_impE*[*OF \_ nprv\_hyp \_*, *simped*]

**lemmas** *nprv\_impE2* = *nprv\_impE*[*OF \_ \_ nprv\_hyp*, *simped*]

**lemmas** *nprv\_impE01* = *nprv\_impE*[*OF nprv\_hyp nprv\_hyp \_*, *simped*]

**lemmas** *nprv\_impE02* = *nprv\_impE*[*OF nprv\_hyp \_ nprv\_hyp*, *simped*]

**lemmas** *nprv\_impE12* = *nprv\_impE*[*OF \_ nprv\_hyp nprv\_hyp*, *simped*]

**lemmas** *nprv\_impE012* = *nprv\_impE*[*OF nprv\_hyp nprv\_hyp nprv\_hyp*, *simped*]

**lemma** *nprv\_cnjI*:

$nprv\ F\ \varphi \implies nprv\ F\ \psi \implies$   
 $F \subseteq fmla \implies finite\ F \implies \varphi \in fmla \implies \psi \in fmla \implies$   
 $nprv\ F\ (cnj\ \varphi\ \psi)$   
(proof)

**lemma** *nprv\_cnjE*:

$nprv\ F\ (cnj\ \varphi1\ \varphi2) \implies nprv\ (insert\ \varphi1\ (insert\ \varphi2\ F))\ \psi \implies$   
 $F \subseteq fmla \implies finite\ F \implies \varphi1 \in fmla \implies \varphi2 \in fmla \implies \psi \in fmla \implies$   
 $nprv\ F\ \psi$   
(proof)

**lemmas** *nprv\_cnjE0* = *nprv\_cnjE*[*OF nprv\_hyp \_*, *simped*]

**lemmas** *nprv\_cnjE1* = *nprv\_cnjE*[*OF \_ nprv\_hyp*, *simped*]

**lemmas** *nprv\_cnjE01* = *nprv\_cnjE*[*OF nprv\_hyp nprv\_hyp*, *simped*]

**lemma** *nprv\_dsjIL*:

$nprv\ F\ \varphi \implies$   
 $F \subseteq fmla \implies finite\ F \implies \varphi \in fmla \implies \psi \in fmla \implies$   
 $nprv\ F\ (dsj\ \varphi\ \psi)$   
(proof)

**lemma** *nprv\_dsjIR*:

$nprv\ F\ \psi \implies$   
 $F \subseteq fmla \implies finite\ F \implies \varphi \in fmla \implies \psi \in fmla \implies$   
 $nprv\ F\ (dsj\ \varphi\ \psi)$   
(proof)

**lemma** *nprv\_dsjE*:

**assumes**  $nprv\ F\ (dsj\ \varphi\ \psi)$   
**and**  $nprv\ (insert\ \varphi\ F)\ \chi$   $nprv\ (insert\ \psi\ F)\ \chi$   
**and**  $F \subseteq fmla$   $finite\ F$   $\varphi \in fmla$   $\psi \in fmla$   $\chi \in fmla$   
**shows**  $nprv\ F\ \chi$   
(proof)

**lemmas** *nprv\_dsjE0* = *nprv\_dsjE*[*OF nprv\_hyp \_ \_*, *simped*]

**lemmas** *nprv\_dsjE1* = *nprv\_dsjE*[*OF \_ nprv\_hyp \_*, *simped*]

**lemmas** *nprv\_dsjE2* = *nprv\_dsjE*[*OF \_ \_ nprv\_hyp*, *simped*]

**lemmas**  $nprv\_dsjE01 = nprv\_dsjE[OF\ nprv\_hyp\ nprv\_hyp\ \_,\ simped]$   
**lemmas**  $nprv\_dsjE02 = nprv\_dsjE[OF\ nprv\_hyp\ \_ \ nprv\_hyp,\ simped]$   
**lemmas**  $nprv\_dsjE12 = nprv\_dsjE[OF\ \_ \ nprv\_hyp\ nprv\_hyp,\ simped]$   
**lemmas**  $nprv\_dsjE012 = nprv\_dsjE[OF\ nprv\_hyp\ nprv\_hyp\ nprv\_hyp,\ simped]$

**lemma**  $nprv\_flsE: nprv\ F\ fls \implies F \subseteq fmla \implies finite\ F \implies \varphi \in fmla \implies nprv\ F\ \varphi$   
 $\langle proof \rangle$

**lemmas**  $nprv\_flsE0 = nprv\_flsE[OF\ nprv\_hyp,\ simped]$

**lemma**  $nprv\_truI: F \subseteq fmla \implies finite\ F \implies nprv\ F\ tru$   
 $\langle proof \rangle$

**lemma**  $nprv\_negI:$   
 $nprv\ (insert\ \varphi\ F)\ fls \implies$   
 $F \subseteq fmla \implies finite\ F \implies \varphi \in fmla \implies$   
 $nprv\ F\ (neg\ \varphi)$   
 $\langle proof \rangle$

**lemma**  $nprv\_neg\_fls:$   
 $nprv\ F\ (neg\ \varphi) \implies nprv\ F\ \varphi \implies$   
 $F \subseteq fmla \implies finite\ F \implies \varphi \in fmla \implies \psi \in fmla \implies$   
 $nprv\ F\ fls$   
 $\langle proof \rangle$

**lemma**  $nprv\_negE:$   
 $nprv\ F\ (neg\ \varphi) \implies nprv\ F\ \varphi \implies$   
 $F \subseteq fmla \implies finite\ F \implies \varphi \in fmla \implies \psi \in fmla \implies$   
 $nprv\ F\ \psi$   
 $\langle proof \rangle$

**lemmas**  $nprv\_negE0 = nprv\_negE[OF\ nprv\_hyp\ \_,\ simped]$   
**lemmas**  $nprv\_negE1 = nprv\_negE[OF\ \_ \ nprv\_hyp,\ simped]$   
**lemmas**  $nprv\_negE01 = nprv\_negE[OF\ nprv\_hyp\ nprv\_hyp,\ simped]$

**lemma**  $nprv\_scnjI:$   
 $(\bigwedge\ \psi.\ \psi \in G \implies nprv\ F\ \psi) \implies$   
 $F \subseteq fmla \implies finite\ F \implies G \subseteq fmla \implies finite\ G \implies$   
 $nprv\ F\ (scnj\ G)$   
 $\langle proof \rangle$

**lemma**  $nprv\_scnjE:$   
 $nprv\ F\ (scnj\ G) \implies nprv\ (G \cup F)\ \psi \implies$   
 $F \subseteq fmla \implies finite\ F \implies G \subseteq fmla \implies finite\ G \implies \psi \in fmla \implies$   
 $nprv\ F\ \psi$   
 $\langle proof \rangle$

**lemmas**  $nprv\_scnjE0 = nprv\_scnjE[OF\ nprv\_hyp\ \_,\ simped]$   
**lemmas**  $nprv\_scnjE1 = nprv\_scnjE[OF\ \_ \ nprv\_hyp,\ simped]$   
**lemmas**  $nprv\_scnjE01 = nprv\_scnjE[OF\ nprv\_hyp\ nprv\_hyp,\ simped]$

**lemma**  $nprv\_lcnjI:$   
 $(\bigwedge\ \psi.\ \psi \in set\ \psi s \implies nprv\ F\ \psi) \implies$   
 $F \subseteq fmla \implies finite\ F \implies set\ \psi s \subseteq fmla \implies$   
 $nprv\ F\ (lcnj\ \psi s)$   
 $\langle proof \rangle$

**lemma**  $nprv\_lcnjE:$

$nprv\ F\ (lcnj\ \varphi s) \implies nprv\ (set\ \varphi s \cup F)\ \psi \implies$   
 $F \subseteq fmla \implies finite\ F \implies set\ \varphi s \subseteq fmla \implies \psi \in fmla \implies$   
 $nprv\ F\ \psi$   
 <proof>

**lemmas**  $nprv\_lcnjE0 = nprv\_lcnjE[OF\ nprv\_hyp\ \_,\ simped]$   
**lemmas**  $nprv\_lcnjE1 = nprv\_lcnjE[OF\ \_\ nprv\_hyp,\ simped]$   
**lemmas**  $nprv\_lcnjE01 = nprv\_lcnjE[OF\ nprv\_hyp\ nprv\_hyp,\ simped]$

**lemma**  $nprv\_sdsjI$ :  
 $nprv\ F\ \varphi \implies$   
 $F \subseteq fmla \implies finite\ F \implies G \subseteq fmla \implies finite\ G \implies \varphi \in G \implies$   
 $nprv\ F\ (sdsj\ G)$   
 <proof>

**lemma**  $nprv\_sdsjE$ :  
**assumes**  $nprv\ F\ (sdsj\ G)$   
**and**  $\bigwedge\ \psi.\ \psi \in G \implies nprv\ (insert\ \psi\ F)\ \chi$   
**and**  $F \subseteq fmla\ finite\ F\ G \subseteq fmla\ finite\ G\ \chi \in fmla$   
**shows**  $nprv\ F\ \chi$   
 <proof>

**lemmas**  $nprv\_sdsjE0 = nprv\_sdsjE[OF\ nprv\_hyp\ \_,\ simped]$   
**lemmas**  $nprv\_sdsjE1 = nprv\_sdsjE[OF\ \_\ nprv\_hyp,\ simped]$   
**lemmas**  $nprv\_sdsjE01 = nprv\_sdsjE[OF\ nprv\_hyp\ nprv\_hyp,\ simped]$

**lemma**  $nprv\_ldsji$ :  
 $nprv\ F\ \varphi \implies$   
 $F \subseteq fmla \implies finite\ F \implies set\ \varphi s \subseteq fmla \implies \varphi \in set\ \varphi s \implies$   
 $nprv\ F\ (ldsji\ \varphi s)$   
 <proof>

**lemma**  $nprv\_ldsje$ :  
**assumes**  $nprv\ F\ (ldsji\ \psi s)$   
**and**  $\bigwedge\ \psi.\ \psi \in set\ \psi s \implies nprv\ (insert\ \psi\ F)\ \chi$   
**and**  $F \subseteq fmla\ finite\ F\ set\ \psi s \subseteq fmla\ \chi \in fmla$   
**shows**  $nprv\ F\ \chi$   
 <proof>

**lemmas**  $nprv\_ldsje0 = nprv\_ldsje[OF\ nprv\_hyp\ \_,\ simped]$   
**lemmas**  $nprv\_ldsje1 = nprv\_ldsje[OF\ \_\ nprv\_hyp,\ simped]$   
**lemmas**  $nprv\_ldsje01 = nprv\_ldsje[OF\ nprv\_hyp\ nprv\_hyp,\ simped]$

**lemma**  $nprv\_allI$ :  
 $nprv\ F\ \varphi \implies$   
 $F \subseteq fmla \implies finite\ F \implies \varphi \in fmla \implies x \in var \implies x \notin \bigcup\ (Fvars\ 'F) \implies$   
 $nprv\ F\ (all\ x\ \varphi)$   
 <proof>

**lemma**  $nprv\_allE$ :  
**assumes**  $nprv\ F\ (all\ x\ \varphi)\ nprv\ (insert\ (subst\ \varphi\ t\ x)\ F)\ \psi$   
 $F \subseteq fmla\ finite\ F\ \varphi \in fmla\ t \in trm\ x \in var\ \psi \in fmla$   
**shows**  $nprv\ F\ \psi$   
 <proof>

**lemmas**  $nprv\_allE0 = nprv\_allE[OF\ nprv\_hyp\ \_,\ simped]$   
**lemmas**  $nprv\_allE1 = nprv\_allE[OF\ \_\ nprv\_hyp,\ simped]$   
**lemmas**  $nprv\_allE01 = nprv\_allE[OF\ nprv\_hyp\ nprv\_hyp,\ simped]$

**lemma** *nprv\_exiI*:  
*nprv F (subst  $\varphi$  t x)  $\implies$*   
*F  $\subseteq$  fmla  $\implies$  finite F  $\implies$   $\varphi \in$  fmla  $\implies$  t  $\in$  trm  $\implies$  x  $\in$  var  $\implies$*   
*nprv F (exi x  $\varphi$ )*  
*<proof>*

**lemma** *nprv\_exiE*:  
**assumes** *n: nprv F (exi x  $\varphi$ )*  
**and** *nn: nprv (insert  $\varphi$  F)  $\psi$*   
**and** *0[simp]: F  $\subseteq$  fmla finite F  $\varphi \in$  fmla x  $\in$  var  $\psi \in$  fmla*  
**and** *x: x  $\notin$   $\bigcup$  (Fvars ' F) x  $\notin$  Fvars  $\psi$*   
**shows** *nprv F  $\psi$*   
*<proof>*

**lemmas** *nprv\_exiE0 = nprv\_exiE[OF nprv\_hyp \_, simped]*  
**lemmas** *nprv\_exiE1 = nprv\_exiE[OF \_ nprv\_hyp, simped]*  
**lemmas** *nprv\_exiE01 = nprv\_exiE[OF nprv\_hyp nprv\_hyp, simped]*

## 4.7 Adding Lemmas of Various Shapes into the Proof Context

**lemma** *nprv\_addLemmaE*:  
**assumes** *prv  $\varphi$  nprv (insert  $\varphi$  F)  $\psi$*   
**and**  *$\varphi \in$  fmla  $\psi \in$  fmla and F  $\subseteq$  fmla and finite F*  
**shows** *nprv F  $\psi$*   
*<proof>*

**lemmas** *nprv\_addLemmaE1 = nprv\_addLemmaE[OF \_ nprv\_hyp, simped]*

**lemma** *nprv\_addImpLemmaI*:  
**assumes** *prv (imp  $\varphi$ 1  $\varphi$ 2)*  
**and** *F  $\subseteq$  fmla finite F  $\varphi$ 1  $\in$  fmla  $\varphi$ 2  $\in$  fmla*  
**and** *nprv F  $\varphi$ 1*  
**shows** *nprv F  $\varphi$ 2*  
*<proof>*

**lemma** *nprv\_addImpLemmaE*:  
**assumes** *prv (imp  $\varphi$ 1  $\varphi$ 2) and nprv F  $\varphi$ 1 and nprv ((insert  $\varphi$ 2) F)  $\psi$*   
**and** *F  $\subseteq$  fmla finite F  $\varphi$ 1  $\in$  fmla  $\varphi$ 2  $\in$  fmla  $\psi \in$  fmla*  
**shows** *nprv F  $\psi$*   
*<proof>*

**lemmas** *nprv\_addImpLemmaE1 = nprv\_addImpLemmaE[OF \_ nprv\_hyp \_, simped]*  
**lemmas** *nprv\_addImpLemmaE2 = nprv\_addImpLemmaE[OF \_ \_ nprv\_hyp, simped]*  
**lemmas** *nprv\_addImpLemmaE12 = nprv\_addImpLemmaE[OF \_ nprv\_hyp nprv\_hyp, simped]*

**lemma** *nprv\_addImp2LemmaI*:  
**assumes** *prv (imp  $\varphi$ 1 (imp  $\varphi$ 2  $\varphi$ 3))*  
**and** *F  $\subseteq$  fmla finite F  $\varphi$ 1  $\in$  fmla  $\varphi$ 2  $\in$  fmla  $\varphi$ 3  $\in$  fmla*  
**and** *nprv F  $\varphi$ 1 nprv F  $\varphi$ 2*  
**shows** *nprv F  $\varphi$ 3*  
*<proof>*

**lemma** *nprv\_addImp2LemmaE*:  
**assumes** *prv (imp  $\varphi$ 1 (imp  $\varphi$ 2  $\varphi$ 3)) and nprv F  $\varphi$ 1 and nprv F  $\varphi$ 2 and nprv ((insert  $\varphi$ 3) F)  $\psi$*   
**and** *F  $\subseteq$  fmla finite F  $\varphi$ 1  $\in$  fmla  $\varphi$ 2  $\in$  fmla  $\varphi$ 3  $\in$  fmla  $\psi \in$  fmla*

**shows**  $nprv\ F\ \psi$   
(*proof*)

**lemmas**  $nprv\_addImp2LemmaE1 = nprv\_addImp2LemmaE[OF\ \_ \ nprv\_hyp\ \_ \_,\ simped]$   
**lemmas**  $nprv\_addImp2LemmaE2 = nprv\_addImp2LemmaE[OF\ \_ \_ \ nprv\_hyp\ \_,\ simped]$   
**lemmas**  $nprv\_addImp2LemmaE3 = nprv\_addImp2LemmaE[OF\ \_ \_ \_ \ nprv\_hyp,\ simped]$   
**lemmas**  $nprv\_addImp2LemmaE12 = nprv\_addImp2LemmaE[OF\ \_ \ nprv\_hyp\ nprv\_hyp\ \_,\ simped]$   
**lemmas**  $nprv\_addImp2LemmaE13 = nprv\_addImp2LemmaE[OF\ \_ \ nprv\_hyp\ \_ \ nprv\_hyp,\ simped]$   
**lemmas**  $nprv\_addImp2LemmaE23 = nprv\_addImp2LemmaE[OF\ \_ \_ \ nprv\_hyp\ nprv\_hyp,\ simped]$   
**lemmas**  $nprv\_addImp2LemmaE123 = nprv\_addImp2LemmaE[OF\ \_ \ nprv\_hyp\ nprv\_hyp\ nprv\_hyp,\ simped]$

**lemma**  $nprv\_addImp3LemmaI$ :  
**assumes**  $prv\ (imp\ \varphi1\ (imp\ \varphi2\ (imp\ \varphi3\ \varphi4)))$   
**and**  $F \subseteq fmla\ finite\ F\ \varphi1 \in fmla\ \varphi2 \in fmla\ \varphi3 \in fmla\ \varphi4 \in fmla$   
**and**  $nprv\ F\ \varphi1\ nprv\ F\ \varphi2\ nprv\ F\ \varphi3$   
**shows**  $nprv\ F\ \varphi4$   
(*proof*)

**lemma**  $nprv\_addImp3LemmaE$ :  
**assumes**  $prv\ (imp\ \varphi1\ (imp\ \varphi2\ (imp\ \varphi3\ \varphi4)))$  **and**  $nprv\ F\ \varphi1$  **and**  $nprv\ F\ \varphi2$  **and**  $nprv\ F\ \varphi3$   
**and**  $nprv\ ((insert\ \varphi4)\ F)\ \psi$   
**and**  $F \subseteq fmla\ finite\ F\ \varphi1 \in fmla\ \varphi2 \in fmla\ \varphi3 \in fmla\ \varphi4 \in fmla\ \psi \in fmla$   
**shows**  $nprv\ F\ \psi$   
(*proof*)

**lemmas**  $nprv\_addImp3LemmaE1 = nprv\_addImp3LemmaE[OF\ \_ \ nprv\_hyp\ \_ \_ \_,\ simped]$   
**lemmas**  $nprv\_addImp3LemmaE2 = nprv\_addImp3LemmaE[OF\ \_ \_ \_ \ nprv\_hyp\ \_ \_,\ simped]$   
**lemmas**  $nprv\_addImp3LemmaE3 = nprv\_addImp3LemmaE[OF\ \_ \_ \_ \_ \ nprv\_hyp\ \_,\ simped]$   
**lemmas**  $nprv\_addImp3LemmaE4 = nprv\_addImp3LemmaE[OF\ \_ \_ \_ \_ \_ \_ \ nprv\_hyp,\ simped]$   
**lemmas**  $nprv\_addImp3LemmaE12 = nprv\_addImp3LemmaE[OF\ \_ \ nprv\_hyp\ nprv\_hyp\ \_ \_,\ simped]$   
**lemmas**  $nprv\_addImp3LemmaE13 = nprv\_addImp3LemmaE[OF\ \_ \ nprv\_hyp\ \_ \ nprv\_hyp\ \_,\ simped]$   
**lemmas**  $nprv\_addImp3LemmaE14 = nprv\_addImp3LemmaE[OF\ \_ \ nprv\_hyp\ \_ \_ \_ \ nprv\_hyp,\ simped]$   
**lemmas**  $nprv\_addImp3LemmaE23 = nprv\_addImp3LemmaE[OF\ \_ \_ \_ \ nprv\_hyp\ nprv\_hyp\ \_,\ simped]$   
**lemmas**  $nprv\_addImp3LemmaE24 = nprv\_addImp3LemmaE[OF\ \_ \_ \_ \ nprv\_hyp\ \_ \ nprv\_hyp,\ simped]$   
**lemmas**  $nprv\_addImp3LemmaE34 = nprv\_addImp3LemmaE[OF\ \_ \_ \_ \_ \ nprv\_hyp\ nprv\_hyp,\ simped]$   
**lemmas**  $nprv\_addImp3LemmaE123 = nprv\_addImp3LemmaE[OF\ \_ \ nprv\_hyp\ nprv\_hyp\ nprv\_hyp\ \_,\ simped]$   
**lemmas**  $nprv\_addImp3LemmaE124 = nprv\_addImp3LemmaE[OF\ \_ \ nprv\_hyp\ nprv\_hyp\ \_ \ nprv\_hyp,\ simped]$   
**lemmas**  $nprv\_addImp3LemmaE134 = nprv\_addImp3LemmaE[OF\ \_ \ nprv\_hyp\ \_ \ nprv\_hyp\ nprv\_hyp,\ simped]$   
**lemmas**  $nprv\_addImp3LemmaE234 = nprv\_addImp3LemmaE[OF\ \_ \_ \_ \ nprv\_hyp\ nprv\_hyp\ nprv\_hyp,\ simped]$   
**lemmas**  $nprv\_addImp3LemmaE1234 = nprv\_addImp3LemmaE[OF\ \_ \ nprv\_hyp\ nprv\_hyp\ nprv\_hyp\ nprv\_hyp,\ simped]$

**lemma**  $nprv\_addDsjLemmaE$ :  
**assumes**  $prv\ (dsj\ \varphi1\ \varphi2)$  **and**  $nprv\ (insert\ \varphi1\ F)\ \psi$  **and**  $nprv\ ((insert\ \varphi2)\ F)\ \psi$   
**and**  $F \subseteq fmla\ finite\ F\ \varphi1 \in fmla\ \varphi2 \in fmla\ \psi \in fmla$   
**shows**  $nprv\ F\ \psi$   
(*proof*)

**lemmas**  $nprv\_addDsjLemmaE1 = nprv\_addDsjLemmaE[OF\ \_ \ nprv\_hyp\ \_,\ simped]$   
**lemmas**  $nprv\_addDsjLemmaE2 = nprv\_addDsjLemmaE[OF\ \_ \_ \_ \ nprv\_hyp,\ simped]$   
**lemmas**  $nprv\_addDsjLemmaE12 = nprv\_addDsjLemmaE[OF\ \_ \ nprv\_hyp\ nprv\_hyp,\ simped]$

## 4.8 Rules for Equality

Reflexivity:

**lemma** *nprv\_eq\_reflI*:  $F \subseteq \text{fmla} \implies \text{finite } F \implies t \in \text{trm} \implies \text{nprv } F \text{ (eql } t \text{ } t)$   
*<proof>*

**lemma** *nprv\_eq\_eqlI*:  $t1 = t2 \implies F \subseteq \text{fmla} \implies \text{finite } F \implies t1 \in \text{trm} \implies \text{nprv } F \text{ (eql } t1 \text{ } t2)$   
*<proof>*

Symmetry:

**lemmas** *nprv\_eq\_symI* = *nprv\_addImpLemmaI*[*OF prv\_eq\_sym, simped, rotated 4*]  
**lemmas** *nprv\_eq\_symE* = *nprv\_addImpLemmaE*[*OF prv\_eq\_sym, simped, rotated 2*]

**lemmas** *nprv\_eq\_symE0* = *nprv\_eq\_symE*[*OF nprv\_hyp \_, simped*]  
**lemmas** *nprv\_eq\_symE1* = *nprv\_eq\_symE*[*OF \_ nprv\_hyp, simped*]  
**lemmas** *nprv\_eq\_symE01* = *nprv\_eq\_symE*[*OF nprv\_hyp nprv\_hyp, simped*]

Transitivity:

**lemmas** *nprv\_eq\_transI* = *nprv\_addImp2LemmaI*[*OF prv\_eq\_imp\_trans, simped, rotated 5*]  
**lemmas** *nprv\_eq\_transE* = *nprv\_addImp2LemmaE*[*OF prv\_eq\_imp\_trans, simped, rotated 3*]

**lemmas** *nprv\_eq\_transE0* = *nprv\_eq\_transE*[*OF nprv\_hyp \_ \_, simped*]  
**lemmas** *nprv\_eq\_transE1* = *nprv\_eq\_transE*[*OF \_ nprv\_hyp \_, simped*]  
**lemmas** *nprv\_eq\_transE2* = *nprv\_eq\_transE*[*OF \_ \_ nprv\_hyp, simped*]  
**lemmas** *nprv\_eq\_transE01* = *nprv\_eq\_transE*[*OF nprv\_hyp nprv\_hyp \_, simped*]  
**lemmas** *nprv\_eq\_transE02* = *nprv\_eq\_transE*[*OF nprv\_hyp \_ nprv\_hyp, simped*]  
**lemmas** *nprv\_eq\_transE12* = *nprv\_eq\_transE*[*OF \_ nprv\_hyp nprv\_hyp, simped*]  
**lemmas** *nprv\_eq\_transE012* = *nprv\_eq\_transE*[*OF nprv\_hyp nprv\_hyp nprv\_hyp, simped*]

Substitutivity:

**lemmas** *nprv\_eq\_substI* =  
*nprv\_addImp2LemmaI*[*OF prv\_eq\_subst\_trm\_rev, simped, rotated 6*]  
**lemmas** *nprv\_eq\_substE* = *nprv\_addImp2LemmaE*[*OF prv\_eq\_subst\_trm\_rev, simped, rotated 4*]

**lemmas** *nprv\_eq\_substE0* = *nprv\_eq\_substE*[*OF nprv\_hyp \_ \_, simped*]  
**lemmas** *nprv\_eq\_substE1* = *nprv\_eq\_substE*[*OF \_ nprv\_hyp \_, simped*]  
**lemmas** *nprv\_eq\_substE2* = *nprv\_eq\_substE*[*OF \_ \_ nprv\_hyp, simped*]  
**lemmas** *nprv\_eq\_substE01* = *nprv\_eq\_substE*[*OF nprv\_hyp nprv\_hyp \_, simped*]  
**lemmas** *nprv\_eq\_substE02* = *nprv\_eq\_substE*[*OF nprv\_hyp \_ nprv\_hyp, simped*]  
**lemmas** *nprv\_eq\_substE12* = *nprv\_eq\_substE*[*OF \_ nprv\_hyp nprv\_hyp, simped*]  
**lemmas** *nprv\_eq\_substE012* = *nprv\_eq\_substE*[*OF nprv\_hyp nprv\_hyp nprv\_hyp, simped*]

## 4.9 Other Rules

**lemma** *nprv\_cnjH*:  
*nprv (insert  $\varphi1$  (insert  $\varphi2$   $F$ ))  $\psi \implies$   
 $F \subseteq \text{fmla} \implies \text{finite } F \implies \varphi1 \in \text{fmla} \implies \varphi2 \in \text{fmla} \implies \psi \in \text{fmla} \implies$   
*nprv (insert (cnj  $\varphi1$   $\varphi2$ )  $F$ )  $\psi$*   
*<proof>**

**lemma** *nprv\_exi\_commute*:  
**assumes** [*simp*]:  $x \in \text{var } y \in \text{var } \varphi \in \text{fmla}$   
**shows** *nprv {exi  $x$  (exi  $y$   $\varphi$ )} (exi  $y$  (exi  $x$   $\varphi$ ))*  
*<proof>*

**lemma** *prv\_exi\_commute*:

**assumes**  $[simp]: x \in var\ y \in var\ \varphi \in fmla$   
**shows**  $prv\ (imp\ (exi\ x\ (exi\ y\ \varphi))\ (exi\ y\ (exi\ x\ \varphi)))$   
 $\langle proof \rangle$

**end**

## 4.10 Natural Deduction for the Exists-Unique Quantifier

**context**  $Deduct\_with\_False\_Disj\_Rename$   
**begin**

**lemma**  $nprv\_exuI$ :

**assumes**  $n1: nprv\ F\ (subst\ \varphi\ t\ x)$  **and**  $n2: nprv\ (insert\ \varphi\ F)\ (eq\ (Var\ x)\ t)$   
**and**  $i[simp]: F \subseteq fmla\ finite\ F\ \varphi \in fmla\ t \in trm\ x \in var\ x \notin FvarsT\ t$   
**and**  $u: x \notin (\bigcup \varphi \in F. Fvars\ \varphi)$   
**shows**  $nprv\ F\ (exu\ x\ \varphi)$   
 $\langle proof \rangle$

**lemma**  $nprv\_exuI\_var$ :

**assumes**  $n1: nprv\ F\ (subst\ \varphi\ t\ x)$  **and**  $n2: nprv\ (insert\ (subst\ \varphi\ (Var\ y)\ x)\ F)\ (eq\ (Var\ y)\ t)$   
**and**  $i[simp]: F \subseteq fmla\ finite\ F\ \varphi \in fmla\ t \in trm\ x \in var$   
 $y \in var\ y \notin FvarsT\ t$  **and**  $u: y \notin (\bigcup \varphi \in F. Fvars\ \varphi)$  **and**  $yx: y = x \vee y \notin Fvars\ \varphi$   
**shows**  $nprv\ F\ (exu\ x\ \varphi)$   
 $\langle proof \rangle$

This turned out to be the most useful introduction rule for arithmetic:

**lemma**  $nprv\_exuI\_exi$ :

**assumes**  $n1: nprv\ F\ (exi\ x\ \varphi)$  **and**  $n2: nprv\ (insert\ (subst\ \varphi\ (Var\ y)\ x)\ (insert\ \varphi\ F))\ (eq\ (Var\ y)\ (Var\ x))$   
**and**  $i[simp]: F \subseteq fmla\ finite\ F\ \varphi \in fmla\ x \in var\ y \in var\ y \neq x\ y \notin Fvars\ \varphi$   
**and**  $u: x \notin (\bigcup \varphi \in F. Fvars\ \varphi)\ y \notin (\bigcup \varphi \in F. Fvars\ \varphi)$   
**shows**  $nprv\ F\ (exu\ x\ \varphi)$   
 $\langle proof \rangle$

**lemma**  $prv\_exu\_imp\_exi$ :

**assumes**  $[simp]: \varphi \in fmla\ x \in var$   
**shows**  $prv\ (imp\ (exu\ x\ \varphi)\ (exi\ x\ \varphi))$   
 $\langle proof \rangle$

**lemma**  $prv\_exu\_exi$ :

**assumes**  $x \in var\ \varphi \in fmla\ prv\ (exu\ x\ \varphi)$   
**shows**  $prv\ (exi\ x\ \varphi)$   
 $\langle proof \rangle$

This is just  $exu$  behaving for elimination and forward like  $exi$ :

**lemma**  $nprv\_exuE\_exi$ :

**assumes**  $n1: nprv\ F\ (exu\ x\ \varphi)$  **and**  $n2: nprv\ (insert\ \varphi\ F)\ \psi$   
**and**  $i[simp]: F \subseteq fmla\ finite\ F\ \varphi \in fmla\ x \in var\ \psi \in fmla\ x \notin Fvars\ \psi$   
**and**  $u: x \notin (\bigcup \varphi \in F. Fvars\ \varphi)$   
**shows**  $nprv\ F\ \psi$   
 $\langle proof \rangle$

**lemma**  $nprv\_exuF\_exi$ :

**assumes**  $n1: exu\ x\ \varphi \in F$  **and**  $n2: nprv\ (insert\ \varphi\ F)\ \psi$   
**and**  $i[simp]: F \subseteq fmla\ finite\ F\ \varphi \in fmla\ x \in var\ \psi \in fmla\ x \notin Fvars\ \psi$   
**and**  $u: x \notin (\bigcup \varphi \in F. Fvars\ \varphi)$   
**shows**  $nprv\ F\ \psi$

*<proof>*

**lemma** *prv\_exu\_uni*:

**assumes** [*simp*]:  $\varphi \in \text{fm}la$   $x \in \text{var}$   $t1 \in \text{trm}$   $t2 \in \text{trm}$

**shows** *prv* (*imp* (*exu*  $x$   $\varphi$ ) (*imp* (*subst*  $\varphi$   $t1$   $x$ ) (*imp* (*subst*  $\varphi$   $t2$   $x$ ) (*eql*  $t1$   $t2$ ))))

*<proof>*

**lemmas** *nprv\_exuE\_uni* = *nprv\_addImp3LemmaE*[*OF prv\_exu\_uni,simped,rotated 4*]

**lemmas** *nprv\_exuF\_uni* = *nprv\_exuE\_uni*[*OF nprv\_hyp,simped*]

**end** — context *Deduct\_with\_False\_Disj*

## 4.11 Eisbach Notation for Natural Deduction Proofs

The proof pattern will be: On a goal of the form *nprv*  $F$   $\varphi$ , we apply a rule (usually an introduction, elimination, or cut/lemma-addition rule), then discharge the side-conditions with *auto*, ending up with zero, one or two goals of the same *nprv*-shape. This process is abstracted away in the Eisbach *nrule* method:

**method** *nrule* **uses**  $r = (\text{rule } r, \text{auto}?)$

**method** *nrule2* **uses**  $r = (\text{rule } r, \text{auto}?)$

Methods for chaining several *nrule* applications:

**method** *nprover2* **uses**  $r1$   $r2 =$

$(-,(((\text{nrule } r: r1)?, (\text{nrule } r: r2)?); \text{fail}))$

**method** *nprover3* **uses**  $r1$   $r2$   $r3 =$

$(-,(((\text{nrule } r: r1)?, (\text{nrule } r: r2)?, (\text{nrule } r: r3)?); \text{fail}))$

**method** *nprover4* **uses**  $r1$   $r2$   $r3$   $r4 =$

$(-,(((\text{nrule } r: r1)?, (\text{nrule } r: r2)?, (\text{nrule } r: r3)?, (\text{nrule } r: r4)?); \text{fail}))$

**method** *nprover5* **uses**  $r1$   $r2$   $r3$   $r4$   $r5 =$

$(-,((\text{nrule } r: r1)?, (\text{nrule } r: r2)?, (\text{nrule } r: r3)?,$

$(\text{nrule } r: r4)?, (\text{nrule } r: r5)?); \text{fail})$

**method** *nprover6* **uses**  $r1$   $r2$   $r3$   $r4$   $r5$   $r6 =$

$(-,((\text{nrule } r: r1)?, (\text{nrule } r: r2)?, (\text{nrule } r: r3)?,$

$(\text{nrule } r: r4)?, (\text{nrule } r: r5)?, (\text{nrule } r: r6)?); \text{fail})$



# Chapter 5

## Pseudo-Terms

Pseudo-terms are formulas that satisfy the exists-unique property on one of their variables.

### 5.1 Basic Setting

**context** *Generic\_Syntax*  
**begin**

We choose a specific variable, *out*, that will represent the "output" of pseudo-terms, i.e., the variable on which the exists-unique property holds:

**abbreviation**  $out \equiv Variable\ 0$

Many facts will involve pseudo-terms with only one additional "input" variable, *inp*:

**abbreviation**  $inp \equiv Variable\ (Suc\ 0)$

**lemma** *out\_inp\_distinct[simp]*:

$out \neq inp\ inp \neq out$

$out \neq xx\ out \neq yy\ yy \neq out\ out \neq zz\ zz \neq out\ out \neq xx'\ xx' \neq out$

$out \neq yy'\ yy' \neq out\ out \neq zz'\ zz' \neq out$

$inp \neq xx\ inp \neq yy\ yy \neq inp\ inp \neq zz\ zz \neq inp\ inp \neq xx'\ xx' \neq inp$

$inp \neq yy'\ yy' \neq inp\ inp \neq zz'\ zz' \neq inp$

*<proof>*

**end**

**context** *Deduct\_with\_False\_Disj\_Rename*  
**begin**

Pseudo-terms over the first  $n + 1$  variables, i.e., having  $n$  input variables (Variable 1 to Variable  $n$ ), and an output variable, *out* (which is an abbreviation for Variable 0).

**definition** *ptrm* ::  $nat \Rightarrow 'fmla\ set$  **where**

$ptrm\ n \equiv \{\sigma \in fmla . Fvars\ \sigma = Variable\ ' \{0..n\} \wedge prv\ (exu\ out\ \sigma)\}$

**lemma** *ptrm[intro,simp]*:  $\sigma \in ptrm\ n \implies \sigma \in fmla$

*<proof>*

**lemma** *ptrm\_1\_Fvars[simp]*:  $\sigma \in ptrm\ (Suc\ 0) \implies Fvars\ \sigma = \{out,inp\}$

*<proof>*

**lemma** *ptrm\_prv\_exu*:  $\sigma \in ptrm\ n \implies prv\ (exu\ out\ \sigma)$   
 ⟨proof⟩

**lemma** *ptrm\_prv\_exi*:  $\sigma \in ptrm\ n \implies prv\ (exi\ out\ \sigma)$   
 ⟨proof⟩

**lemma** *nprv\_ptrmE\_exi*:  
 $\sigma \in ptrm\ n \implies nprv\ (insert\ \sigma\ F)\ \psi \implies$   
 $F \subseteq fmla \implies finite\ F \implies$   
 $\psi \in fmla \implies out \notin Fvars\ \psi \implies out \notin \bigcup (Fvars\ 'F) \implies nprv\ F\ \psi$   
 ⟨proof⟩

**lemma** *nprv\_ptrmE\_uni*:  
 $\sigma \in ptrm\ n \implies nprv\ F\ (subst\ \sigma\ t1\ out) \implies nprv\ F\ (subst\ \sigma\ t2\ out) \implies$   
 $nprv\ (insert\ (eql\ t1\ t2)\ F)\ \psi \implies$   
 $F \subseteq fmla \implies finite\ F \implies \psi \in fmla \implies t1 \in trm \implies t2 \in trm$   
 $\implies nprv\ F\ \psi$   
 ⟨proof⟩

**lemma** *nprv\_ptrmE\_uni0*:  
 $\sigma \in ptrm\ n \implies nprv\ F\ \sigma \implies nprv\ F\ (subst\ \sigma\ t\ out) \implies$   
 $nprv\ (insert\ (eql\ (Var\ out)\ t)\ F)\ \psi \implies$   
 $F \subseteq fmla \implies finite\ F \implies \psi \in fmla \implies t \in trm$   
 $\implies nprv\ F\ \psi$   
 ⟨proof⟩

**lemma** *nprv\_ptrmE0\_uni0*:  
 $\sigma \in ptrm\ n \implies \sigma \in F \implies nprv\ F\ (subst\ \sigma\ t\ out) \implies$   
 $nprv\ (insert\ (eql\ (Var\ out)\ t)\ F)\ \psi \implies$   
 $F \subseteq fmla \implies finite\ F \implies \psi \in fmla \implies t \in trm$   
 $\implies nprv\ F\ \psi$   
 ⟨proof⟩

## 5.2 The $\forall$ - $\exists$ Equivalence

There are two natural ways to state that (unique) "output" of a pseudo-term  $\sigma$  satisfies a property  $\varphi$ : (1) using  $\exists$ : there exists an "out" such that  $\sigma$  and  $\varphi$  hold for it; (2) using  $\forall$ : for all "out" such that  $\sigma$  holds for it,  $\varphi$  holds for it as well.

We prove the well-known fact that these two ways are equivalent. (Intuitionistic logic suffice to prove that.)

**lemma** *ptrm\_nprv\_exi*:  
**assumes**  $\sigma: \sigma \in ptrm\ n$  **and**  $[simp]: \varphi \in fmla$   
**shows**  $nprv\ \{\sigma, exi\ out\ (cnj\ \sigma\ \varphi)\} \varphi$   
 ⟨proof⟩

**lemma** *ptrm\_nprv\_exi\_all*:  
**assumes**  $\sigma: \sigma \in ptrm\ n$  **and**  $[simp]: \varphi \in fmla$   
**shows**  $nprv\ \{exi\ out\ (cnj\ \sigma\ \varphi)\} (all\ out\ (imp\ \sigma\ \varphi))$   
 ⟨proof⟩

**lemma** *ptrm\_prv\_exi\_imp\_all*:  
**assumes**  $\sigma: \sigma \in ptrm\ n$  **and**  $[simp]: \varphi \in fmla$   
**shows**  $prv\ (imp\ (exi\ out\ (cnj\ \sigma\ \varphi))\ (all\ out\ (imp\ \sigma\ \varphi)))$   
 ⟨proof⟩

**lemma** *ptrm\_nprv\_all\_imp\_exi*:

**assumes**  $\sigma: \sigma \in ptrm\ n$  **and**  $[simp]: \varphi \in fmla$   
**shows**  $nprv\ \{all\ out\ (imp\ \sigma\ \varphi)\ (exi\ out\ (cnj\ \sigma\ \varphi))\}$   
 $\langle proof \rangle$

**lemma**  $ptrm\_prv\_all\_imp\_exi$ :  
**assumes**  $\sigma: \sigma \in ptrm\ n$  **and**  $[simp]: \varphi \in fmla$   
**shows**  $prv\ (imp\ (all\ out\ (imp\ \sigma\ \varphi))\ (exi\ out\ (cnj\ \sigma\ \varphi)))$   
 $\langle proof \rangle$

**end** — context  $Deduct\_with\_False\_Disj\_Rename$

## 5.3 Instantiation

We define the notion of instantiating the "inp" variable of a formula (in particular, of a pseudo-term): – first with a term; – then with a pseudo-term.

### 5.3.1 Instantiation with terms

Instantiation with terms is straightforward using substitution. In the name of the operator, the suffix "Inp" is a reminder that we instantiate  $\varphi$  on its variable "inp".

**context**  $Generic\_Syntax$   
**begin**

**definition**  $instInp :: 'fmla \Rightarrow 'trm \Rightarrow 'fmla$  **where**  
 $instInp\ \varphi\ t \equiv subst\ \varphi\ t\ inp$

**lemma**  $instInp\_fmla[simp,intro]$ :  
**assumes**  $\varphi \in fmla$  **and**  $t \in trm$   
**shows**  $instInp\ \varphi\ t \in fmla$   
 $\langle proof \rangle$

**lemma**  $Fvars\_instInp[simp,intro]$ :  
**assumes**  $\varphi \in fmla$  **and**  $t \in trm$   $Fvars\ \varphi = \{inp\}$   
**shows**  $Fvars\ (instInp\ \varphi\ t) = FvarsT\ t$   
 $\langle proof \rangle$

**end** — context  $Generic\_Syntax$

**context**  $Deduct\_with\_False\_Disj\_Rename$   
**begin**

**lemma**  $Fvars\_instInp\_ptrm\_1[simp,intro]$ :  
**assumes**  $\tau: \tau \in ptrm\ (Suc\ 0)$  **and**  $t \in trm$   
**shows**  $Fvars\ (instInp\ \tau\ t) = insert\ out\ (FvarsT\ t)$   
 $\langle proof \rangle$

**lemma**  $instInp$ :  
**assumes**  $\tau: \tau \in ptrm\ (Suc\ 0)$  **and**  $[simp]: t \in trm$   
**and**  $[simp]: FvarsT\ t = Variable\ '\{(Suc\ 0)..n\}$   
**shows**  $instInp\ \tau\ t \in ptrm\ n$   
 $\langle proof \rangle$

**lemma**  $instInp\_0$ :  
**assumes**  $\tau: \tau \in ptrm\ (Suc\ 0)$  **and**  $t \in trm$  **and**  $FvarsT\ t = \{\}$   
**shows**  $instInp\ \tau\ t \in ptrm\ 0$

*<proof>*

**lemma** *instInp\_1*:

**assumes**  $\tau: \tau \in ptrm (Suc\ 0)$  **and**  $t \in trm$  **and**  $FvarsT\ t = \{inp\}$

**shows**  $instInp\ \tau\ t \in ptrm (Suc\ 0)$

*<proof>*

### 5.3.2 Instantiation with pseudo-terms

Instantiation of a formula  $\varphi$  with a pseudo-term  $\tau$  yields a formula that could be casually written  $\varphi(\tau)$ . It states the existence of an output  $zz$  of  $\tau$  on which  $\varphi$  holds. Instead of  $\varphi(\tau)$ , we write  $instInpP\ \varphi\ n\ \tau$  where  $n$  is the number of input variables of  $\tau$ . In the name *instInpP*, *Inp* is as before a reminder that we instantiate  $\varphi$  on its variable "inp" and the suffix "P" stands for "Pseudo".

**definition** *instInpP* ::  $'fmla \Rightarrow nat \Rightarrow 'fmla \Rightarrow 'fmla$  **where**  
 $instInpP\ \varphi\ n\ \tau \equiv let\ zz = Variable\ (Suc\ (Suc\ n))\ in$   
 $\quad cxi\ zz\ (cnj\ (subst\ \tau\ (Var\ zz)\ out)\ (subst\ \varphi\ (Var\ zz)\ inp))$

**lemma** *instInpP\_fmla[simp, intro]*:

**assumes**  $\varphi \in fmla$  **and**  $\tau \in fmla$

**shows**  $instInpP\ \varphi\ n\ \tau \in fmla$

*<proof>*

**lemma** *Fvars\_instInpP[simp]*:

**assumes**  $\varphi \in fmla$  **and**  $\tau: \tau \in ptrm\ n$   $Variable\ (Suc\ (Suc\ n)) \notin Fvars\ \varphi$

**shows**  $Fvars\ (instInpP\ \varphi\ n\ \tau) = Fvars\ \varphi - \{inp\} \cup Variable\ '\{(Suc\ 0)..n\}$

*<proof>*

**lemma** *Fvars\_instInpP2[simp]*:

**assumes**  $\varphi \in fmla$  **and**  $\tau: \tau \in ptrm\ n$  **and**  $Fvars\ \varphi \subseteq \{inp\}$

**shows**  $Fvars\ (instInpP\ \varphi\ n\ \tau) = Fvars\ \varphi - \{inp\} \cup Variable\ '\{(Suc\ 0)..n\}$

*<proof>*

### 5.3.3 Closure and compositionality properties of instantiation

Instantiating a 1-pseudo-term with an n-pseudo-term yields an n pseudo-term:

**lemma** *instInpP1[simp,intro]*:

**assumes**  $\sigma: \sigma \in ptrm (Suc\ 0)$  **and**  $\tau: \tau \in ptrm\ n$

**shows**  $instInpP\ \sigma\ n\ \tau \in ptrm\ n$

*<proof>*

Term and pseudo-term instantiation compose smoothly:

**lemma** *instInp\_instInpP*:

**assumes**  $\varphi: \varphi \in fmla$   $Fvars\ \varphi \subseteq \{inp\}$  **and**  $\tau: \tau \in ptrm (Suc\ 0)$

**and**  $t \in trm$  **and**  $FvarsT\ t = \{\}$

**shows**  $instInp\ (instInpP\ \varphi\ (Suc\ 0)\ \tau)\ t = instInpP\ \varphi\ 0\ (instInp\ \tau\ t)$

*<proof>*

Pseudo-term instantiation also composes smoothly with itself:

**lemma** *nprv\_instInpP\_compose*:

**assumes** *[simp]*:  $\chi \in fmla$   $Fvars\ \chi = \{inp\}$

**and**  $\sigma$  *[simp]*:  $\sigma \in ptrm (Suc\ 0)$  **and**  $\tau$  *[simp]*:  $\tau \in ptrm\ 0$

**shows**  $nprv\ \{instInpP\ (instInpP\ \chi\ (Suc\ 0)\ \sigma)\ 0\ \tau\}$

$(instInpP\ \chi\ 0\ (instInpP\ \sigma\ 0\ \tau))$  **(is ?A)**

**and**

$nprv\ \{instInpP\ \chi\ 0\ (instInpP\ \sigma\ 0\ \tau)\}$

$(instInpP\ (instInpP\ \chi\ (Suc\ 0)\ \sigma)\ 0\ \tau)$  **(is ?B)**

*<proof>*

**lemma** *prv\_instInpP\_compose*:

**assumes** *[simp]*:  $\chi \in \text{fm}la$   $Fvars \chi = \{inp\}$   
**and**  $\sigma$  *[simp]*:  $\sigma \in \text{ptrm} (Suc\ 0)$  **and**  $\tau$  *[simp]*:  $\tau \in \text{ptrm}\ 0$   
**shows** *prv* (*imp* (*instInpP* (*instInpP*  $\chi$  (*Suc* 0)  $\sigma$ ) 0)  $\tau$ )  
          (*instInpP*  $\chi$  0 (*instInpP*  $\sigma$  0)  $\tau$ )) (**is** ?A)

**and**

*prv* (*imp* (*instInpP*  $\chi$  0 (*instInpP*  $\sigma$  0)  $\tau$ )  
          (*instInpP* (*instInpP*  $\chi$  (*Suc* 0)  $\sigma$ ) 0)  $\tau$ )) (**is** ?B)

**and**

*prv* (*eqv* (*instInpP* (*instInpP*  $\chi$  (*Suc* 0)  $\sigma$ ) 0)  $\tau$ )  
          (*instInpP*  $\chi$  0 (*instInpP*  $\sigma$  0)  $\tau$ )) (**is** ?C)

*<proof>*

## 5.4 Equality between Pseudo-Terms and Terms

Casually, the equality between a pseudo-term  $\tau$  and a term  $t$  can be written as  $\vdash \tau = t$ . This is in fact the (provability of) the instantiation of  $\tau$  with  $t$  on  $\tau$ 's output variable out. Indeed, this formula says that the unique entity denoted by  $\tau$  is exactly  $t$ .

**definition** *prveqLPT* ::  $'fm}la \Rightarrow 'trm \Rightarrow bool$  **where**  
*prveqLPT*  $\tau$   $t \equiv prv$  (*subst*  $\tau$   $t$  out)

We prove that term–pseudo-term equality indeed acts like an equality, in that it satisfies the substitutivity principle (shown only in the particular case of formula-input instantiation).

**lemma** *prveqLPT\_nprv\_instInp\_instInpP*:

**assumes** *[simp]*:  $\varphi \in \text{fm}la$  **and**  $f$ :  $Fvars \varphi \subseteq \{inp\}$  **and**  $\tau$ :  $\tau \in \text{ptrm}\ 0$   
**and** *[simp]*:  $t \in \text{trm}$   $FvarsT\ t = \{\}$   
**and**  $\tau t$ : *prveqLPT*  $\tau$   $t$   
**shows** *nprv* {*instInpP*  $\varphi$  0}  $\tau$ ) (*instInpP*  $\varphi$   $t$ )  
*<proof>*

**lemma** *prveqLPT\_prv\_instInp\_instInpP*:

**assumes**  $\varphi \in \text{fm}la$  **and**  $f$ :  $Fvars \varphi \subseteq \{inp\}$  **and**  $\tau$ :  $\tau \in \text{ptrm}\ 0$   
**and**  $t \in \text{trm}$   $FvarsT\ t = \{\}$   
**and**  $\tau t$ : *prveqLPT*  $\tau$   $t$   
**shows** *prv* (*imp* (*instInpP*  $\varphi$  0)  $\tau$ ) (*instInpP*  $\varphi$   $t$ )  
*<proof>*

**lemma** *prveqLPT\_nprv\_instInpP\_instInp*:

**assumes** *[simp]*:  $\varphi \in \text{fm}la$  **and**  $f$ :  $Fvars \varphi \subseteq \{inp\}$  **and**  $\tau$ :  $\tau \in \text{ptrm}\ 0$   
**and** *[simp]*:  $t \in \text{trm}$   $FvarsT\ t = \{\}$   
**and**  $\tau t$ : *prveqLPT*  $\tau$   $t$   
**shows** *nprv* {*instInpP*  $\varphi$   $t$ } (*instInpP*  $\varphi$  0)  $\tau$ )  
*<proof>*

**lemma** *prveqLPT\_prv\_instInpP\_instInp*:

**assumes**  $\varphi \in \text{fm}la$  **and**  $f$ :  $Fvars \varphi \subseteq \{inp\}$  **and**  $\tau$ :  $\tau \in \text{ptrm}\ 0$   
**and**  $t \in \text{trm}$   $FvarsT\ t = \{\}$   
**and**  $\tau t$ : *prveqLPT*  $\tau$   $t$   
**shows** *prv* (*imp* (*instInpP*  $\varphi$   $t$ ) (*instInpP*  $\varphi$  0)  $\tau$ )  
*<proof>*

**lemma** *prveqLPT\_prv\_instInp\_eqv\_instInpP*:

**assumes**  $\varphi \in \text{fm}la$  **and**  $f$ :  $Fvars \varphi \subseteq \{inp\}$  **and**  $\tau$ :  $\tau \in \text{ptrm}\ 0$   
**and**  $t \in \text{trm}$   $FvarsT\ t = \{\}$   
**and**  $\tau t$ : *prveqLPT*  $\tau$   $t$

**shows**  $prv (eqv (instInpP \varphi 0 \tau) (instInp \varphi t))$   
 $\langle proof \rangle$

**end** — context *Deduct\_with\_False\_Disj\_Rename*

## Chapter 6

# Truth in a Standard Model

Abstract notion of standard model and truth.

First some minimal assumptions, involving implication, negation and (universal and existential) quantification:

```
locale Minimal_Truth =  
Syntax_with_Numerals_and_Connectives_False_Disj  
  var trm fmla Var FvarsT substT Fvars subst  
  eql cnj imp all exi  
  fls  
  dsj  
  num  
for  
var :: 'var set and trm :: 'trm set and fmla :: 'fmla set  
and Var FvarsT substT Fvars subst  
and eql cnj imp all exi  
and fls  
and dsj  
and num  
+  
— The notion of truth for sentences:  
fixes isTrue :: 'fmla  $\Rightarrow$  bool  
assumes  
not_isTrue_fls:  $\neg$  isTrue fls  
and  
isTrue_imp:  
 $\bigwedge \varphi \psi. \varphi \in \text{fmla} \Rightarrow \psi \in \text{fmla} \Rightarrow \text{Fvars } \varphi = \{\} \Rightarrow \text{Fvars } \psi = \{\} \Rightarrow$   
   $\text{isTrue } \varphi \Rightarrow \text{isTrue } (\text{imp } \varphi \psi) \Rightarrow \text{isTrue } \psi$   
and  
isTrue_all:  
 $\bigwedge x \varphi. x \in \text{var} \Rightarrow \varphi \in \text{fmla} \Rightarrow \text{Fvars } \varphi = \{x\} \Rightarrow$   
   $(\forall n \in \text{num}. \text{isTrue } (\text{subst } \varphi n x)) \Rightarrow \text{isTrue } (\text{all } x \varphi)$   
and  
isTrue_exi:  
 $\bigwedge x \varphi. x \in \text{var} \Rightarrow \varphi \in \text{fmla} \Rightarrow \text{Fvars } \varphi = \{x\} \Rightarrow$   
   $\text{isTrue } (\text{exi } x \varphi) \Rightarrow (\exists n \in \text{num}. \text{isTrue } (\text{subst } \varphi n x))$   
and  
isTrue_neg:  
 $\bigwedge \varphi. \varphi \in \text{fmla} \Rightarrow \text{Fvars } \varphi = \{\} \Rightarrow$   
   $\text{isTrue } \varphi \vee \text{isTrue } (\text{neg } \varphi)$   
begin  
  
lemma isTrue_neg_excl:
```

$\varphi \in \text{fmla} \implies \text{Fvars } \varphi = \{\} \implies$   
 $\text{isTrue } \varphi \implies \text{isTrue } (\text{neg } \varphi) \implies \text{False}$   
 ⟨proof⟩

**lemma** *isTrue\_neg\_neg*:  
**assumes**  $\varphi \in \text{fmla}$   $\text{Fvars } \varphi = \{\}$   
**and**  $\text{isTrue } (\text{neg } (\text{neg } \varphi))$   
**shows**  $\text{isTrue } \varphi$   
 ⟨proof⟩

**end** — context *Minimal\_Truth*

**locale** *Minimal\_Truth\_Soundness* =  
*Minimal\_Truth*  
 var *trm fmla Var FvarsT substT Fvars subst*  
 eql *cnj imp all exi*  
 fls  
 dsj  
 num  
 isTrue  
 +  
*Deduct\_with\_False\_Disj*  
 var *trm fmla Var FvarsT substT Fvars subst*  
 eql *cnj imp all exi*  
 fls  
 dsj  
 num  
 prv  
**for**  
 var :: 'var set **and** trm :: 'trm set **and** fmla :: 'fmla set  
**and** Var FvarsT substT Fvars subst  
**and** eql *cnj imp all exi*  
**and** fls  
**and** dsj  
**and** num  
**and** prv  
**and** isTrue  
 +  
**assumes**  
 — We assume soundness of the provability for sentences (w.r.t. truth):  
*sound\_isTrue*:  $\bigwedge \varphi. \varphi \in \text{fmla} \implies \text{Fvars } \varphi = \{\} \implies \text{prv } \varphi \implies \text{isTrue } \varphi$   
**begin**

For sound theories, consistency is a fact rather than a hypothesis:

**lemma** *consistent*: *consistent*  
 ⟨proof⟩

**lemma** *prv\_neg\_excl*:  
 $\varphi \in \text{fmla} \implies \text{Fvars } \varphi = \{\} \implies \text{prv } \varphi \implies \text{prv } (\text{neg } \varphi) \implies \text{False}$   
 ⟨proof⟩

**lemma** *prv\_imp\_implies\_isTrue*:  
**assumes** [*simp*]:  $\varphi \in \text{fmla}$   $\chi \in \text{fmla}$   $\text{Fvars } \varphi = \{\}$   $\text{Fvars } \chi = \{\}$   
**and**  $p$ :  $\text{prv } (\text{imp } \varphi \chi)$  **and**  $i$ :  $\text{isTrue } \varphi$   
**shows**  $\text{isTrue } \chi$   
 ⟨proof⟩

Sound theories are not only consistent, but also  $\omega$ -consistent (in the strong, intuitionistic sense):



**lemma**  $\omega$ consistent:  $\omega$ consistent  
*<proof>*

**lemma**  $\omega$ consistentStd1:  $\omega$ consistentStd1  
*<proof>*

**lemma**  $\omega$ consistentStd2:  $\omega$ consistentStd2  
*<proof>*

**end** — context *Minimal\_Truth\_Soundness*

## Chapter 7

# Arithmetic Constructs

Less generic syntax, more committed towards embedding arithmetics

(An embedding of) the syntax of arithmetic, obtained by adding plus and times

```
locale Syntax_Arith_aux =  
Syntax_with_Connectives_Rename  
  var trm fmla Var FvarsT substT Fvars subst  
  eql cnj imp all exi  
+  
Syntax_with_Numerals_and_Connectives_False_Disj  
  var trm fmla Var FvarsT substT Fvars subst  
  eql cnj imp all exi  
  fls  
  dsj  
  num  
for  
  var :: 'var set and trm :: 'trm set and fmla :: 'fmla set  
and Var FvarsT substT Fvars subst  
and eql cnj imp all exi  
and fls  
and dsj  
and num  
+  
fixes  
zer :: 'trm  
and  
suc :: 'trm  $\Rightarrow$  'trm  
and  
pls :: 'trm  $\Rightarrow$  'trm  $\Rightarrow$  'trm  
and  
tms :: 'trm  $\Rightarrow$  'trm  $\Rightarrow$  'trm  
assumes  
Fvars_zero[simp,intro!]: FvarsT zer = {}  
and  
substT_zer[simp]:  $\bigwedge t x. t \in trm \implies x \in var \implies$   
  substT zer t x = zer  
and  
suc[simp]:  $\bigwedge t. t \in trm \implies suc t \in trm$   
and  
FvarsT_suc[simp]:  $\bigwedge t. t \in trm \implies$   
  FvarsT (suc t) = FvarsT t  
and  
substT_suc[simp]:  $\bigwedge t1 t x. t1 \in trm \implies t \in trm \implies x \in var \implies$ 
```

```

  substT (suc t1) t x = suc (substT t1 t x)
and
pls[simp]:  $\bigwedge t1\ t2. t1 \in trm \implies t2 \in trm \implies pls\ t1\ t2 \in trm$ 
and
Fvars_pls[simp]:  $\bigwedge t1\ t2. t1 \in trm \implies t2 \in trm \implies$ 
  FvarsT (pls t1 t2) = FvarsT t1  $\cup$  FvarsT t2
and
substT_pls[simp]:  $\bigwedge t1\ t2\ t\ x. t1 \in trm \implies t2 \in trm \implies t \in trm \implies x \in var \implies$ 
  substT (pls t1 t2) t x = pls (substT t1 t x) (substT t2 t x)
and
tms[simp]:  $\bigwedge t1\ t2. t1 \in trm \implies t2 \in trm \implies tms\ t1\ t2 \in trm$ 
and
Fvars_tms[simp]:  $\bigwedge t1\ t2. t1 \in trm \implies t2 \in trm \implies$ 
  FvarsT (tms t1 t2) = FvarsT t1  $\cup$  FvarsT t2
and
substT_tms[simp]:  $\bigwedge t1\ t2\ t\ x. t1 \in trm \implies t2 \in trm \implies t \in trm \implies x \in var \implies$ 
  substT (tms t1 t2) t x = tms (substT t1 t x) (substT t2 t x)
begin

```

The embedding of numbers into our abstract notion of numerals (not required to be surjective)

```

fun Num :: nat  $\Rightarrow$  'trm where
  Num 0 = zer
|Num (Suc n) = suc (Num n)

end — context Syntax_Arith_aux

```

```

locale Syntax_Arith =
Syntax_Arith_aux
  var trm fmla Var FvarsT substT Fvars subst
  eql cnj imp all exi
  fls
  dsj
  num
  zer suc pls tms
for
var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
and Var FvarsT substT Fvars subst
and eql cnj imp all exi
and fls
and dsj
and num
zer suc pls tms
+
assumes
— We assume that numbers are the only numerals:
num_Num: num = range Num
begin

```

```

lemma Num[simp,intro!]: Num n  $\in$  num
  <proof>

```

```

lemma FvarsT_Num[simp]: FvarsT (Num n) = {}
  <proof>

```

```

lemma substT_Num[simp]:  $x \in var \implies t \in trm \implies substT (Num n) t x = Num n$ 
  <proof>

```

**lemma** *zer*[*simp,intro!*]:  $zer \in num$   
**and** *suc\_num*[*simp*]:  $\bigwedge n. n \in num \implies suc\ n \in num$   
*<proof>*

## 7.1 Arithmetic Terms

Arithmetic terms are inductively defined to contain the numerals and the variables and be closed under the arithmetic operators:

**inductive\_set** *atrm* :: 'trm set **where**  
*atrm\_num*[*simp*]:  $n \in num \implies n \in atrm$   
*atrm\_Var*[*simp,intro*]:  $x \in var \implies Var\ x \in atrm$   
*atrm\_suc*[*simp,intro*]:  $t \in atrm \implies suc\ t \in atrm$   
*atrm\_pls*[*simp,intro*]:  $t \in atrm \implies t' \in atrm \implies pls\ t\ t' \in atrm$   
*atrm\_tms*[*simp,intro*]:  $t \in atrm \implies t' \in atrm \implies tms\ t\ t' \in atrm$

**lemma** *atrm\_imp\_trm*[*simp*]: **assumes**  $t \in atrm$  **shows**  $t \in trm$   
*<proof>*

**lemma** *atrm\_trm*:  $atrm \subseteq trm$   
*<proof>*

**lemma** *zer\_atrm*[*simp*]:  $zer \in atrm$  *<proof>*

**lemma** *Num\_atrm*[*simp*]:  $Num\ n \in atrm$   
*<proof>*

**lemma** *substT\_atrm*[*simp*]:  
**assumes**  $r \in atrm$  **and**  $x \in var$  **and**  $t \in atrm$   
**shows**  $substT\ r\ t\ x \in atrm$   
*<proof>*

Whereas we did not assume the rich set of formula-substitution properties to hold for all terms, we can prove that these properties hold for arithmetic terms.

Properties for arithmetic terms corresponding to the axioms for formulas:

**lemma** *FvarsT\_substT*:  
**assumes**  $s \in atrm$   $t \in trm$   $x \in var$   
**shows**  $FvarsT\ (substT\ s\ t\ x) = (FvarsT\ s - \{x\}) \cup (if\ x \in FvarsT\ s\ then\ FvarsT\ t\ else\ \{\})$   
*<proof>*

**lemma** *substT\_compose\_eq\_or*:  
**assumes**  $s \in atrm$   $t1 \in trm$   $t2 \in trm$   $x1 \in var$   $x2 \in var$   
**and**  $x1 = x2 \vee x2 \notin FvarsT\ s$   
**shows**  $substT\ (substT\ s\ t1\ x1)\ t2\ x2 = substT\ s\ (substT\ t1\ t2\ x2)\ x1$   
*<proof>*

**lemma** *substT\_compose\_diff*:  
**assumes**  $s \in atrm$   $t1 \in trm$   $t2 \in trm$   $x1 \in var$   $x2 \in var$   
**and**  $x1 \neq x2$   $x1 \notin FvarsT\ t2$   
**shows**  $substT\ (substT\ s\ t1\ x1)\ t2\ x2 = substT\ (substT\ s\ t2\ x2)\ (substT\ t1\ t2\ x2)\ x1$   
*<proof>*

**lemma** *substT\_same\_Var*[*simp*]:  
**assumes**  $s \in atrm$   $x \in var$   
**shows**  $substT\ s\ (Var\ x)\ x = s$   
*<proof>*

... and corresponding to some corollaries we proved for formulas (with essentially the same proofs):

**lemma** *in\_FvarsT\_substTD*:

$y \in \text{FvarsT } (\text{substT } r \ t \ x) \implies r \in \text{atrm} \implies t \in \text{trm} \implies x \in \text{var}$   
 $\implies y \in (\text{FvarsT } r - \{x\}) \cup (\text{if } x \in \text{FvarsT } r \text{ then } \text{FvarsT } t \text{ else } \{\})$   
 ⟨proof⟩

**lemma** *substT\_compose\_same*:

$\bigwedge s \ t1 \ t2 \ x. s \in \text{atrm} \implies t1 \in \text{trm} \implies t2 \in \text{trm} \implies x \in \text{var} \implies$   
 $\text{substT } (\text{substT } s \ t1 \ x) \ t2 \ x = \text{substT } s \ (\text{substT } t1 \ t2 \ x) \ x$   
 ⟨proof⟩

**lemma** *substT\_substT[simp]*:

**assumes**  $s[\text{simp}]: s \in \text{atrm}$  **and**  $t[\text{simp}]: t \in \text{trm}$  **and**  $x[\text{simp}]: x \in \text{var}$  **and**  $y[\text{simp}]: y \in \text{var}$   
**assumes**  $yy: x \neq y \ y \notin \text{FvarsT } s$   
**shows**  $\text{substT } (\text{substT } s \ (\text{Var } y) \ x) \ t \ y = \text{substT } s \ t \ x$   
 ⟨proof⟩

**lemma** *substT\_comp*:

$\bigwedge x \ y \ s \ t. s \in \text{atrm} \implies t \in \text{trm} \implies x \in \text{var} \implies y \in \text{var} \implies$   
 $x \neq y \implies y \notin \text{FvarsT } t \implies$   
 $\text{substT } (\text{substT } s \ (\text{Var } x) \ y) \ t \ x = \text{substT } (\text{substT } s \ t \ x) \ t \ y$   
 ⟨proof⟩

Now the corresponding development of parallel substitution for arithmetic terms:

**lemma** *rawpsubstT\_atrm[simp,intro]*:

**assumes**  $r \in \text{atrm}$  **and**  $\text{snd } '(\text{set } \text{txs}) \subseteq \text{var}$  **and**  $\text{fst } '(\text{set } \text{txs}) \subseteq \text{atrm}$   
**shows**  $\text{rawpsubstT } r \ \text{txs} \in \text{atrm}$   
 ⟨proof⟩

**lemma** *psubstT\_atrm[simp,intro]*:

**assumes**  $r \in \text{atrm}$  **and**  $\text{snd } '(\text{set } \text{txs}) \subseteq \text{var}$  **and**  $\text{fst } '(\text{set } \text{txs}) \subseteq \text{atrm}$   
**shows**  $\text{psubstT } r \ \text{txs} \in \text{atrm}$   
 ⟨proof⟩

**lemma** *Fvars\_rawpsubst\_su*:

**assumes**  $r \in \text{atrm}$  **and**  $\text{snd } '(\text{set } \text{txs}) \subseteq \text{var}$  **and**  $\text{fst } '(\text{set } \text{txs}) \subseteq \text{atrm}$   
**shows**  $\text{FvarsT } (\text{rawpsubstT } r \ \text{txs}) \subseteq$   
 $(\text{FvarsT } r - \text{snd } '(\text{set } \text{txs})) \cup (\bigcup \{ \text{FvarsT } t \mid t \ x. (t,x) \in \text{set } \text{txs} \})$   
 ⟨proof⟩

**lemma** *in\_FvarsT\_rawpsubstT\_imp*:

**assumes**  $y \in \text{FvarsT } (\text{rawpsubstT } r \ \text{txs})$   
**and**  $r \in \text{atrm}$  **and**  $\text{snd } '(\text{set } \text{txs}) \subseteq \text{var}$  **and**  $\text{fst } '(\text{set } \text{txs}) \subseteq \text{atrm}$   
**shows**  $(y \in \text{FvarsT } r - \text{snd } '(\text{set } \text{txs})) \vee$   
 $(y \in \bigcup \{ \text{FvarsT } t \mid t \ x. (t,x) \in \text{set } \text{txs} \})$   
 ⟨proof⟩

**lemma** *FvarsT\_rawpsubstT*:

**assumes**  $r \in \text{atrm}$  **and**  $\text{snd } '(\text{set } \text{txs}) \subseteq \text{var}$  **and**  $\text{fst } '(\text{set } \text{txs}) \subseteq \text{atrm}$   
**and** *distinct*  $(\text{map } \text{snd } \ \text{txs})$  **and**  $\forall x \in \text{snd } '(\text{set } \text{txs}). \forall t \in \text{fst } '(\text{set } \text{txs}). x \notin \text{FvarsT } t$   
**shows**  $\text{FvarsT } (\text{rawpsubstT } r \ \text{txs}) =$   
 $(\text{FvarsT } r - \text{snd } '(\text{set } \text{txs})) \cup$   
 $(\bigcup \{ \text{if } x \in \text{FvarsT } r \text{ then } \text{FvarsT } t \text{ else } \{\} \mid t \ x. (t,x) \in \text{set } \text{txs} \})$   
 ⟨proof⟩

**lemma** *in\_FvarsT\_rawpsubstTD*:

**assumes**  $y \in \text{FvarsT } (\text{rawpsubstT } r \ \text{txs})$   
**and**  $r \in \text{atrm}$  **and**  $\text{snd } '(\text{set } \text{txs}) \subseteq \text{var}$  **and**  $\text{fst } '(\text{set } \text{txs}) \subseteq \text{atrm}$

**and** *distinct* (*map snd txs*) **and**  $\forall x \in \text{snd } '(\text{set txs}). \forall t \in \text{fst } '(\text{set txs}). x \notin \text{FvarsT } t$   
**shows**  $(y \in \text{FvarsT } r - \text{snd } '(\text{set txs})) \vee$   
 $(y \in \bigcup \{ \text{if } x \in \text{FvarsT } r \text{ then } \text{FvarsT } t \text{ else } \{ \} \mid t x . (t,x) \in \text{set txs} \})$   
*<proof>*

**lemma** *FvarsT\_psubstT*:  
**assumes**  $r \in \text{atrm}$  **and**  $\text{snd } '(\text{set txs}) \subseteq \text{var}$  **and**  $\text{fst } '(\text{set txs}) \subseteq \text{atrm}$   
**and** *distinct* (*map snd txs*)  
**shows**  $\text{FvarsT } (\text{psubstT } r \text{ txs}) =$   
 $(\text{FvarsT } r - \text{snd } '(\text{set txs})) \cup$   
 $(\bigcup \{ \text{if } x \in \text{FvarsT } r \text{ then } \text{FvarsT } t \text{ else } \{ \} \mid t x . (t,x) \in \text{set txs} \})$   
*<proof>*

**lemma** *in\_FvarsT\_psubstTD*:  
**assumes**  $y \in \text{FvarsT } (\text{psubstT } r \text{ txs})$   
**and**  $r \in \text{atrm}$  **and**  $\text{snd } '(\text{set txs}) \subseteq \text{var}$  **and**  $\text{fst } '(\text{set txs}) \subseteq \text{atrm}$   
**and** *distinct* (*map snd txs*)  
**shows**  $y \in (\text{FvarsT } r - \text{snd } '(\text{set txs})) \cup$   
 $(\bigcup \{ \text{if } x \in \text{FvarsT } r \text{ then } \text{FvarsT } t \text{ else } \{ \} \mid t x . (t,x) \in \text{set txs} \})$   
*<proof>*

**lemma** *substT2\_fresh\_switch*:  
**assumes**  $r \in \text{atrm}$   $t \in \text{trm}$   $s \in \text{trm}$   $x \in \text{var}$   $y \in \text{var}$   
**and**  $x \neq y$   $x \notin \text{FvarsT } s$   $y \notin \text{FvarsT } t$   
**shows**  $\text{substT } (\text{substT } r \text{ s } y) \text{ t } x = \text{substT } (\text{substT } r \text{ t } x) \text{ s } y$  (**is** ?L = ?R)  
*<proof>*

**lemma** *rawpsubst2\_fresh\_switch*:  
**assumes**  $r \in \text{atrm}$   $t \in \text{trm}$   $s \in \text{trm}$   $x \in \text{var}$   $y \in \text{var}$   
**and**  $x \neq y$   $x \notin \text{FvarsT } s$   $y \notin \text{FvarsT } t$   
**shows**  $\text{rawpsubstT } r ((s,y),(t,x)) = \text{rawpsubstT } r ((t,x),(s,y))$   
*<proof>*

**lemma** *rawpsubstT\_compose*:  
**assumes**  $t \in \text{trm}$  **and**  $\text{snd } '(\text{set txs1}) \subseteq \text{var}$  **and**  $\text{fst } '(\text{set txs1}) \subseteq \text{atrm}$   
**and**  $\text{snd } '(\text{set txs2}) \subseteq \text{var}$  **and**  $\text{fst } '(\text{set txs2}) \subseteq \text{atrm}$   
**shows**  $\text{rawpsubstT } (\text{rawpsubstT } t \text{ txs1}) \text{ txs2} = \text{rawpsubstT } t (\text{txs1} @ \text{txs2})$   
*<proof>*

**lemma** *rawpsubstT\_subst\_fresh\_switch*:  
**assumes**  $r \in \text{atrm}$   $\text{snd } '(\text{set txs}) \subseteq \text{var}$  **and**  $\text{fst } '(\text{set txs}) \subseteq \text{atrm}$   
**and**  $\forall x \in \text{snd } '(\text{set txs}). x \notin \text{FvarsT } s$   
**and**  $\forall t \in \text{fst } '(\text{set txs}). y \notin \text{FvarsT } t$   
**and** *distinct* (*map snd txs*)  
**and**  $s \in \text{atrm}$  **and**  $y \in \text{var}$   $y \notin \text{snd } '(\text{set txs})$   
**shows**  $\text{rawpsubstT } (\text{substT } r \text{ s } y) \text{ txs} = \text{rawpsubstT } r (\text{txs} @ [(s,y)])$   
*<proof>*

**lemma** *substT\_rawpsubstT\_fresh\_switch*:  
**assumes**  $r \in \text{atrm}$   $\text{snd } '(\text{set txs}) \subseteq \text{var}$  **and**  $\text{fst } '(\text{set txs}) \subseteq \text{atrm}$   
**and**  $\forall x \in \text{snd } '(\text{set txs}). x \notin \text{FvarsT } s$   
**and**  $\forall t \in \text{fst } '(\text{set txs}). y \notin \text{FvarsT } t$   
**and** *distinct* (*map snd txs*)  
**and**  $s \in \text{atrm}$  **and**  $y \in \text{var}$   $y \notin \text{snd } '(\text{set txs})$   
**shows**  $\text{substT } (\text{rawpsubstT } r \text{ txs}) \text{ s } y = \text{rawpsubstT } r ((s,y) \# \text{txs})$   
*<proof>*

**lemma** *rawpsubstT\_compose\_freshVar*:  
**assumes**  $r \in \text{atrm}$   $\text{snd} \text{ ' (set txs) } \subseteq \text{var}$  **and**  $\text{fst} \text{ ' (set txs) } \subseteq \text{atrm}$   
**and**  $\text{distinct (map snd txs)}$   
**and**  $\bigwedge i j. i < j \implies j < \text{length txs} \implies \text{snd (txs!j)} \notin \text{FvarsT (fst (txs!i))}$   
**and**  $\text{us\_facts: set us } \subseteq \text{var}$   
 $\text{set us } \cap \text{FvarsT } r = \{\}$   
 $\text{set us } \cap \bigcup (\text{FvarsT} \text{ ' (fst ' (set txs))}) = \{\}$   
 $\text{set us } \cap \text{snd} \text{ ' (set txs) } = \{\}$   
 $\text{length us} = \text{length txs}$   
 $\text{distinct us}$   
**shows**  $\text{rawpsubstT (rawpsubstT } r \text{ (zip (map Var us) (map snd txs))) (zip (map fst txs) us) = rawpsubstT } r \text{ txs}$   
 $\langle \text{proof} \rangle$

**lemma** *rawpsubstT\_compose\_freshVar2\_aux*:  
**assumes**  $r[\text{simp}]$ :  $r \in \text{atrm}$   
**and**  $\text{ts: set ts } \subseteq \text{atrm}$   
**and**  $\text{xs: set xs } \subseteq \text{var}$   $\text{distinct xs}$   
**and**  $\text{us\_facts: set us } \subseteq \text{var}$   $\text{distinct us}$   
 $\text{set us } \cap \text{FvarsT } r = \{\}$   
 $\text{set us } \cap \bigcup (\text{FvarsT} \text{ ' (set ts)}) = \{\}$   
 $\text{set us } \cap \text{set xs} = \{\}$   
**and**  $\text{vs\_facts: set vs } \subseteq \text{var}$   $\text{distinct vs}$   
 $\text{set vs } \cap \text{FvarsT } r = \{\}$   
 $\text{set vs } \cap \bigcup (\text{FvarsT} \text{ ' (set ts)}) = \{\}$   
 $\text{set vs } \cap \text{set xs} = \{\}$   
**and**  $l$ :  $\text{length us} = \text{length xs}$   $\text{length vs} = \text{length xs}$   $\text{length ts} = \text{length xs}$   
**and**  $d$ :  $\text{set us } \cap \text{set vs} = \{\}$   
**shows**  $\text{rawpsubstT (rawpsubstT } r \text{ (zip (map Var us) xs)) (zip ts us) = rawpsubstT (rawpsubstT } r \text{ (zip (map Var vs) xs)) (zip ts vs)}$   
 $\langle \text{proof} \rangle$

**lemma** *rawpsubstT\_compose\_freshVar2*:  
**assumes**  $r[\text{simp}]$ :  $r \in \text{atrm}$   
**and**  $\text{ts: set ts } \subseteq \text{atrm}$   
**and**  $\text{xs: set xs } \subseteq \text{var}$   $\text{distinct xs}$   
**and**  $\text{us\_facts: set us } \subseteq \text{var}$   $\text{distinct us}$   
 $\text{set us } \cap \text{FvarsT } r = \{\}$   
 $\text{set us } \cap \bigcup (\text{FvarsT} \text{ ' (set ts)}) = \{\}$   
 $\text{set us } \cap \text{set xs} = \{\}$   
**and**  $\text{vs\_facts: set vs } \subseteq \text{var}$   $\text{distinct vs}$   
 $\text{set vs } \cap \text{FvarsT } r = \{\}$   
 $\text{set vs } \cap \bigcup (\text{FvarsT} \text{ ' (set ts)}) = \{\}$   
 $\text{set vs } \cap \text{set xs} = \{\}$   
**and**  $l$ :  $\text{length us} = \text{length xs}$   $\text{length vs} = \text{length xs}$   $\text{length ts} = \text{length xs}$   
**shows**  $\text{rawpsubstT (rawpsubstT } r \text{ (zip (map Var us) xs)) (zip ts us) = rawpsubstT (rawpsubstT } r \text{ (zip (map Var vs) xs)) (zip ts vs) (is ?L = ?R)}$   
 $\langle \text{proof} \rangle$

**lemma** *in\_fst\_image*:  $a \in \text{fst} \text{ ' } AB \iff (\exists b. (a,b) \in AB)$   $\langle \text{proof} \rangle$

**lemma** *psubstT\_eq\_rawpsubstT*:  
**assumes**  $r \in \text{atrm}$   $\text{snd} \text{ ' (set txs) } \subseteq \text{var}$  **and**  $\text{fst} \text{ ' (set txs) } \subseteq \text{atrm}$   
**and**  $\text{distinct (map snd txs)}$

**and**  $\bigwedge i j. i < j \implies j < \text{length } \text{txs} \implies \text{snd } (\text{txs}!j) \notin \text{FvarsT } (\text{fst } (\text{txs}!i))$   
**shows**  $\text{psubstT } r \ \text{txs} = \text{rawpsubstT } r \ \text{txs}$   
*<proof>*

**lemma** *psubstT\_eq\_substT*:  
**assumes**  $r \in \text{atrm } x \in \text{var}$  **and**  $t \in \text{atrm}$   
**shows**  $\text{psubstT } r \ [(t,x)] = \text{substT } r \ t \ x$   
*<proof>*

**lemma** *psubstT\_eq\_rawpsubst2*:  
**assumes**  $r \in \text{atrm } x1 \in \text{var } x2 \in \text{var } t1 \in \text{atrm } t2 \in \text{atrm}$   
**and**  $x1 \neq x2 \ x2 \notin \text{FvarsT } t1$   
**shows**  $\text{psubstT } r \ [(t1,x1),(t2,x2)] = \text{rawpsubstT } r \ [(t1,x1),(t2,x2)]$   
*<proof>*

**lemma** *psubstT\_eq\_rawpsubst3*:  
**assumes**  $r \in \text{atrm } x1 \in \text{var } x2 \in \text{var } x3 \in \text{var } t1 \in \text{atrm } t2 \in \text{atrm } t3 \in \text{atrm}$   
**and**  $x1 \neq x2 \ x1 \neq x3 \ x2 \neq x3$   
 $x2 \notin \text{FvarsT } t1 \ x3 \notin \text{FvarsT } t1 \ x3 \notin \text{FvarsT } t2$   
**shows**  $\text{psubstT } r \ [(t1,x1),(t2,x2),(t3,x3)] = \text{rawpsubstT } r \ [(t1,x1),(t2,x2),(t3,x3)]$   
*<proof>*

**lemma** *psubstT\_eq\_rawpsubst4*:  
**assumes**  $r \in \text{atrm } x1 \in \text{var } x2 \in \text{var } x3 \in \text{var } x4 \in \text{var}$   
 $t1 \in \text{atrm } t2 \in \text{atrm } t3 \in \text{atrm } t4 \in \text{atrm}$   
**and**  $x1 \neq x2 \ x1 \neq x3 \ x2 \neq x3 \ x1 \neq x4 \ x2 \neq x4 \ x3 \neq x4$   
 $x2 \notin \text{FvarsT } t1 \ x3 \notin \text{FvarsT } t1 \ x3 \notin \text{FvarsT } t2 \ x4 \notin \text{FvarsT } t1 \ x4 \notin \text{FvarsT } t2 \ x4 \notin \text{FvarsT } t3$   
**shows**  $\text{psubstT } r \ [(t1,x1),(t2,x2),(t3,x3),(t4,x4)] = \text{rawpsubstT } r \ [(t1,x1),(t2,x2),(t3,x3),(t4,x4)]$   
*<proof>*

**lemma** *rawpsubstT\_same\_Var[simp]*:  
**assumes**  $r \in \text{atrm } \text{set } \text{xs} \subseteq \text{var}$   
**shows**  $\text{rawpsubstT } r \ (\text{map } (\lambda x. (\text{Var } x,x)) \ \text{xs}) = r$   
*<proof>*

**lemma** *psubstT\_same\_Var[simp]*:  
**assumes**  $r \in \text{atrm } \text{set } \text{xs} \subseteq \text{var}$  **and** *distinct xs*  
**shows**  $\text{psubstT } r \ (\text{map } (\lambda x. (\text{Var } x,x)) \ \text{xs}) = r$   
*<proof>*

**thm** *psubstT\_notIn*

**lemma** *rawpsubst\_eq1*:  
**assumes**  $t1 \in \text{trm } t2 \in \text{trm}$   
**and**  $\text{snd } ' (\text{set } \text{txs}) \subseteq \text{var } \text{fst } ' (\text{set } \text{txs}) \subseteq \text{trm}$   
**shows**  $\text{rawpsubst } (\text{eq1 } t1 \ t2) \ \text{txs} = \text{eq1 } (\text{rawpsubstT } t1 \ \text{txs}) \ (\text{rawpsubstT } t2 \ \text{txs})$   
*<proof>*

**lemma** *psubst\_eq1[simp]*:  
**assumes**  $t1 \in \text{atrm } t2 \in \text{atrm}$   
**and**  $\text{snd } ' (\text{set } \text{txs}) \subseteq \text{var } \text{fst } ' (\text{set } \text{txs}) \subseteq \text{atrm}$   
**and** *distinct (map snd txs)*



**shows**  $psubst (eql\ t1\ t2)\ txs = eql (psubstT\ t1\ txs) (psubstT\ t2\ txs)$   
 ⟨proof⟩

**lemma**  $psubst\_exu[simp]$ :  
**assumes**  $\varphi \in fmla\ x \in var\ snd \ ' (set\ txs \subseteq var\ fst \ ' (set\ txs \subseteq atrm)$   
 $x \notin snd \ ' (set\ txs\ x \notin (\bigcup t \in fst \ ' (set\ txs.\ FvarsT\ t)\ distinct\ (map\ snd\ txs)$   
**shows**  $psubst (exu\ x\ \varphi)\ txs = exu\ x\ (psubst\ \varphi\ txs)$   
 ⟨proof⟩

**thm**  $psubstT\_Var\_not[no\_vars]$

**lemma**  $rawpsubstT\_Var\_in$ :  
**assumes**  $snd \ ' (set\ txs) \subseteq var\ fst \ ' (set\ txs) \subseteq trm$   
**and**  $distinct\ (map\ snd\ txs)$  **and**  $(s,y) \in set\ txs$   
**and**  $\bigwedge i\ j.\ i < j \implies j < length\ txs \implies snd\ (txs!j) \notin FvarsT\ (fst\ (txs!i))$   
**shows**  $rawpsubstT\ (Var\ y)\ txs = s$   
 ⟨proof⟩

**lemma**  $psubstT\_Var\_in$ :  
**assumes**  $y \in var\ snd \ ' (set\ txs) \subseteq var\ fst \ ' (set\ txs) \subseteq trm$   
**and**  $distinct\ (map\ snd\ txs)$  **and**  $(s,y) \in set\ txs$   
**shows**  $psubstT\ (Var\ y)\ txs = s$   
 ⟨proof⟩

**lemma**  $psubstT\_Var\_Cons\_aux$ :  
**assumes**  $y \in var\ x \in var\ t \in atrm$   
 $snd \ ' (set\ txs) \subseteq var\ fst \ ' (set\ txs) \subseteq atrm\ x \notin snd \ ' (set\ txs)$   
 $distinct\ (map\ snd\ txs)\ y \neq x$   
**shows**  $psubstT\ (Var\ y)\ ((t,x) \# txs) = psubstT\ (Var\ y)\ txs$   
 ⟨proof⟩

Simplification rules for parallel substitution:

**lemma**  $psubstT\_Var\_Cons[simp]$ :  
 $y \in var \implies x \in var \implies t \in atrm \implies$   
 $snd \ ' (set\ txs) \subseteq var \implies fst \ ' (set\ txs) \subseteq atrm \implies distinct\ (map\ snd\ txs) \implies x \notin snd \ ' (set\ txs) \implies$   
 $psubstT\ (Var\ y)\ ((t,x) \# txs) = (if\ y = x\ then\ t\ else\ psubstT\ (Var\ y)\ txs)$   
 ⟨proof⟩

**lemma**  $psubstT\_zer[simp]$ :  
**assumes**  $snd \ ' (set\ txs) \subseteq var$  **and**  $fst \ ' (set\ txs) \subseteq trm$   
**shows**  $psubstT\ zer\ txs = zer$   
 ⟨proof⟩

**lemma**  $rawpsubstT\_suc$ :  
**assumes**  $r \in trm$  **and**  $snd \ ' (set\ txs) \subseteq var$  **and**  $fst \ ' (set\ txs) \subseteq trm$   
**shows**  $rawpsubstT\ (suc\ r)\ txs = suc\ (rawpsubstT\ r\ txs)$   
 ⟨proof⟩

**lemma**  $psubstT\_suc[simp]$ :  
**assumes**  $r \in atrm$  **and**  $snd \ ' (set\ txs) \subseteq var$  **and**  $fst \ ' (set\ txs) \subseteq atrm$   
**and**  $distinct\ (map\ snd\ txs)$   
**shows**  $psubstT\ (suc\ r)\ txs = suc\ (psubstT\ r\ txs)$   
 ⟨proof⟩

**lemma** *rawpsubstT\_pls*:  
**assumes**  $r1 \in trm$   $r2 \in trm$  **and**  $snd \text{ ` (set txs) } \subseteq var$  **and**  $fst \text{ ` (set txs) } \subseteq trm$   
**shows**  $rawpsubstT (pls\ r1\ r2)\ txs = pls (rawpsubstT\ r1\ txs) (rawpsubstT\ r2\ txs)$   
 $\langle proof \rangle$

**lemma** *psubstT\_pls[simp]*:  
**assumes**  $r1 \in atrm$   $r2 \in atrm$  **and**  $snd \text{ ` (set txs) } \subseteq var$  **and**  $fst \text{ ` (set txs) } \subseteq atrm$   
**and** *distinct* ( $map\ snd\ txs$ )  
**shows**  $psubstT (pls\ r1\ r2)\ txs = pls (psubstT\ r1\ txs) (psubstT\ r2\ txs)$   
 $\langle proof \rangle$

**lemma** *rawpsubstT\_tms*:  
**assumes**  $r1 \in trm$   $r2 \in trm$  **and**  $snd \text{ ` (set txs) } \subseteq var$  **and**  $fst \text{ ` (set txs) } \subseteq trm$   
**shows**  $rawpsubstT (tms\ r1\ r2)\ txs = tms (rawpsubstT\ r1\ txs) (rawpsubstT\ r2\ txs)$   
 $\langle proof \rangle$

**lemma** *psubstT\_tms[simp]*:  
**assumes**  $r1 \in atrm$   $r2 \in atrm$  **and**  $snd \text{ ` (set txs) } \subseteq var$  **and**  $fst \text{ ` (set txs) } \subseteq atrm$   
**and** *distinct* ( $map\ snd\ txs$ )  
**shows**  $psubstT (tms\ r1\ r2)\ txs = tms (psubstT\ r1\ txs) (psubstT\ r2\ txs)$   
 $\langle proof \rangle$

## 7.2 The (Nonstrict and Strict) Order Relations

$Lq$  (less than or equal to) is a formula with free vars  $xx$  and  $yy$ . NB: Out of the two possible ways, adding  $zz$  to the left or to the right, we choose the former, since this seems to enable Q (Robinson arithmetic) to prove as many useful properties as possible.

**definition**  $Lq :: \text{ `fmla where}$   
 $Lq \equiv exi\ zz\ (eql\ (Var\ yy)\ (pls\ (Var\ zz)\ (Var\ xx)))$

Alternative, more flexible definition , for any non-capturing bound variable:

**lemma**  $Lq\_def2: z \in var \implies z \neq yy \implies z \neq xx \implies Lq = exi\ z\ (eql\ (Var\ yy)\ (pls\ (Var\ z)\ (Var\ xx)))$   
 $\langle proof \rangle$

**lemma**  $Lq[simp,intro!]: Lq \in fmla$   
 $\langle proof \rangle$

**lemma**  $Fvars\_Lq[simp]: Fvars\ Lq = \{xx,yy\}$   
 $\langle proof \rangle$

As usual, we also define a predicate version:

**definition**  $LLq \text{ where } LLq \equiv \lambda\ t1\ t2.\ psubst\ Lq\ [(t1,xx), (t2,yy)]$

**lemma**  $LLq[simp,intro!]:$   
**assumes**  $t1 \in trm$   $t2 \in trm$   
**shows**  $LLq\ t1\ t2 \in fmla$   
 $\langle proof \rangle$

**lemma**  $LLq2[simp,intro!]:$   
 $n \in num \implies LLq\ n\ (Var\ yy') \in fmla$   
 $\langle proof \rangle$

**lemma**  $Fvars\_LLq[simp]: t1 \in trm \implies t2 \in trm \implies$   
 $Fvars\ (LLq\ t1\ t2) = FvarsT\ t1 \cup FvarsT\ t2$   
 $\langle proof \rangle$

This lemma will be the working definition of LLq:

**lemma** *LLq\_pls*:

**assumes** [*simp*]:  $t1 \in atrm \ t2 \in atrm \ z \in var \ z \notin FvarsT \ t1 \ z \notin FvarsT \ t2$

**shows**  $LLq \ t1 \ t2 = exi \ z \ (eq \ t2 \ (pls \ (Var \ z) \ t1))$

*<proof>*

**lemma** *LLq\_pls\_zz*:

**assumes**  $t1 \in atrm \ t2 \in atrm \ zz \notin FvarsT \ t1 \ zz \notin FvarsT \ t2$

**shows**  $LLq \ t1 \ t2 = exi \ zz \ (eq \ t2 \ (pls \ (Var \ zz) \ t1))$

*<proof>*

If we restrict attention to arithmetic terms, we can prove a uniform substitution property for LLq:

**lemma** *subst\_LLq[simp]*:

**assumes** [*simp*]:  $t1 \in atrm \ t2 \in atrm \ s \in atrm \ x \in var$

**shows**  $subst \ (LLq \ t1 \ t2) \ s \ x = LLq \ (substT \ t1 \ s \ x) \ (substT \ t2 \ s \ x)$

*<proof>*

**lemma** *psubst\_LLq[simp]*:

**assumes** 1:  $t1 \in atrm \ t2 \in atrm \ fst \ ' \ set \ txs \subseteq atrm$

**and** 2:  $snd \ ' \ set \ txs \subseteq var$

**and** 3: *distinct* (*map snd txs*)

**shows**  $psubst \ (LLq \ t1 \ t2) \ txs = LLq \ (psubstT \ t1 \ txs) \ (psubstT \ t2 \ txs)$

*<proof>*

Lq less than) is the strict version of the order relation. We prove similar facts as for Lq

**definition** *Ls* :: 'fmla **where**

$Ls \equiv cnj \ Lq \ (neg \ (eq \ (Var \ xx) \ (Var \ yy)))$

**lemma** *Ls[simp,intro!]*:  $Ls \in fmla$

*<proof>*

**lemma** *Fvars\_Ls[simp]*:  $Fvars \ Ls = \{xx,yy\}$

*<proof>*

**definition** *LLs* **where**  $LLs \equiv \lambda \ t1 \ t2. \ psubst \ Ls \ [(t1,xx), (t2,yy)]$

**lemma** *LLs[simp,intro]*:

**assumes**  $t1 \in trm \ t2 \in trm$

**shows**  $LLs \ t1 \ t2 \in fmla$

*<proof>*

**lemma** *LLs2[simp,intro!]*:

$n \in num \implies LLs \ n \ (Var \ yy') \in fmla$

*<proof>*

**lemma** *Fvars\_LLs[simp]*:  $t1 \in trm \implies t2 \in trm \implies$

$Fvars \ (LLs \ t1 \ t2) = FvarsT \ t1 \cup FvarsT \ t2$

*<proof>*

The working definition of LLs:

**lemma** *LLs\_LLq*:

$t1 \in atrm \implies t2 \in atrm \implies$

$LLs \ t1 \ t2 = cnj \ (LLq \ t1 \ t2) \ (neg \ (eq \ t1 \ t2))$

*<proof>*

**lemma** *subst\_LLs[simp]*:

**assumes** [*simp*]:  $t1 \in atrm \ t2 \in atrm \ s \in atrm \ x \in var$

**shows**  $\text{subst } (LLs \ t1 \ t2) \ s \ x = LLs \ (\text{substT } t1 \ s \ x) \ (\text{substT } t2 \ s \ x)$   
 ⟨proof⟩

**lemma**  $\text{psubst\_LLs}[simp]$ :  
**assumes**  $1: t1 \in \text{atrm} \ t2 \in \text{atrm} \ \text{fst } ' \ \text{set } \text{txs} \subseteq \text{atrm}$   
**and**  $2: \text{snd } ' \ \text{set } \text{txs} \subseteq \text{var}$   
**and**  $3: \text{distinct } (\text{map } \text{snd } \text{txs})$   
**shows**  $\text{psubst } (LLs \ t1 \ t2) \ \text{txs} = LLs \ (\text{psubstT } t1 \ \text{txs}) \ (\text{psubstT } t2 \ \text{txs})$   
 ⟨proof⟩

## 7.3 Bounded Quantification

Bounded forall

**definition**  $\text{ball} :: 'var \Rightarrow 'trm \Rightarrow 'fmla \Rightarrow 'fmla \ \text{where}$   
 $\text{ball } x \ t \ \varphi \equiv \text{all } x \ (\text{imp } (LLq \ (\text{Var } x) \ t) \ \varphi)$

**lemma**  $\text{ball}[simp, \ \text{intro}]$ :  $x \in \text{var} \Longrightarrow t \in \text{trm} \Longrightarrow \varphi \in \text{fmla} \Longrightarrow \text{ball } x \ t \ \varphi \in \text{fmla}$   
 ⟨proof⟩

**lemma**  $\text{Fvars\_ball}[simp]$ :  
 $x \in \text{var} \Longrightarrow \varphi \in \text{fmla} \Longrightarrow t \in \text{trm} \Longrightarrow \text{Fvars } (\text{ball } x \ t \ \varphi) = (\text{Fvars } \varphi \cup \text{FvarsT } t) - \{x\}$   
 ⟨proof⟩

**lemma**  $\text{subst\_ball}$ :  
 $\varphi \in \text{fmla} \Longrightarrow t \in \text{atrm} \Longrightarrow t1 \in \text{atrm} \Longrightarrow x \in \text{var} \Longrightarrow y \in \text{var} \Longrightarrow x \neq y \Longrightarrow x \notin \text{FvarsT } t1 \Longrightarrow$   
 $\text{subst } (\text{ball } x \ t \ \varphi) \ t1 \ y = \text{ball } x \ (\text{substT } t \ t1 \ y) \ (\text{subst } \varphi \ t1 \ y)$   
 ⟨proof⟩

**lemma**  $\text{psubst\_ball}$ :  
 $\varphi \in \text{fmla} \Longrightarrow y \in \text{var} \Longrightarrow \text{snd } ' \ \text{set } \text{txs} \subseteq \text{var} \Longrightarrow t \in \text{atrm} \Longrightarrow$   
 $\text{fst } ' \ \text{set } \text{txs} \subseteq \text{trm} \Longrightarrow \text{fst } ' \ \text{set } \text{txs} \subseteq \text{atrm} \Longrightarrow y \notin \text{snd } ' \ \text{set } \text{txs} \Longrightarrow y \notin (\bigcup t \in \text{fst } ' \ \text{set } \text{txs}. \text{FvarsT } t)$   
 $\Longrightarrow$   
 $\text{distinct } (\text{map } \text{snd } \text{txs}) \Longrightarrow$   
 $\text{psubst } (\text{ball } y \ t \ \varphi) \ \text{txs} = \text{ball } y \ (\text{psubstT } t \ \text{txs}) \ (\text{psubst } \varphi \ \text{txs})$   
 ⟨proof⟩

Bounded exists

**definition**  $\text{bexi} :: 'var \Rightarrow 'trm \Rightarrow 'fmla \Rightarrow 'fmla \ \text{where}$   
 $\text{bexi } x \ t \ \varphi \equiv \text{exi } x \ (\text{cnj } (LLq \ (\text{Var } x) \ t) \ \varphi)$

**lemma**  $\text{bexi}[simp, \ \text{intro}]$ :  $x \in \text{var} \Longrightarrow t \in \text{trm} \Longrightarrow \varphi \in \text{fmla} \Longrightarrow \text{bexi } x \ t \ \varphi \in \text{fmla}$   
 ⟨proof⟩

**lemma**  $\text{Fvars\_bexi}[simp]$ :  
 $x \in \text{var} \Longrightarrow \varphi \in \text{fmla} \Longrightarrow t \in \text{trm} \Longrightarrow \text{Fvars } (\text{bexi } x \ t \ \varphi) = (\text{Fvars } \varphi \cup \text{FvarsT } t) - \{x\}$   
 ⟨proof⟩

**lemma**  $\text{subst\_bexi}$ :  
 $\varphi \in \text{fmla} \Longrightarrow t \in \text{atrm} \Longrightarrow t1 \in \text{atrm} \Longrightarrow x \in \text{var} \Longrightarrow y \in \text{var} \Longrightarrow x \neq y \Longrightarrow x \notin \text{FvarsT } t1 \Longrightarrow$   
 $\text{subst } (\text{bexi } x \ t \ \varphi) \ t1 \ y = \text{bexi } x \ (\text{substT } t \ t1 \ y) \ (\text{subst } \varphi \ t1 \ y)$   
 ⟨proof⟩

**lemma**  $\text{psubst\_bexi}$ :  
 $\varphi \in \text{fmla} \Longrightarrow y \in \text{var} \Longrightarrow \text{snd } ' \ \text{set } \text{txs} \subseteq \text{var} \Longrightarrow t \in \text{atrm} \Longrightarrow$   
 $\text{fst } ' \ \text{set } \text{txs} \subseteq \text{trm} \Longrightarrow \text{fst } ' \ \text{set } \text{txs} \subseteq \text{atrm} \Longrightarrow y \notin \text{snd } ' \ \text{set } \text{txs} \Longrightarrow y \notin (\bigcup t \in \text{fst } ' \ \text{set } \text{txs}. \text{FvarsT } t)$   
 $\Longrightarrow$   
 $\text{distinct } (\text{map } \text{snd } \text{txs}) \Longrightarrow$

$psubst (bexi\ y\ t\ \varphi)\ txs = bexi\ y\ (psubstT\ t\ txs)\ (psubst\ \varphi\ txs)$   
(*proof*)

**end** — context *Syntax\_Arith*

## Chapter 8

# Deduction in a System Embedding the Intuitionistic Robinson Arithmetic

NB: Robinson arithmetic, also known as system Q, is Peano arithmetic without the induction axiom schema.

### 8.1 Natural Deduction with the Bounded Quantifiers

We start by simply putting together deduction with the arithmetic syntax, which allows us to reason about bounded quantifiers:

```
locale Deduct_with_False_Disj_Arith =
  Syntax_Arith
  var trm fmla Var FvarsT substT Fvars subst
  eql cnj imp all exi
  fls
  dsj
  num
  zer suc pls tms
+
  Deduct_with_False_Disj
  var trm fmla Var FvarsT substT Fvars subst
  eql cnj imp all exi
  fls
  dsj
  num
  prv
for
var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
and Var FvarsT substT Fvars subst
and eql cnj imp all exi
and fls
and dsj
and num
and zer suc pls tms
and prv
begin

lemma nprv_ballI:
```

$nprv (insert (LLq (Var x) t) F) \varphi \implies$   
 $F \subseteq fmla \implies finite F \implies \varphi \in fmla \implies t \in trm \implies x \in var \implies$   
 $x \notin (\bigcup \varphi \in F. Fvars \varphi) \implies x \notin FvarsT t \implies$   
 $nprv F (ball x t \varphi)$   
 <proof>

**lemma** *nprv\_ballE\_aux*:

$nprv F (ball x t \varphi) \implies nprv F (LLq t1 t) \implies$   
 $F \subseteq fmla \implies finite F \implies \varphi \in fmla \implies t \in atrm \implies t1 \in atrm \implies x \in var \implies x \notin FvarsT t \implies$   
 $nprv F (subst \varphi t1 x)$   
 <proof>

**lemma** *nprv\_ballE*:

$nprv F (ball x t \varphi) \implies nprv F (LLq t1 t) \implies nprv (insert (subst \varphi t1 x) F) \psi \implies$   
 $F \subseteq fmla \implies finite F \implies \varphi \in fmla \implies t \in atrm \implies t1 \in atrm \implies x \in var \implies \psi \in fmla \implies$   
 $x \notin FvarsT t \implies$   
 $nprv F \psi$   
 <proof>

**lemmas** *nprv\_ballE0* = *nprv\_ballE*[*OF nprv\_hyp \_ \_*, *simped*]

**lemmas** *nprv\_ballE1* = *nprv\_ballE*[*OF \_ nprv\_hyp \_*, *simped*]

**lemmas** *nprv\_ballE2* = *nprv\_ballE*[*OF \_ \_ nprv\_hyp*, *simped*]

**lemmas** *nprv\_ballE01* = *nprv\_ballE*[*OF nprv\_hyp nprv\_hyp \_*, *simped*]

**lemmas** *nprv\_ballE02* = *nprv\_ballE*[*OF nprv\_hyp \_ nprv\_hyp*, *simped*]

**lemmas** *nprv\_ballE12* = *nprv\_ballE*[*OF \_ nprv\_hyp nprv\_hyp*, *simped*]

**lemmas** *nprv\_ballE012* = *nprv\_ballE*[*OF nprv\_hyp nprv\_hyp nprv\_hyp*, *simped*]

**lemma** *nprv\_bexiI*:

$nprv F (subst \varphi t1 x) \implies nprv F (LLq t1 t) \implies$   
 $F \subseteq fmla \implies finite F \implies \varphi \in fmla \implies t \in atrm \implies t1 \in atrm \implies x \in var \implies$   
 $x \notin FvarsT t \implies$   
 $nprv F (bexi x t \varphi)$   
 <proof>

**lemma** *nprv\_bexiE*:

$nprv F (bexi x t \varphi) \implies nprv (insert (LLq (Var x) t) (insert \varphi F)) \psi \implies$   
 $F \subseteq fmla \implies finite F \implies \varphi \in fmla \implies x \in var \implies \psi \in fmla \implies t \in atrm \implies$   
 $x \notin (\bigcup \varphi \in F. Fvars \varphi) \implies x \notin Fvars \psi \implies x \notin FvarsT t \implies$   
 $nprv F \psi$   
 <proof>

**lemmas** *nprv\_bexiE0* = *nprv\_bexiE*[*OF nprv\_hyp \_*, *simped*]

**lemmas** *nprv\_bexiE1* = *nprv\_bexiE*[*OF \_ nprv\_hyp*, *simped*]

**lemmas** *nprv\_bexiE01* = *nprv\_bexiE*[*OF nprv\_hyp nprv\_hyp*, *simped*]

**end** — context *Deduct\_with\_False\_Disj*

## 8.2 Deduction with the Robinson Arithmetic Axioms

**locale** *Deduct\_Q* =

*Deduct\_with\_False\_Disj\_Arith*

*var trm fmla*

*Var FvarsT substT Fvars subst*

*eql cnj imp all exi*

*fls*

*dsj*

*num*

*zer suc pls tms*

```

prv
for
var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
and Var FvarsT substT Fvars subst
and eql cnj imp all exi
and fls
and dsj
and num
and zer suc pls tms
and prv
+

```

**assumes**

— The Q axioms are stated for some fixed variables; we will prove more useful versions, for arbitrary terms substituting the variables.

*prv\_neg\_zer\_suc\_var:*

*prv (neg (eql zer (suc (Var xx))))*

**and**

*prv\_inj\_suc\_var:*

*prv (imp (eql (suc (Var xx)) (suc (Var yy)))  
(eql (Var xx) (Var yy)))*

**and**

*prv\_zer\_dsj\_suc\_var:*

*prv (dsj (eql (Var yy) zer)  
(exi xx (eql (Var yy) (suc (Var xx))))*

**and**

*prv\_pls\_zer\_var:*

*prv (eql (pls (Var xx) zer) (Var xx))*

**and**

*prv\_pls\_suc\_var:*

*prv (eql (pls (Var xx) (suc (Var yy)))  
(suc (pls (Var xx) (Var yy))))*

**and**

*prv\_tms\_zer\_var:*

*prv (eql (tms (Var xx) zer) zer)*

**and**

*prv\_tms\_suc\_var:*

*prv (eql (tms (Var xx) (suc (Var yy)))  
(pls (tms (Var xx) (Var yy)) (Var xx)))*

**begin**

Rules for quantifiers that allow changing, on the fly, the bound variable with one that is fresh for the proof context:

**lemma** *nprv\_allI\_var:*

**assumes** *nI[simp]: nprv F (subst  $\varphi$  (Var y) x)*

**and** *i[simp]: F  $\subseteq$  fmla finite F  $\varphi \in$  fmla  $x \in$  var  $y \in$  var*

**and** *u: y  $\notin$  ( $\bigcup \varphi \in F$ . Fvars  $\varphi$ ) **and**  $yx[simp]: y = x \vee y \notin$  Fvars  $\varphi$*

**shows** *nprv F (all x  $\varphi$ )*

*<proof>*

**lemma** *nprv\_exiE\_var:*

**assumes** *n: nprv F (exi x  $\varphi$ )*

**and** *nn: nprv (insert (subst  $\varphi$  (Var y) x) F)  $\psi$*

**and** *0: F  $\subseteq$  fmla finite F  $\varphi \in$  fmla  $x \in$  var  $y \in$  var  $\psi \in$  fmla*

**and** *yx: y = x  $\vee$  y  $\notin$  Fvars  $\varphi$  y  $\notin$  ( $\bigcup$  (Fvars ' F) y  $\notin$  Fvars  $\psi$*

**shows** *nprv F  $\psi$*

*<proof>*



**lemma** *prv\_neg\_zer\_suc*:  
**assumes** [*simp*]:  $t \in \text{atrm}$  **shows**  $\text{prv} (\text{neg} (\text{eql zer} (\text{suc } t)))$   
 ⟨*proof*⟩

**lemma** *prv\_neg\_suc\_zer*:  
**assumes**  $t \in \text{atrm}$  **shows**  $\text{prv} (\text{neg} (\text{eql} (\text{suc } t) \text{zer}))$   
 ⟨*proof*⟩

**lemmas** *nprv\_zer\_suc\_contrE* =  
 $\text{nprv\_flsE}[OF \text{nprv\_addImpLemmaE}[OF \text{prv\_neg\_zer\_suc}[\text{unfolded neg\_def}], OF \_ \_ \text{nprv\_hyp}, \text{simped}, \text{rotated}]$

**lemmas** *nprv\_zer\_suc\_contrE0* =  $\text{nprv\_zer\_suc\_contrE}[OF \text{nprv\_hyp}, \text{simped}]$

**lemma** *nprv\_zer\_suc\_2contrE*:  
 $\text{nprv } F (\text{eql } t \text{ zer}) \implies \text{nprv } F (\text{eql } t (\text{suc } t1)) \implies$   
 $\text{finite } F \implies F \subseteq \text{fmla} \implies t \in \text{atrm} \implies t1 \in \text{atrm} \implies \varphi \in \text{fmla} \implies$   
 $\text{nprv } F \varphi$   
 ⟨*proof*⟩

**lemmas** *nprv\_zer\_suc\_2contrE0* =  $\text{nprv\_zer\_suc\_2contrE}[OF \text{nprv\_hyp} \_ , \text{simped}]$

**lemmas** *nprv\_zer\_suc\_2contrE1* =  $\text{nprv\_zer\_suc\_2contrE}[OF \_ \_ \text{nprv\_hyp}, \text{simped}]$

**lemmas** *nprv\_zer\_suc\_2contrE01* =  $\text{nprv\_zer\_suc\_2contrE}[OF \text{nprv\_hyp} \text{nprv\_hyp}, \text{simped}]$

**lemma** *prv\_inj\_suc*:  
 $t \in \text{atrm} \implies t' \in \text{atrm} \implies$   
 $\text{prv} (\text{imp} (\text{eql} (\text{suc } t) (\text{suc } t'))$   
 $(\text{eql } t \text{ } t'))$   
 ⟨*proof*⟩

**lemmas** *nprv\_eql\_sucI* =  $\text{nprv\_addImpLemmaI}[OF \text{prv\_inj\_suc}, \text{simped}, \text{rotated } 4]$

**lemmas** *nprv\_eql\_sucE* =  $\text{nprv\_addImpLemmaE}[OF \text{prv\_inj\_suc}, \text{simped}, \text{rotated } 2]$

**lemmas** *nprv\_eql\_sucE0* =  $\text{nprv\_eql\_sucE}[OF \text{nprv\_hyp} \_ , \text{simped}]$

**lemmas** *nprv\_eql\_sucE1* =  $\text{nprv\_eql\_sucE}[OF \_ \_ \text{nprv\_hyp}, \text{simped}]$

**lemmas** *nprv\_eql\_sucE01* =  $\text{nprv\_eql\_sucE}[OF \text{nprv\_hyp} \text{nprv\_hyp}, \text{simped}]$

**lemma** *prv\_zer\_dsj\_suc*:  
**assumes**  $t[\text{simp}]$ :  $t \in \text{atrm}$  **and**  $x[\text{simp}]$ :  $x \in \text{var } x \notin \text{FvarsT } t$   
**shows**  $\text{prv} (\text{dsj} (\text{eql } t \text{ zer})$   
 $(\text{exi } x (\text{eql } t (\text{suc} (\text{Var } x))))))$   
 ⟨*proof*⟩

**lemma** *nprv\_zer\_suc\_casesE*:  
 $\text{nprv} (\text{insert} (\text{eql } t \text{ zer}) F) \varphi \implies \text{nprv} (\text{insert} (\text{eql } t (\text{suc} (\text{Var } x))) F) \varphi \implies$   
 $\text{finite } F \implies F \subseteq \text{fmla} \implies \varphi \in \text{fmla} \implies x \in \text{var} \implies t \in \text{atrm} \implies$   
 $x \notin \text{Fvars} \varphi \implies x \notin \text{FvarsT } t \implies x \notin \bigcup (\text{Fvars } ' F) \implies$   
 $\text{nprv } F \varphi$   
 ⟨*proof*⟩

**lemmas**  $nprv\_zer\_suc\_casesE0 = nprv\_zer\_suc\_casesE[OF\ nprv\_hyp\_ ,\ simped]$   
**lemmas**  $nprv\_zer\_suc\_casesE1 = nprv\_zer\_suc\_casesE[OF\ \_ nprv\_hyp,\ simped]$   
**lemmas**  $nprv\_zer\_suc\_casesE01 = nprv\_zer\_suc\_casesE[OF\ nprv\_hyp\ nprv\_hyp,\ simped]$

**lemma**  $prv\_pls\_zer$ :  
**assumes**  $[simp]: t \in atrm$  **shows**  $prv\ (eql\ (pls\ t\ zer)\ t)$   
 $\langle proof \rangle$

**lemma**  $prv\_pls\_suc$ :  
 $t \in atrm \implies t' \in atrm \implies$   
 $prv\ (eql\ (pls\ t\ (suc\ t'))\ (suc\ (pls\ t\ t')))$   
 $\langle proof \rangle$

**lemma**  $prv\_tms\_zer$ :  
**assumes**  $[simp]: t \in atrm$  **shows**  $prv\ (eql\ (tms\ t\ zer)\ zer)$   
 $\langle proof \rangle$

**lemma**  $prv\_tms\_suc$ :  
 $t \in atrm \implies t' \in atrm \implies$   
 $prv\ (eql\ (tms\ t\ (suc\ t'))\ (pls\ (tms\ t\ t')\ t))$   
 $\langle proof \rangle$

**lemma**  $prv\_suc\_imp\_cong$ :  
**assumes**  $t1[simp]: t1 \in atrm$  **and**  $t2[simp]: t2 \in atrm$   
**shows**  $prv\ (imp\ (eql\ t1\ t2)\ (eql\ (suc\ t1)\ (suc\ t2)))$   
 $\langle proof \rangle$

**lemmas**  $nprv\_suc\_congI = nprv\_addImpLemmaI[OF\ prv\_suc\_imp\_cong,\ simped,\ rotated\ 4]$   
**lemmas**  $nprv\_suc\_congE = nprv\_addImpLemmaE[OF\ prv\_suc\_imp\_cong,\ simped,\ rotated\ 2]$

**lemmas**  $nprv\_suc\_congE0 = nprv\_suc\_congE[OF\ nprv\_hyp\_ ,\ simped]$   
**lemmas**  $nprv\_suc\_congE1 = nprv\_suc\_congE[OF\ \_ nprv\_hyp,\ simped]$   
**lemmas**  $nprv\_suc\_congE01 = nprv\_suc\_congE[OF\ nprv\_hyp\ nprv\_hyp,\ simped]$

**lemma**  $prv\_suc\_cong$ :  
**assumes**  $t1[simp]: t1 \in atrm$  **and**  $t2[simp]: t2 \in atrm$   
**assumes**  $prv\ (eql\ t1\ t2)$   
**shows**  $prv\ (eql\ (suc\ t1)\ (suc\ t2))$   
 $\langle proof \rangle$

**lemma**  $prv\_pls\_imp\_cong$ :  
**assumes**  $t1[simp]: t1 \in atrm$  **and**  $t1'[simp]: t1' \in atrm$   
**and**  $t2[simp]: t2 \in atrm$  **and**  $t2'[simp]: t2' \in atrm$   
**shows**  $prv\ (imp\ (eql\ t1\ t1')\ (imp\ (eql\ t2\ t2')\ (eql\ (pls\ t1\ t2)\ (pls\ t1'\ t2'))))$   
 $\langle proof \rangle$

**lemmas**  $nprv\_pls\_congI = nprv\_addImp2LemmaI[OF\ prv\_pls\_imp\_cong,\ simped,\ rotated\ 6]$

**lemmas**  $nprv\_pls\_congE = nprv\_addImp2LemmaE[OF\ prv\_pls\_imp\_cong, simped, rotated\ 4]$

**lemmas**  $nprv\_pls\_congE0 = nprv\_pls\_congE[OF\ nprv\_hyp\ \_\_, simped]$

**lemmas**  $nprv\_pls\_congE1 = nprv\_pls\_congE[OF\ \_\_ nprv\_hyp\ \_, simped]$

**lemmas**  $nprv\_pls\_congE2 = nprv\_pls\_congE[OF\ \_\_ nprv\_hyp, simped]$

**lemmas**  $nprv\_pls\_congE01 = nprv\_pls\_congE[OF\ nprv\_hyp\ nprv\_hyp\ \_, simped]$

**lemmas**  $nprv\_pls\_congE02 = nprv\_pls\_congE[OF\ nprv\_hyp\ \_ nprv\_hyp, simped]$

**lemmas**  $nprv\_pls\_congE12 = nprv\_pls\_congE[OF\ \_ nprv\_hyp\ nprv\_hyp, simped]$

**lemmas**  $nprv\_pls\_congE012 = nprv\_pls\_congE[OF\ nprv\_hyp\ nprv\_hyp\ nprv\_hyp, simped]$

**lemma**  $prv\_pls\_cong$ :

**assumes**  $t1 \in atrm\ t1' \in atrm\ t2 \in atrm\ t2' \in atrm$

**and**  $prv\ (eql\ t1\ t1')$  **and**  $prv\ (eql\ t2\ t2')$

**shows**  $prv\ (eql\ (pls\ t1\ t2)\ (pls\ t1'\ t2'))$

*<proof>*

**lemma**  $prv\_pls\_congL$ :

$t1 \in atrm \implies t1' \in atrm \implies t2 \in atrm \implies$

$prv\ (eql\ t1\ t1') \implies prv\ (eql\ (pls\ t1\ t2)\ (pls\ t1'\ t2'))$

*<proof>*

**lemma**  $prv\_pls\_congR$ :

$t1 \in atrm \implies t2 \in atrm \implies t2' \in atrm \implies$

$prv\ (eql\ t2\ t2') \implies prv\ (eql\ (pls\ t1\ t2)\ (pls\ t1\ t2'))$

*<proof>*

**lemma**  $nprv\_pls\_cong$ :

**assumes**  $[simp]: t1 \in atrm\ t1' \in atrm\ t2 \in atrm\ t2' \in atrm$

**shows**  $nprv\ \{eql\ t1\ t1',\ eql\ t2\ t2'\}\ (eql\ (pls\ t1\ t2)\ (pls\ t1'\ t2'))$

*<proof>*

**lemma**  $prv\_tms\_imp\_cong$ :

**assumes**  $t1[simp]: t1 \in atrm$  **and**  $t1'[simp]: t1' \in atrm$

**and**  $t2[simp]: t2 \in atrm$  **and**  $t2'[simp]: t2' \in atrm$

**shows**  $prv\ (imp\ (eql\ t1\ t1')$

$(imp\ (eql\ t2\ t2')\ (eql\ (tms\ t1\ t2)\ (tms\ t1'\ t2'))))$

*<proof>*

**lemmas**  $nprv\_tms\_congI = nprv\_addImp2LemmaI[OF\ prv\_tms\_imp\_cong, simped, rotated\ 6]$

**lemmas**  $nprv\_tms\_congE = nprv\_addImp2LemmaE[OF\ prv\_tms\_imp\_cong, simped, rotated\ 4]$

**lemmas**  $nprv\_tms\_congE0 = nprv\_tms\_congE[OF\ nprv\_hyp\ \_\_, simped]$

**lemmas**  $nprv\_tms\_congE1 = nprv\_tms\_congE[OF\ \_\_ nprv\_hyp\ \_, simped]$

**lemmas**  $nprv\_tms\_congE2 = nprv\_tms\_congE[OF\ \_\_ nprv\_hyp, simped]$

**lemmas**  $nprv\_tms\_congE01 = nprv\_tms\_congE[OF\ nprv\_hyp\ nprv\_hyp\ \_, simped]$

**lemmas**  $nprv\_tms\_congE02 = nprv\_tms\_congE[OF\ nprv\_hyp\ \_ nprv\_hyp, simped]$

**lemmas**  $nprv\_tms\_congE12 = nprv\_tms\_congE[OF\ \_ nprv\_hyp\ nprv\_hyp, simped]$

**lemmas**  $nprv\_tms\_congE012 = nprv\_tms\_congE[OF\ nprv\_hyp\ nprv\_hyp\ nprv\_hyp, simped]$

**lemma**  $prv\_tms\_cong$ :

**assumes**  $t1 \in atrm\ t1' \in atrm\ t2 \in atrm\ t2' \in atrm$

**and**  $prv\ (eql\ t1\ t1')$  **and**  $prv\ (eql\ t2\ t2')$

**shows**  $prv\ (eql\ (tms\ t1\ t2)\ (tms\ t1'\ t2'))$

*<proof>*

**lemma**  $nprv\_tms\_cong$ :

**assumes**  $[simp]: t1 \in atrm\ t1' \in atrm\ t2 \in atrm\ t2' \in atrm$

**shows**  $nprv \{eql\ t1\ t1', eql\ t2\ t2'\} (eql\ (tms\ t1\ t2)\ (tms\ t1'\ t2'))$   
 ⟨proof⟩

**lemma** *prv\_tms\_congL*:  
 $t1 \in atrm \implies t1' \in atrm \implies t2 \in atrm \implies$   
 $prv\ (eql\ t1\ t1') \implies prv\ (eql\ (tms\ t1\ t2)\ (tms\ t1'\ t2'))$   
 ⟨proof⟩

**lemma** *prv\_tms\_congR*:  
 $t1 \in atrm \implies t2 \in atrm \implies t2' \in atrm \implies$   
 $prv\ (eql\ t2\ t2') \implies prv\ (eql\ (tms\ t1\ t2)\ (tms\ t1\ t2'))$   
 ⟨proof⟩

## 8.3 Properties Provable in Q

### 8.3.1 General properties, unconstrained by numerals

**lemma** *prv\_pls\_suc\_zer*:  
 $t \in atrm \implies prv\ (eql\ (pls\ t\ (suc\ zer))\ (suc\ t))$   
 ⟨proof⟩

**lemma** *prv\_LLq\_suc\_imp*:  
**assumes** [*simp*]:  $t1 \in atrm\ t2 \in atrm$   
**shows**  $prv\ (imp\ (LLq\ (suc\ t1)\ (suc\ t2))\ (LLq\ t1\ t2))$   
 ⟨proof⟩

**lemmas**  $nprv\_LLq\_sucI = nprv\_addImpLemmaI[OF\ prv\_LLq\_suc\_imp,\ simped,\ rotated\ 4]$   
**lemmas**  $nprv\_LLq\_sucE = nprv\_addImpLemmaE[OF\ prv\_LLq\_suc\_imp,\ simped,\ rotated\ 2]$

**lemmas**  $nprv\_LLq\_sucE0 = nprv\_LLq\_sucE[OF\ nprv\_hyp\ \_,\ simped]$   
**lemmas**  $nprv\_LLq\_sucE1 = nprv\_LLq\_sucE[OF\ \_ nprv\_hyp,\ simped]$   
**lemmas**  $nprv\_LLq\_sucE01 = nprv\_LLq\_sucE[OF\ nprv\_hyp\ nprv\_hyp,\ simped]$

**lemma** *prv\_LLs\_imp\_LLq*:  
**assumes** [*simp*]:  $t1 \in atrm\ t2 \in atrm$   
**shows**  $prv\ (imp\ (LLs\ t1\ t2)\ (LLq\ t1\ t2))$   
 ⟨proof⟩

**lemma** *prv\_LLq\_refl*:  
 $prv\ (LLq\ zer\ zer)$   
 ⟨proof⟩

NB: Monotonicity of pls and tms w.r.t. LLq cannot be proved in Q.

**lemma** *prv\_suc\_mono\_LLq*:  
**assumes**  $t1 \in atrm\ t2 \in atrm$   
**shows**  $prv\ (imp\ (LLq\ t1\ t2)\ (LLq\ (suc\ t1)\ (suc\ t2)))$   
 ⟨proof⟩

**lemmas**  $nprv\_suc\_mono\_LLqI = nprv\_addImpLemmaI[OF\ prv\_suc\_mono\_LLq,\ simped,\ rotated\ 4]$   
**lemmas**  $nprv\_suc\_mono\_LLqE = nprv\_addImpLemmaE[OF\ prv\_suc\_mono\_LLq,\ simped,\ rotated\ 2]$

**lemmas**  $nprv\_suc\_mono\_LLqE0 = nprv\_suc\_mono\_LLqE[OF\ nprv\_hyp\ \_,\ simped]$   
**lemmas**  $nprv\_suc\_mono\_LLqE1 = nprv\_suc\_mono\_LLqE[OF\ \_ nprv\_hyp,\ simped]$   
**lemmas**  $nprv\_suc\_mono\_LLqE01 = nprv\_suc\_mono\_LLqE[OF\ nprv\_hyp\ nprv\_hyp,\ simped]$

### 8.3.2 Representability properties

Representability of number inequality

**lemma** *prv\_neg\_eql\_suc\_Num\_zer*:  
*prv (neg (eql (suc (Num n)) zer))*  
*<proof>*

**lemma** *diff\_prv\_eql\_Num*:  
**assumes**  $m \neq n$   
**shows** *prv (neg (eql (Num m) (Num n)))*  
*<proof>*

**lemma** *consistent\_prv\_eql\_Num\_equal*:  
**assumes** *consistent* **and** *prv (eql (Num m) (Num n))*  
**shows**  $m = n$   
*<proof>*

Representability of addition

**lemma** *prv\_pls\_zer\_zer*:  
*prv (eql (pls zer zer) zer)*  
*<proof>*

**lemma** *prv\_eql\_pls\_plus*:  
*prv (eql (pls (Num m) (Num n))*  
*(Num (m+n)))*  
*<proof>*

**lemma** *not\_plus\_prv\_neg\_eql\_pls*:  
**assumes**  $m + n \neq k$   
**shows** *prv (neg (eql (pls (Num m) (Num n)) (Num k)))*  
*<proof>*

**lemma** *consistent\_prv\_eql\_pls\_plus\_rev*:  
**assumes** *consistent* *prv (eql (pls (Num m) (Num n)) (Num k))*  
**shows**  $k = m + n$   
*<proof>*

Representability of multiplication

**lemma** *prv\_tms\_Num\_zer*:  
*prv (eql (tms (Num n) zer) zer)*  
*<proof>*

**lemma** *prv\_eql\_tms\_times*:  
*prv (eql (tms (Num m) (Num n)) (Num (m \* n)))*  
*<proof>*

**lemma** *ge\_prv\_neg\_eql\_pls\_Num\_zer*:  
**assumes** [*simp*]:  $t \in \text{atrm}$  **and**  $m > k$   
**shows** *prv (neg (eql (pls t (Num m)) (Num k)))*  
*<proof>*

**lemma** *nprv\_pls\_Num\_injectR*:  
**assumes** [*simp*]:  $t1 \in \text{atrm}$   $t2 \in \text{atrm}$   
**shows** *prv (imp (eql (pls t1 (Num m)) (pls t2 (Num m)))*  
*(eql t1 t2))*  
*<proof>*

**lemmas** *nprv\_pls\_Num\_injectI* = *nprv\_addImpLemmaI*[*OF nprv\_pls\_Num\_injectR, simped, rotated 4*]

**lemmas** *nprv\_pls\_Num\_injectE* = *nprv\_addImpLemmaE*[*OF nprv\_pls\_Num\_injectR, simped, rotated 2*]

**lemmas** *nprv\_pls\_Num\_injectE0* = *nprv\_pls\_Num\_injectE*[*OF nprv\_hyp \_, simped*]

**lemmas** *nprv\_pls\_Num\_injectE1* = *nprv\_pls\_Num\_injectE*[*OF \_ nprv\_hyp, simped*]

**lemmas** *nprv\_pls\_Num\_injectE01* = *nprv\_pls\_Num\_injectE*[*OF nprv\_hyp nprv\_hyp, simped*]

**lemma** *not\_times\_prv\_neg\_eql\_tms*:

**assumes**  $m * n \neq k$

**shows** *prv (neg (eql (tms (Num m) (Num n)) (Num k)))*

*<proof>*

**lemma** *consistent\_prv\_eql\_tms\_times\_rev*:

**assumes** *consistent prv (eql (tms (Num m) (Num n)) (Num k))*

**shows**  $k = m * n$

*<proof>*

Representability of the order

**lemma** *leq\_prv\_LLq\_Num*:

**assumes**  $m \leq n$

**shows** *prv (LLq (Num m) (Num n))*

*<proof>*

### 8.3.3 The "order-adequacy" properties

These are properties Q1–O9 from Peter Smith, An Introduction to Gödel's theorems, Second Edition, Page 73.

**lemma** *prv\_LLq\_zer*: — O1

**assumes** [*simp*]:  $t \in \text{atrm}$

**shows** *prv (LLq zer t)*

*<proof>*

**lemmas** *Q1* = *prv\_LLq\_zer*

**lemma** *prv\_LLq\_zer\_imp\_eql*:

**assumes** [*simp*]:  $t \in \text{atrm}$

**shows** *prv (imp (LLq t zer) (eql t zer))*

*<proof>*

**lemmas** *nprv\_LLq\_zer\_eqlI* = *nprv\_addImpLemmaI*[*OF prv\_LLq\_zer\_imp\_eql, simped, rotated 3*]

**lemmas** *nprv\_LLq\_zer\_eqlE* = *nprv\_addImpLemmaE*[*OF prv\_LLq\_zer\_imp\_eql, simped, rotated 1*]

**lemmas** *nprv\_LLq\_zer\_eqlE0* = *nprv\_LLq\_zer\_eqlE*[*OF nprv\_hyp \_, simped*]

**lemmas** *nprv\_LLq\_zer\_eqlE1* = *nprv\_LLq\_zer\_eqlE*[*OF \_ nprv\_hyp, simped*]

**lemmas** *nprv\_LLq\_zer\_eqlE01* = *nprv\_LLq\_zer\_eqlE*[*OF nprv\_hyp nprv\_hyp, simped*]

**lemma** *prv\_sdsj\_eql\_imp\_LLq*: — O2

**assumes** [*simp*]:  $t \in \text{atrm}$

**shows** *prv (imp (lds (map ( $\lambda i. \text{eql } t \text{ (Num } i)) \text{ (toN } n))) \text{ (LLq } t \text{ (Num } n)))$*

*<proof>*

**declare** *subset\_eq*[simp]  
**lemmas** *nprv\_sdsj\_eql\_LLqI* = *nprv\_addImpLemmaI*[OF *nprv\_sdsj\_eql\_imp\_LLq*, *simped*, *rotated* 3]  
**lemmas** *nprv\_sdsj\_eql\_LLqE* = *nprv\_addImpLemmaE*[OF *nprv\_sdsj\_eql\_imp\_LLq*, *simped*, *rotated* 1]  
**declare** *subset\_eq*[simp del]

**lemmas** *nprv\_sdsj\_eql\_LLqE0* = *nprv\_sdsj\_eql\_LLqE*[OF *nprv\_hyp* \_, *simped*]  
**lemmas** *nprv\_sdsj\_eql\_LLqE1* = *nprv\_sdsj\_eql\_LLqE*[OF \_ *nprv\_hyp*, *simped*]  
**lemmas** *nprv\_sdsj\_eql\_LLqE01* = *nprv\_sdsj\_eql\_LLqE*[OF *nprv\_hyp* *nprv\_hyp*, *simped*]

**lemmas** *O2I* = *nprv\_sdsj\_eql\_LLqI*  
**lemmas** *O2E* = *nprv\_sdsj\_eql\_LLqE*  
**lemmas** *O2E0* = *nprv\_sdsj\_eql\_LLqE0*  
**lemmas** *O2E1* = *nprv\_sdsj\_eql\_LLqE1*  
**lemmas** *O2E01* = *nprv\_sdsj\_eql\_LLqE01*

**lemma** *prv\_LLq\_imp\_sdsj\_eql*: — O3  
**assumes** [simp]:  $t \in \text{atrm}$   
**shows** *prv* (*imp* (*LLq*  $t$  (*Num*  $n$ )) (*lds**j* (*map* ( $\lambda i.$  *eql*  $t$  (*Num*  $i$ )) (*toN*  $n$ ))))  
*<proof>*

**declare** *subset\_eq*[simp]  
**lemmas** *prv\_LLq\_sdsj\_eqlI* = *nprv\_addImpLemmaI*[OF *prv\_LLq\_imp\_sdsj\_eql*, *simped*, *rotated* 3]  
**lemmas** *prv\_LLq\_sdsj\_eqlE* = *nprv\_addImpLemmaE*[OF *prv\_LLq\_imp\_sdsj\_eql*, *simped*, *rotated* 1]  
**declare** *subset\_eq*[simp del]

**lemmas** *prv\_LLq\_sdsj\_eqlE0* = *prv\_LLq\_sdsj\_eqlE*[OF *nprv\_hyp* \_, *simped*]  
**lemmas** *prv\_LLq\_sdsj\_eqlE1* = *prv\_LLq\_sdsj\_eqlE*[OF \_ *nprv\_hyp*, *simped*]  
**lemmas** *prv\_LLq\_sdsj\_eqlE01* = *prv\_LLq\_sdsj\_eqlE*[OF *nprv\_hyp* *nprv\_hyp*, *simped*]

**lemmas** *O3I* = *prv\_LLq\_sdsj\_eqlI*  
**lemmas** *O3E* = *prv\_LLq\_sdsj\_eqlE*  
**lemmas** *O3E0* = *prv\_LLq\_sdsj\_eqlE0*  
**lemmas** *O3E1* = *prv\_LLq\_sdsj\_eqlE1*  
**lemmas** *O3E01* = *prv\_LLq\_sdsj\_eqlE01*

**lemma** *not\_leq\_prv\_neg\_LLq\_Num*:  
**assumes**  $\neg m \leq n$   
**shows** *prv* (*neg* (*LLq* (*Num*  $m$ ) (*Num*  $n$ )))  
*<proof>*

**lemma** *consistent\_prv\_LLq\_Num\_leq*:  
**assumes** *consistent* *prv* (*LLq* (*Num*  $m$ ) (*Num*  $n$ ))  
**shows**  $m \leq n$   
*<proof>*

**lemma** *prv\_ball\_NumI*: — O4  
**assumes** [simp]:  $x \in \text{var}$   $\varphi \in \text{fmla}$   
**and** [simp]:  $\bigwedge i. i \leq n \implies \text{prv} (\text{subst } \varphi (\text{Num } i) x)$   
**shows** *prv* (*ball*  $x$  (*Num*  $n$ )  $\varphi$ )  
*<proof>*

**lemmas** *O4* = *prv\_ball\_NumI*

**lemma** *prv\_bexi\_NumI*: — O5

**assumes**  $[simp]: x \in var \ \varphi \in fmla$   
**and**  $[simp]: i \leq n \ \text{prv} \ (subst \ \varphi \ (Num \ i) \ x)$   
**shows**  $\text{prv} \ (bexi \ x \ (Num \ n) \ \varphi)$   
 $\langle proof \rangle$

**lemmas**  $O5 = \text{prv\_bexi\_NumI}$

**lemma**  $\text{prv\_LLq\_Num\_imp\_Suc}$ : — O6  
**assumes**  $[simp]: t \in atrm$   
**shows**  $\text{prv} \ (imp \ (LLq \ t \ (Num \ n)) \ (LLq \ t \ (suc \ (Num \ n))))$   
 $\langle proof \rangle$

**lemmas**  $\text{nprv\_LLq\_Num\_SucI} = \text{nprv\_addImpLemmaI}[OF \ \text{prv\_LLq\_Num\_imp\_Suc}, \ \text{simped}, \ \text{rotated} \ 3]$

**lemmas**  $\text{nprv\_LLq\_Num\_SucE} = \text{nprv\_addImpLemmaE}[OF \ \text{prv\_LLq\_Num\_imp\_Suc}, \ \text{simped}, \ \text{rotated} \ 1]$

**lemmas**  $\text{nprv\_LLq\_Num\_SucE0} = \text{nprv\_LLq\_Num\_SucE}[OF \ \text{nprv\_hyp} \ \_, \ \text{simped}]$   
**lemmas**  $\text{nprv\_LLq\_Num\_SucE1} = \text{nprv\_LLq\_Num\_SucE}[OF \ \_ \ \text{nprv\_hyp}, \ \text{simped}]$   
**lemmas**  $\text{nprv\_LLq\_Num\_SucE01} = \text{nprv\_LLq\_Num\_SucE}[OF \ \text{nprv\_hyp} \ \text{nprv\_hyp}, \ \text{simped}]$

**lemmas**  $O6I = \text{nprv\_LLq\_Num\_SucI}$   
**lemmas**  $O6E = \text{nprv\_LLq\_Num\_SucE}$   
**lemmas**  $O6E0 = \text{nprv\_LLq\_Num\_SucE0}$   
**lemmas**  $O6E1 = \text{nprv\_LLq\_Num\_SucE1}$   
**lemmas**  $O6E01 = \text{nprv\_LLq\_Num\_SucE01}$

Crucial for proving O7:

**lemma**  $\text{prv\_LLq\_suc\_Num\_pls\_Num}$ :  
**assumes**  $[simp]: t \in atrm$   
**shows**  $\text{prv} \ (LLq \ (suc \ (Num \ n)) \ (pls \ (suc \ t) \ (Num \ n)))$   
 $\langle proof \rangle$

**lemma**  $\text{prv\_Num\_LLq\_imp\_eql\_suc}$ : — O7  
**assumes**  $[simp]: t \in atrm$   
**shows**  $\text{prv} \ (imp \ (LLq \ (Num \ n) \ t) \ (dsj \ (eql \ (Num \ n) \ t) \ (LLq \ (suc \ (Num \ n)) \ t)))$   
 $\langle proof \rangle$

**lemma**  $\text{prv\_Num\_LLq\_eql\_sucE}$ :  
 $\text{nprv} \ F \ (LLq \ (Num \ n) \ t) \implies$   
 $\text{nprv} \ (insert \ (eql \ (Num \ n) \ t) \ F) \ \psi \implies$   
 $\text{nprv} \ (insert \ (LLq \ (suc \ (Num \ n)) \ t) \ F) \ \psi \implies$   
 $t \in atrm \implies F \subseteq fmla \implies \text{finite} \ F \implies \psi \in fmla \implies$   
 $\text{nprv} \ F \ \psi$   
 $\langle proof \rangle$

**lemmas**  $\text{prv\_Num\_LLq\_eql\_sucE0} = \text{prv\_Num\_LLq\_eql\_sucE}[OF \ \text{nprv\_hyp} \ \_, \ \text{simped}]$   
**lemmas**  $\text{prv\_Num\_LLq\_eql\_sucE1} = \text{prv\_Num\_LLq\_eql\_sucE}[OF \ \_ \ \text{nprv\_hyp} \ \_, \ \text{simped}]$   
**lemmas**  $\text{prv\_Num\_LLq\_eql\_sucE2} = \text{prv\_Num\_LLq\_eql\_sucE}[OF \ \_ \ \_ \ \text{nprv\_hyp}, \ \text{simped}]$   
**lemmas**  $\text{prv\_Num\_LLq\_eql\_sucE01} = \text{prv\_Num\_LLq\_eql\_sucE}[OF \ \text{nprv\_hyp} \ \text{nprv\_hyp} \ \_, \ \text{simped}]$   
**lemmas**  $\text{prv\_Num\_LLq\_eql\_sucE02} = \text{prv\_Num\_LLq\_eql\_sucE}[OF \ \text{nprv\_hyp} \ \_ \ \text{nprv\_hyp}, \ \text{simped}]$   
**lemmas**  $\text{prv\_Num\_LLq\_eql\_sucE12} = \text{prv\_Num\_LLq\_eql\_sucE}[OF \ \_ \ \text{nprv\_hyp} \ \text{nprv\_hyp}, \ \text{simped}]$   
**lemmas**  $\text{prv\_Num\_LLq\_eql\_sucE012} = \text{prv\_Num\_LLq\_eql\_sucE}[OF \ \text{nprv\_hyp} \ \text{nprv\_hyp} \ \text{nprv\_hyp}, \ \text{simped}]$



**lemmas**  $O7E = prv\_Num\_LLq\_eql\_sucE$   
**lemmas**  $O7E0 = prv\_Num\_LLq\_eql\_sucE0$

**lemma**  $prv\_dsj\_eql\_Num\_neg$ :  
**assumes**  $t \in atrm$   
**shows**  $prv (dsj (eql t (Num n)) (neg (eql t (Num n))))$   
 $\langle proof \rangle$

**lemmas**  $nprv\_eql\_Num\_casesE = nprv\_addDsjLemmaE[OF prv\_dsj\_eql\_Num\_neg, simped, rotated]$

**lemmas**  $nprv\_eql\_Num\_casesE0 = nprv\_eql\_Num\_casesE[OF nprv\_hyp \_, simped]$   
**lemmas**  $nprv\_eql\_Num\_casesE1 = nprv\_eql\_Num\_casesE[OF \_ nprv\_hyp, simped]$   
**lemmas**  $nprv\_eql\_Num\_casesE01 = nprv\_eql\_Num\_casesE[OF nprv\_hyp nprv\_hyp, simped]$

**lemma**  $prv\_LLq\_Num\_dsj$ : — O8  
**assumes**  $[simp]: t \in atrm$   
**shows**  $prv (dsj (LLq t (Num n)) (LLq (Num n) t))$   
 $\langle proof \rangle$

**lemma**  $prv\_LLq\_Num\_casesE$ :  
 $nprv (insert (LLq t (Num n)) F) \psi \implies$   
 $nprv (insert (LLq (Num n) t) F) \psi \implies$   
 $t \in atrm \implies F \subseteq fmla \implies finite F \implies \psi \in fmla \implies$   
 $nprv F \psi$   
 $\langle proof \rangle$

**lemmas**  $prv\_LLq\_Num\_casesE0 = prv\_LLq\_Num\_casesE[OF nprv\_hyp \_, simped]$   
**lemmas**  $prv\_LLq\_Num\_casesE1 = prv\_LLq\_Num\_casesE[OF \_ nprv\_hyp, simped]$   
**lemmas**  $prv\_LLq\_Num\_casesE01 = prv\_LLq\_Num\_casesE[OF nprv\_hyp nprv\_hyp, simped]$

**lemmas**  $O8E = prv\_LLq\_Num\_casesE$   
**lemmas**  $O8E0 = prv\_LLq\_Num\_casesE0$   
**lemmas**  $O8E1 = prv\_LLq\_Num\_casesE1$   
**lemmas**  $O8E01 = prv\_LLq\_Num\_casesE01$

**lemma**  $prv\_imp\_LLq\_neg\_Num\_suc$ :  
**assumes**  $[simp]: t \in atrm$   
**shows**  $prv (imp (LLq t (suc (Num n)))$   
 $(imp ((neg (eql t (suc (Num n))))$   
 $(LLq t (Num n))))$   
 $\langle proof \rangle$

**lemmas**  $nprv\_LLq\_neg\_Num\_sucI = nprv\_addImp2LemmaI[OF prv\_imp\_LLq\_neg\_Num\_suc, simped, rotated 3]$   
**lemmas**  $nprv\_LLq\_neg\_Num\_sucE = nprv\_addImp2LemmaE[OF prv\_imp\_LLq\_neg\_Num\_suc, simped, rotated 1]$

**lemmas**  $nprv\_LLq\_neg\_Num\_sucE0 = nprv\_LLq\_neg\_Num\_sucE[OF nprv\_hyp \_, simped]$   
**lemmas**  $nprv\_LLq\_neg\_Num\_sucE1 = nprv\_LLq\_neg\_Num\_sucE[OF \_ nprv\_hyp \_, simped]$

**lemmas**  $nprv\_LLq\_neg\_Num\_sucE2 = nprv\_LLq\_neg\_Num\_sucE[OF \_ \_ nprv\_hyp, simped]$   
**lemmas**  $nprv\_LLq\_neg\_Num\_sucE01 = nprv\_LLq\_neg\_Num\_sucE[OF nprv\_hyp nprv\_hyp \_, simped]$   
**lemmas**  $nprv\_LLq\_neg\_Num\_sucE02 = nprv\_LLq\_neg\_Num\_sucE[OF nprv\_hyp \_ nprv\_hyp, simped]$   
**lemmas**  $nprv\_LLq\_neg\_Num\_sucE12 = nprv\_LLq\_neg\_Num\_sucE[OF \_ nprv\_hyp nprv\_hyp, simped]$   
**lemmas**  $nprv\_LLq\_neg\_Num\_sucE012 = nprv\_LLq\_neg\_Num\_sucE[OF nprv\_hyp nprv\_hyp nprv\_hyp, simped]$

**lemma**  $prv\_ball\_Num\_imp\_ball\_suc$ : — O9  
**assumes**  $[simp]: x \in var \ \varphi \in fmla$   
**shows**  $prv (imp (ball x (Num n) \varphi) (ball x (suc (Num n)) (imp (neg (eql (Var x) (suc (Num n)))) \varphi)))$   
 $\langle proof \rangle$

**lemmas**  $prv\_ball\_Num\_ball\_sucI = nprv\_addImpLemmaI[OF prv\_ball\_Num\_imp\_ball\_suc, simped, rotated 4]$   
**lemmas**  $prv\_ball\_Num\_ball\_sucE = nprv\_addImpLemmaE[OF prv\_ball\_Num\_imp\_ball\_suc, simped, rotated 2]$

**lemmas**  $prv\_ball\_Num\_ball\_sucE0 = prv\_ball\_Num\_ball\_sucE[OF nprv\_hyp \_, simped]$   
**lemmas**  $prv\_ball\_Num\_ball\_sucE1 = prv\_ball\_Num\_ball\_sucE[OF \_ nprv\_hyp, simped]$   
**lemmas**  $prv\_ball\_Num\_ball\_sucE01 = prv\_ball\_Num\_ball\_sucE[OF nprv\_hyp nprv\_hyp, simped]$

**lemmas**  $O9I = prv\_ball\_Num\_ball\_sucI$   
**lemmas**  $O9E = prv\_ball\_Num\_ball\_sucE$   
**lemmas**  $O9E0 = prv\_ball\_Num\_ball\_sucE0$   
**lemmas**  $O9E1 = prv\_ball\_Num\_ball\_sucE1$   
**lemmas**  $O9E01 = prv\_ball\_Num\_ball\_sucE01$

### 8.3.4 Verifying the abstract ordering assumptions

**lemma**  $LLq\_num$ :  
**assumes**  $\varphi[simp]: \varphi \in fmla \ Fvars \ \varphi = \{zz\}$  **and**  $q: q \in num$  **and**  $p: \forall p \in num. prv (subst \varphi p zz)$   
**shows**  $prv (all zz (imp (LLq (Var zz) q) \varphi))$   
 $\langle proof \rangle$

**lemma**  $LLq\_num2$ :  
**assumes**  $p \in num$   
**shows**  $\exists P \subseteq num. finite P \wedge prv (dsj (sdsj \{eql (Var yy) r \mid r. r \in P\}) (LLq p (Var yy)))$   
 $\langle proof \rangle$

**end** — context  $Deduct\_Q$

**sublocale**  $Deduct\_Q < lab: Deduct\_with\_PseudoOrder$  **where**  $Lq = LLq (Var zz) (Var yy)$   
 $\langle proof \rangle$

# Bibliography

- [1] A. Popescu and D. Traytel. A formally verified abstract account of Gödel's incompleteness theorems. In P. Fontaine, editor, *CADE 27*, volume 11716 of *LNCS*, pages 442–461. Springer, 2019.