

Syntax-Independent Logic Infrastructure

Andrei Popescu Dmitriy Traytel

February 23, 2021

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

1 Preliminaries	3
1.1 Trivia	3
1.2 Some Proof Infrastructure	4
2 Syntax	6
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
3 Deduction	26
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 ω -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

4	Natural Deduction	51
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
5	Pseudo-Terms	63
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
6	Truth in a Standard Model	69
7	Arithmetic Constructs	72
7.1	Arithmetic Terms	74
7.2	The (Nonstrict and Strict) Order Relations	80
7.3	Bounded Quantification	82
8	Deduction in a System Embedding the Intuitionistic Robinson Arithmetic	84
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* =
 $((\text{meth}; \text{fail})+)?$

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* =
 $(\text{meth}?, (\text{subst } s1)?, \text{meth}?)$

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

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

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

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

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

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!]: $\bigwedge x. x \in var \Longrightarrow Var\ x \in trm$

and

inj_Var[simp]: $\bigwedge x\ y. x \in var \Longrightarrow y \in var \Longrightarrow (Var\ x = Var\ y \longleftrightarrow x = y)$

and

finite_FvarsT: $\bigwedge t. t \in trm \Longrightarrow finite\ (FvarsT\ t)$

and

FvarsT: $\bigwedge t. t \in trm \Longrightarrow FvarsT\ t \subseteq var$

and

substT[simp,intro]: $\bigwedge t1\ t\ x. t1 \in trm \Longrightarrow t \in trm \Longrightarrow x \in var \Longrightarrow substT\ t1\ t\ x \in trm$

and

FvarsT_Var[simp]: $\bigwedge x. x \in var \Longrightarrow FvarsT\ (Var\ x) = \{x\}$

and

substT_Var[simp]: $\bigwedge x\ t\ y. x \in var \Longrightarrow y \in var \Longrightarrow t \in trm \Longrightarrow$

substT (Var x) t y = (if x = y then t else Var x)

and

substT_notIn[simp]:

$\bigwedge t1\ t2\ x. x \in var \Longrightarrow t1 \in trm \Longrightarrow t2 \in trm \Longrightarrow x \notin FvarsT\ t1 \Longrightarrow substT\ t1\ t2\ x = t1$

and

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

finite_Fvars: $\bigwedge \varphi. \varphi \in fmla \Longrightarrow finite\ (Fvars\ \varphi)$

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

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

<proof>

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

lemma *getFresh*: finite V \implies *getFresh* V \in var \wedge *getFresh* V \notin V
<proof>

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

lemma *getFr_FvarsT_Fvars*:
assumes set xs \subseteq var set ts \subseteq trm **and** set φ s \subseteq fmla
shows *getFr* xs ts φ s \in var \wedge
 getFr xs ts φ s \notin set xs \wedge
 (t \in set ts \longrightarrow *getFr* xs ts φ s \notin FvarsT t) \wedge
 (φ \in set φ s \longrightarrow *getFr* xs ts φ s \notin Fvars φ)
<proof>

lemma *getFr[simp,intro]*:
assumes set xs \subseteq var set ts \subseteq trm **and** set φ s \subseteq fmla
shows *getFr* xs ts φ s \in var
<proof>

lemma *getFr_var*:
assumes set xs \subseteq var set ts \subseteq trm **and** set φ s \subseteq fmla **and** t \in set ts
shows *getFr* xs ts φ s \notin set xs
<proof>

lemma *getFr_FvarsT*:
assumes set xs \subseteq var set ts \subseteq trm **and** set φ s \subseteq fmla **and** t \in set ts
shows *getFr* xs ts φ s \notin FvarsT t
<proof>

lemma *getFr_Fvars*:
assumes set xs \subseteq var set ts \subseteq trm **and** set φ s \subseteq fmla **and** φ \in set φ s
shows *getFr* xs ts φ s \notin Fvars φ
<proof>

2.1.2 Fresh Variables

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

lemma *getFreshN*: finite V \implies
 set (*getFreshN* V n) \subseteq var \wedge set (*getFreshN* V n) \cap V = {} \wedge length (*getFreshN* V n) = n \wedge distinct
 (*getFreshN* V n)
<proof>

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

lemma *getFrN_FvarsT_Fvars*:
assumes set xs \subseteq var set ts \subseteq trm **and** set φ s \subseteq fmla
shows set (*getFrN* xs ts φ s n) \subseteq 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_rawpsubst*:

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 $\text{Fvars} (\text{rawpsubst } \varphi \text{ txs}) =$
 $(\text{Fvars } \varphi - \text{snd} \text{ ' (set txs)}) \cup$
 $(\bigcup \{ \text{if } x \in \text{Fvars } \varphi \text{ then } \text{FvarsT } t \text{ else } \{ \} \mid t \text{ x . } (t,x) \in \text{set txs} \})$
<proof>

lemma *in_Fvars_rawpsubstD*:

assumes $y \in \text{Fvars} (\text{rawpsubst } \varphi \text{ txs})$
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 *rawpsubst2_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 $\text{rawpsubst } \varphi ((s,y),(t,x)) = \text{rawpsubst } \varphi ((t,x),(s,y))$
<proof>

lemma *rawpsubst_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 $\text{rawpsubst} (\text{rawpsubst } \varphi \text{ txs1}) \text{ txs2} = \text{rawpsubst } \varphi (\text{txs1} @ \text{txs2})$
<proof>

lemma *rawpsubst_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 rawpsubst can replace 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 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_numI[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 $\text{snd} \text{ ' (set txs) } \subseteq \text{var fst ' (set txs) } \subseteq \text{trm}$
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 $\text{snd} \text{ ' (set txs) } \subseteq \text{var fst ' (set txs) } \subseteq \text{trm}$
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 $\text{snd} \text{ ' (set txs) } \subseteq \text{var fst ' (set txs) } \subseteq \text{trm}$
and $y \notin \text{snd} \text{ ' (set txs) } \cup (FvarsT \text{ ' fst ' (set txs)})$
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 $\text{snd} \text{ ' (set txs) } \subseteq \text{var fst ' (set txs) } \subseteq \text{trm}$
and $y \notin \text{snd} \text{ ' (set txs) } \cup (FvarsT \text{ ' fst ' (set txs)})$
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 $\text{snd} \text{ ' (set txs) } \subseteq \text{var fst ' (set txs) } \subseteq \text{trm}$
and $y \notin \text{snd} \text{ ' (set txs) } \cup (FvarsT \text{ ' fst ' (set txs)})$
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 $\text{snd} \text{ ' (set txs) } \subseteq \text{var fst ' (set txs) } \subseteq \text{trm}$
and $y \notin \text{snd} \text{ ' (set txs) } \cup (FvarsT \text{ ' fst ' (set txs)})$
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

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

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
 $subst_tru[simp]: \bigwedge t x. t \in trm \implies x \in var \implies subst tru t x = tru$
 $\langle proof \rangle$

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

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

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

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

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

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

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

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  $\varphi s \subseteq fmla \Longrightarrow$  ldsj  $\varphi s \in fmla$ 
  <proof>
```

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

```
lemma subst_ldsj[simp]:
  set  $\varphi s \subseteq fmla \Longrightarrow$   $t \in trm \Longrightarrow$   $x \in var \Longrightarrow$ 
  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]:  $F \subseteq fmla \Longrightarrow$  finite  $F \Longrightarrow$  sdsj  $F \in fmla$ 
  <proof>
```

```
lemma Fvars_sdsj[simp]:
   $F \subseteq fmla \Longrightarrow$  finite  $F \Longrightarrow$  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  $\varphi s \subseteq fmla \Longrightarrow$   $snd$  ' (set  $txs$ )  $\subseteq$  var  $\Longrightarrow$   $fst$  ' (set  $txs$ )  $\subseteq$  trm  $\Longrightarrow$ 
  distinct (map  $snd$   $txs$ )  $\Longrightarrow$ 
  psubst (ldsjs  $\varphi s$ )  $txs$  = 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: $t1 \in trm \implies t2 \in trm \implies yy \notin FvarsT\ t1 \implies$
 $LLq\ t1\ t2 = subst\ (subst\ Lq\ t1\ zz)\ t2\ yy$
 ⟨proof⟩

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

lemma LLq2[simp,intro!]:
 $n \in num \implies LLq\ n\ (Var\ yy') \in fmla$
 ⟨proof⟩

lemma Fvars_LLq[simp]: $t1 \in trm \implies t2 \in trm \implies yy \notin FvarsT\ t1 \implies$

$Fvars (LLq\ t1\ t2) = FvarsT\ t1 \cup FvarsT\ t2$
 ⟨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 φ) (*exi* y (*all* x (*imp* φ (*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  $\longrightarrow, \wedge, \forall, \exists$ . ( $\perp$ ,  $\neg$  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:
   $\bigwedge \varphi \chi. \varphi \in \text{fmla} \Longrightarrow \chi \in \text{fmla} \Longrightarrow$ 
     $\text{prv} (\text{imp } \varphi \chi) \Longrightarrow \text{prv } \varphi \Longrightarrow \text{prv } \chi$ 
  and
  — The propositional intuitionistic axioms:
  prv_imp_imp_triv:
   $\bigwedge \varphi \chi. \varphi \in \text{fmla} \Longrightarrow \chi \in \text{fmla} \Longrightarrow$ 
     $\text{prv} (\text{imp } \varphi (\text{imp } \chi \varphi))$ 
  and
  prv_imp_trans:
   $\bigwedge \varphi \chi \psi. \varphi \in \text{fmla} \Longrightarrow \chi \in \text{fmla} \Longrightarrow \psi \in \text{fmla} \Longrightarrow$ 
     $\text{prv} (\text{imp} (\text{imp } \varphi (\text{imp } \chi \psi))$ 
       $(\text{imp} (\text{imp } \varphi \chi) (\text{imp } \varphi \psi)))$ 
  and
  prv_imp_cnjL:
   $\bigwedge \varphi \chi. \varphi \in \text{fmla} \Longrightarrow \chi \in \text{fmla} \Longrightarrow$ 
```

```

    prv (imp (cnj  $\varphi$   $\chi$ )  $\varphi$ )
and
prv_imp_cnjR:
 $\bigwedge \varphi \chi. \varphi \in fmla \implies \chi \in fmla \implies$ 
    prv (imp (cnj  $\varphi$   $\chi$ )  $\chi$ )
and
prv_imp_cnjI:
 $\bigwedge \varphi \chi. \varphi \in fmla \implies \chi \in fmla \implies$ 
    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:
 $\bigwedge x \varphi \chi. x \notin Fvars \varphi \implies prv (imp \varphi \chi) \implies prv (imp \varphi (all\ x \chi))$ 
and
prv_exi_imp_gen:
 $\bigwedge x \varphi \chi. x \in var \implies \varphi \in fmla \implies \chi \in fmla \implies$ 
 $x \notin Fvars \chi \implies prv (imp \varphi \chi) \implies prv (imp (exi\ x \varphi) \chi)$ 
and
— Two quantifier instantiation axioms:
prv_all_inst:
 $\bigwedge x \varphi t.$ 
 $x \in var \implies \varphi \in fmla \implies t \in trm \implies$ 
    prv (imp (all  $x \varphi$ ) (subst  $\varphi$   $t$   $x$ ))
and
prv_exi_inst:
 $\bigwedge x \varphi t.$ 
 $x \in var \implies \varphi \in fmla \implies t \in trm \implies$ 
    prv (imp (subst  $\varphi$   $t$   $x$ ) (exi  $x \varphi$ ))
and
— The equality axioms:
prv_eql_refl:
 $\bigwedge x. x \in var \implies$ 
    prv (eql (Var  $x$ ) (Var  $x$ ))
and
prv_eql_subst:
 $\bigwedge \varphi x y.$ 
 $x \in var \implies y \in var \implies \varphi \in fmla \implies$ 
    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  $\phi$ :  $\varphi \in fmla$  and  $\psi$ :  $\psi \in fmla$ 
shows  $prv \psi \implies prv (imp \varphi \psi)$ 
  <proof>

```

```

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

```

```

lemma prv_imp_refl2:  $\varphi \in fmla \implies \psi \in fmla \implies \varphi = \psi \implies prv (imp \varphi \psi)$ 
  <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

$prv (imp (imp \varphi (imp \chi \psi))$
 $(imp (cnj \varphi \chi) \psi))$
 $\langle proof \rangle$

lemma *prv_imp_cnj_imp*:
assumes $\varphi \in fmla$ **and** $\chi \in fmla$ **and** $\psi \in fmla$
shows
 $prv (imp (imp (cnj \varphi \chi) \psi)$
 $(imp \varphi (imp \chi \psi)))$
 $\langle proof \rangle$

lemma *prv_cnj_imp*:
assumes $\varphi \in fmla$ **and** $\chi \in fmla$ **and** $\psi \in fmla$
assumes $prv (imp (cnj \varphi \chi) \psi)$
shows $prv (imp \varphi (imp \chi \psi))$
 $\langle proof \rangle$

Monotonicity of conjunction w.r.t. implication:

lemma *prv_cnj_imp_monoR*:
assumes $\varphi \in fmla$ **and** $\chi \in fmla$ **and** $\psi \in fmla$
shows $prv (imp (imp \varphi \chi) (imp (imp \varphi \psi) (imp \varphi (cnj \chi \psi))))$
 $\langle proof \rangle$

lemma *prv_imp_cnj3L*:
assumes $\varphi1 \in fmla$ **and** $\varphi2 \in fmla$ **and** $\chi \in fmla$
shows $prv (imp (imp \varphi1 \chi) (imp (cnj \varphi1 \varphi2) \chi))$
 $\langle proof \rangle$

lemma *prv_imp_cnj3R*:
assumes $\varphi1 \in fmla$ **and** $\varphi2 \in fmla$ **and** $\chi \in fmla$
shows $prv (imp (imp \varphi2 \chi) (imp (cnj \varphi1 \varphi2) \chi))$
 $\langle proof \rangle$

lemma *prv_cnj_imp_mono*:
assumes $\varphi1 \in fmla$ **and** $\varphi2 \in fmla$ **and** $\chi1 \in fmla$ **and** $\chi2 \in fmla$
shows $prv (imp (imp \varphi1 \chi1) (imp (imp \varphi2 \chi2) (imp (cnj \varphi1 \varphi2) (cnj \chi1 \chi2))))$
 $\langle proof \rangle$

lemma *prv_cnj_mono*:
assumes $\varphi1 \in fmla$ **and** $\varphi2 \in fmla$ **and** $\chi1 \in fmla$ **and** $\chi2 \in fmla$
assumes $prv (imp \varphi1 \chi1)$ **and** $prv (imp \varphi2 \chi2)$
shows $prv (imp (cnj \varphi1 \varphi2) (cnj \chi1 \chi2))$
 $\langle proof \rangle$

lemma *prv_cnj_imp_monoR4*:
assumes $\varphi \in fmla$ **and** $\chi \in fmla$ **and** $\psi1 \in fmla$ **and** $\psi2 \in fmla$
shows
 $prv (imp (imp \varphi (imp \chi \psi1))$
 $(imp (imp \varphi (imp \chi \psi2)) (imp \varphi (imp \chi (cnj \psi1 \psi2))))$
 $\langle proof \rangle$

lemma *prv_imp_cnj4*:
assumes $\varphi \in fmla$ **and** $\chi \in fmla$ **and** $\psi1 \in fmla$ **and** $\psi2 \in fmla$
shows $prv (imp \varphi (imp \chi \psi1)) \implies prv (imp \varphi (imp \chi \psi2)) \implies prv (imp \varphi (imp \chi (cnj \psi1 \psi2)))$
 $\langle proof \rangle$

lemma *prv_cnj_imp_monoR5*:
assumes $\varphi1 \in fmla$ **and** $\varphi2 \in fmla$ **and** $\chi1 \in fmla$ **and** $\chi2 \in fmla$

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 $\varphi 1 \in \text{fmla}$ **and** $\varphi 2 \in \text{fmla}$ **and** $\chi \in \text{fmla}$
shows *prv* (*imp* (*eqv* $\varphi 1$ $\varphi 2$) (*eqv* (*imp* $\varphi 1$ χ) (*imp* $\varphi 2$ χ)))
<proof>

lemma *imp_imp_compat_eqvR*:
assumes $\varphi \in \text{fmla}$ **and** $\chi 1 \in \text{fmla}$ **and** $\chi 2 \in \text{fmla}$
shows *prv* (*imp* (*eqv* $\chi 1$ $\chi 2$) (*eqv* (*imp* φ $\chi 1$) (*imp* φ $\chi 2$)))
<proof>

lemma *imp_imp_compat_eqv*:
assumes $\varphi 1 \in \text{fmla}$ **and** $\varphi 2 \in \text{fmla}$ **and** $\chi 1 \in \text{fmla}$ **and** $\chi 2 \in \text{fmla}$
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 $\varphi 1 \in \text{fmla}$ **and** $\varphi 2 \in \text{fmla}$ **and** $\chi \in \text{fmla}$
assumes *prv* (*eqv* $\varphi 1$ $\varphi 2$)
shows *prv* (*eqv* (*imp* $\varphi 1$ χ) (*imp* $\varphi 2$ χ))
<proof>

lemma *imp_compat_eqvR*:
assumes $\varphi \in \text{fmla}$ **and** $\chi 1 \in \text{fmla}$ **and** $\chi 2 \in \text{fmla}$
assumes *prv* (*eqv* $\chi 1$ $\chi 2$)
shows *prv* (*eqv* (*imp* φ $\chi 1$) (*imp* φ $\chi 2$))
<proof>

lemma *imp_compat_eqv*:
assumes $\varphi 1 \in \text{fmla}$ **and** $\varphi 2 \in \text{fmla}$ **and** $\chi 1 \in \text{fmla}$ **and** $\chi 2 \in \text{fmla}$
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 $\varphi 1 \in \text{fmla}$ **and** $\varphi 2 \in \text{fmla}$ **and** $\chi \in \text{fmla}$
shows *prv* (*imp* (*eqv* $\varphi 1$ $\varphi 2$) (*eqv* (*cnj* $\varphi 1$ χ) (*cnj* $\varphi 2$ χ)))
<proof>

lemma *imp_cnj_compat_eqvR*:
assumes $\varphi \in \text{fmla}$ **and** $\chi 1 \in \text{fmla}$ **and** $\chi 2 \in \text{fmla}$
shows *prv* (*imp* (*eqv* $\chi 1$ $\chi 2$) (*eqv* (*cnj* φ $\chi 1$) (*cnj* φ $\chi 2$)))
<proof>

lemma *imp_cnj_compat_eqv*:
assumes $\varphi 1 \in \text{fmla}$ **and** $\varphi 2 \in \text{fmla}$ **and** $\chi 1 \in \text{fmla}$ **and** $\chi 2 \in \text{fmla}$
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 $\varphi 1 \in \text{fmla}$ **and** $\varphi 2 \in \text{fmla}$ **and** $\chi \in \text{fmla}$
assumes *prv* (*eqv* $\varphi 1$ $\varphi 2$)
shows *prv* (*eqv* (*cnj* $\varphi 1$ χ) (*cnj* $\varphi 2$ χ))
<proof>

lemma *cnj_compat_eqvR*:

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

lemma *cnj_compat_equiv*:
assumes $\varphi1 \in \text{fmla}$ **and** $\varphi2 \in \text{fmla}$ **and** $\chi1 \in \text{fmla}$ **and** $\chi2 \in \text{fmla}$
assumes $\text{prv} (\text{equiv } \varphi1 \varphi2)$ **and** $\text{prv} (\text{equiv } \chi1 \chi2)$
shows $\text{prv} (\text{equiv} (\text{conj } \varphi1 \chi1) (\text{conj } \varphi2 \chi2))$
 <proof>

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

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

lemma *prv_imp_equiv_transi*:
assumes $\varphi \in \text{fmla}$ **and** $\chi1 \in \text{fmla}$ **and** $\chi2 \in \text{fmla}$
assumes $\text{prv} (\text{imp } \varphi \chi1)$ **and** $\text{prv} (\text{equiv } \chi1 \chi2)$
shows $\text{prv} (\text{imp } \varphi \chi2)$
 <proof>

lemma *prv_imp_equiv_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** $\text{prv} (\text{equiv } \chi1 \chi2)$
shows $\text{prv} (\text{imp } \varphi \chi1)$
 <proof>

lemma *prv_equiv_imp_transi*:
assumes $\varphi1 \in \text{fmla}$ **and** $\varphi2 \in \text{fmla}$ **and** $\chi \in \text{fmla}$
assumes $\text{prv} (\text{equiv } \varphi1 \varphi2)$ **and** $\text{prv} (\text{imp } \varphi2 \chi)$
shows $\text{prv} (\text{imp } \varphi1 \chi)$
 <proof>

lemma *prv_equiv_imp_transi_rev*:
assumes $\varphi1 \in \text{fmla}$ **and** $\varphi2 \in \text{fmla}$ **and** $\chi \in \text{fmla}$
assumes $\text{prv} (\text{equiv } \varphi1 \varphi2)$ **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 $\text{prv } \varphi \implies \text{prv } (\text{subst } \varphi t x)$
(*proof*)

lemma *prv_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 $\text{prv } \varphi$
shows $\text{prv } (\text{rawsubst } \varphi \text{ txs})$
(*proof*)

lemma *prv_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 $\text{prv } \varphi$
shows $\text{prv } (\text{psubst } \varphi \text{ txs})$
(*proof*)

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

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

lemma *prv_all_imp_trans*:
assumes $x \in \text{var}$ **and** $\varphi \in \text{fmla}$ **and** $\chi \in \text{fmla}$ **and** $\psi \in \text{fmla}$
shows $\text{prv } (\text{all } x (\text{imp } \varphi \chi)) \implies \text{prv } (\text{all } x (\text{imp } \chi \psi)) \implies \text{prv } (\text{all } x (\text{imp } \varphi \psi))$
(*proof*)

lemma *prv_all_imp_cnj*:
assumes $x \in \text{var}$ **and** $\varphi \in \text{fmla}$ **and** $\chi \in \text{fmla}$ **and** $\psi \in \text{fmla}$
shows $\text{prv } (\text{all } x (\text{imp } \varphi (\text{imp } \psi \chi))) \implies \text{prv } (\text{all } x (\text{imp } (\text{cnj } \psi \varphi) \chi))$
(*proof*)

lemma *prv_all_imp_cnj_rev*:
assumes $x \in \text{var}$ **and** $\varphi \in \text{fmla}$ **and** $\chi \in \text{fmla}$ **and** $\psi \in \text{fmla}$
shows $\text{prv } (\text{all } x (\text{imp } (\text{cnj } \varphi \psi) \chi)) \implies \text{prv } (\text{all } x (\text{imp } \varphi (\text{imp } \psi \chi)))$
(*proof*)

3.1.3 Properties concerning equality

lemma *prv_eql_reflT*:
assumes $t: t \in \text{trm}$
shows $\text{prv } (\text{eql } t t)$
(*proof*)

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

lemma *prv_eql_subst_trm2*:
assumes $x \in \text{var}$ **and** $\varphi \in \text{fmla}$ **and** $t1 \in \text{trm}$ **and** $t2 \in \text{trm}$
assumes $\text{prv } (\text{eql } t1 t2)$
shows $\text{prv } (\text{imp } (\text{subst } \varphi t1 x) (\text{subst } \varphi t2 x))$

<proof>

lemma *prv_eql_sym*:

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

lemma *prv_prv_eql_sym*: $t1 \in trm \implies t2 \in trm \implies prv$ (*eql* $t1$ $t2$) $\implies prv$ (*eql* $t2$ $t1$)
<proof>

lemma *prv_all_eql*:

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* (*eql* $t1$ $t2$)
(*imp* (*subst* φ $t2$ y) (*subst* φ $t1$ y))))))
<proof>

lemma *prv_eql_subst_trm_rev*:

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

lemma *prv_eql_subst_trm_rev2*:

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* φ $t2$ x) (*subst* φ $t1$ x))
<proof>

lemma *prv_eql_subst_trm_eqv*:

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

lemma *prv_eql_subst_trm_id*:

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

lemma *prv_eql_subst_trm_id_back*:

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

lemma *prv_eql_subst_trm_id_rev*:

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

lemma *prv_eql_subst_trm_id_back_rev*:

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

lemma *prv_eql_imp_trans_rev*:

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

<proof>

lemma *prv_eql_imp_trans*:

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

lemma *prv_eql_trans*:

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

3.1.4 The equivalence between soft substitution and substitution

lemma *prv_subst_imp_softSubst*:

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

lemma *prv_subst_implies_softSubst*:

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

lemma *prv_softSubst_imp_subst*:

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

lemma *prv_softSubst_implies_subst*:

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

lemma *prv_softSubst_eqv_subst*:

assumes [*simp,intro!*]: $x \in \text{var}$ $t \in \text{trm}$ $\varphi \in \text{fmla}$ $x \notin \text{FvarsT } t$
shows *prv* (*eqv* (*softSubst* φ *t* *x*) (*subst* φ *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
eql cnj imp all exi
fls
+
Deduct
var trm fmla Var FvarsT substT Fvars subst
num
eql 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 φ*

<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 φ* and *prv (neg φ)*

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 φ*

shows *prv (neg (neg φ))*

<proof>

lemma *prv_imp_imp_neg_rev*:

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

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

(imp (neg χ) (neg φ))

<proof>

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

lemma *prv_eqv_neg_prv_fls*:
 $\varphi \in \text{fmla} \implies$
prv (*eqv* φ (*neg* φ)) \implies *prv fls*
<proof>

lemma *prv_eqv_eqv_neg_prv_fls*:
 $\varphi \in \text{fmla} \implies \chi \in \text{fmla} \implies$
prv (*eqv* φ χ) \implies *prv* (*eqv* φ (*neg* χ)) \implies *prv fls*
<proof>

lemma *prv_eqv_eqv_neg_prv_fls2*:
 $\varphi \in \text{fmla} \implies \chi \in \text{fmla} \implies$
prv (*eqv* φ χ) \implies *prv* (*eqv* χ (*neg* φ)) \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* φ)
and *prv* (*imp* χ (*imp* ψ φ))
shows *prv* (*imp* χ (*neg* ψ))
<proof>

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

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

lemma *prv_imp_neg_imp_cnjL*:
assumes $\varphi \in \text{fmla}$ $\varphi1 \in \text{fmla}$ $\varphi2 \in \text{fmla}$
and *prv* (*imp* φ (*neg* $\varphi1$))
shows *prv* (*imp* φ (*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* φ (*neg* $\varphi2$))
shows *prv* (*imp* φ (*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* φ)) (*neg* (*exi* x φ)))
<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 $prv (eqv (scnj (insert \varphi F)) (cnj \varphi (scnj F)))$
 ⟨proof⟩

lemma *prv_scnj1_imp*:
 $\varphi \in fmla \implies prv (imp (scnj \{\varphi\}) \varphi)$
 ⟨proof⟩

lemma *prv_imp_scnj1*:
 $\varphi \in fmla \implies prv (imp \varphi (scnj \{\varphi\}))$
 ⟨proof⟩

lemma *prv_scnj2_imp_cnj*:
 $\varphi \in fmla \implies \psi \in fmla \implies prv (imp (scnj \{\varphi, \psi\}) (cnj \varphi \psi))$
 ⟨proof⟩

lemma *prv_cnj_imp_scnj2*:
 $\varphi \in fmla \implies \psi \in fmla \implies prv (imp (cnj \varphi \psi) (scnj \{\varphi, \psi\}))$
 ⟨proof⟩

lemma *prv_imp_imp_scnj2*:
 $\varphi \in fmla \implies \psi \in fmla \implies prv (imp \varphi (imp \psi (scnj \{\varphi, \psi\})))$
 ⟨proof⟩

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

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 $prv (eqv (psubst (scnj F) txs) (scnj ((\lambda \varphi. psubst \varphi txs) \text{ ' } F)))$
 ⟨proof⟩

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 $prv (imp (psubst (scnj F) txs) (scnj ((\lambda \varphi. psubst \varphi txs) \text{ ' } F)))$
 ⟨proof⟩

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 $prv (imp (scnj ((\lambda \varphi. psubst \varphi txs) \text{ ' } F)) (psubst (scnj F) txs))$
 ⟨proof⟩

3.2.5 Consistency and ω -consistency

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

lemma *consistent_def2*: $consistent \longleftrightarrow (\exists \varphi \in fmla. \neg prv \varphi)$
 ⟨proof⟩

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⟩

ω -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  $\varphi1 \in \text{fmla}$   $\varphi2 \in \text{fmla}$   $\chi \in \text{fmla}$ 
  and prv (dsj  $\varphi1$   $\varphi2$ ) and prv (imp  $\varphi1$   $\chi$ ) and prv (imp  $\varphi2$   $\chi$ )
  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 $prv (equiv (dsj \varphi \chi) (dsj \chi \varphi))$
 $\langle proof \rangle$

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 $prv (equiv (dsj \varphi (dsj \chi \psi)) (dsj (dsj \varphi \chi) \psi))$
 $\langle proof \rangle$

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 $prv (equiv (cnj \varphi (dsj \chi1 \chi2)) (dsj (cnj \varphi \chi1) (cnj \varphi \chi2)))$
 $\langle proof \rangle$

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_ldsj_in*:

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

$\langle proof \rangle$

lemma *prv_imp_ldsj*:

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

$\langle proof \rangle$

lemma *prv_ldsj_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 $prv (imp (ldsj \varphi s) \chi)$

$\langle proof \rangle$

lemma *prv_ldsj_imp_inner*:

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

$\langle proof \rangle$

lemma *prv_ldsj_imp_remdups*:

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

$\langle proof \rangle$

lemma *prv_ldsj_mono*:

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

$\langle proof \rangle$

lemma *prv_ldsj_eqv*:
assumes $set\ \varphi 1s \subseteq fmla$ **and** $set\ \varphi 2s = set\ \varphi 1s$
shows $prv\ (eqv\ (ldsj\ \varphi 1s)\ (ldsj\ \varphi 2s))$
 $\langle proof \rangle$

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 $prv\ (imp\ (ldsj\ \varphi 1s)\ (ldsj\ \varphi 2s))$
 $\langle proof \rangle$

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

lemma *prv_subst_sdsj*:
 $F \subseteq fmla \implies finite\ F \implies t \in trm \implies x \in var \implies$
 $prv\ (eqv\ (subst\ (sdsj\ F)\ t\ x)\ (sdsj\ ((\lambda\varphi. subst\ \varphi\ t\ x)\ 'F)))$
 $\langle proof \rangle$

lemma *prv_imp_sdsj_in*:
assumes $\varphi \in fmla$ **and** $F \subseteq fmla$ *finite F*
and $\varphi \in F$
shows $prv\ (imp\ \varphi\ (sdsj\ F))$
 $\langle proof \rangle$

lemma *prv_imp_sdsj*:
assumes $\chi \in fmla$ **and** $F \subseteq fmla$ *finite F*
and $\varphi \in F$ **and** $prv\ (imp\ \chi\ \varphi)$
shows $prv\ (imp\ \chi\ (sdsj\ F))$
 $\langle proof \rangle$

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 $prv\ (imp\ (sdsj\ F)\ \chi)$
 $\langle proof \rangle$

lemma *prv_sdsj_mono*:
assumes $F2 \subseteq fmla$ **and** $F1 \subseteq F2$ **and** *finite F2*
shows $prv\ (imp\ (sdsj\ F1)\ (sdsj\ F2))$
 $\langle proof \rangle$

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 $prv\ (imp\ (sdsj\ F1)\ (sdsj\ F2))$
 $\langle proof \rangle$

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 $prv\ \psi$
 $\langle proof \rangle$

lemma *prv_sdsj1_imp*:
 $\varphi \in fmla \implies prv\ (imp\ (sdsj\ \{\varphi\})\ \varphi)$
 $\langle proof \rangle$

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 $\text{snd } \text{'(set } \text{txs)} \subseteq \text{var fst } \text{'(set } \text{txs)} \subseteq \text{trm}$
shows $\text{prv} (\text{eqv} (\text{rawpsubst} (\text{sdsj } F) \text{txs}) (\text{sdsj } ((\lambda\varphi. \text{rawpsubst } \varphi \text{txs}) \text{' } F)))$
 <proof>

lemma *prv_psubst_sdsj*:
assumes $F \subseteq \text{fmla}$ *finite* F
and $\text{snd } \text{'(set } \text{txs)} \subseteq \text{var fst } \text{'(set } \text{txs)} \subseteq \text{trm}$
and *distinct* $(\text{map } \text{snd } \text{txs})$
shows $\text{prv} (\text{eqv} (\text{psubst} (\text{sdsj } F) \text{txs}) (\text{sdsj } ((\lambda\varphi. \text{psubst } \varphi \text{txs}) \text{' } F)))$
 <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
 $\text{var } \text{trm } \text{fmla } \text{Var } \text{FvarsT } \text{substT } \text{Fvars } \text{subst}$
 $\text{eql } \text{cnj } \text{imp } \text{all } \text{exi}$
 fls
 dsj
 num
 prv
 +
Syntax_with_Connectives_Rename
 $\text{var } \text{trm } \text{fmla } \text{Var } \text{FvarsT } \text{substT } \text{Fvars } \text{subst}$
 $\text{eql } \text{cnj } \text{imp } \text{all } \text{exi}$
for
 $\text{var} :: \text{'var set } \text{and } \text{trm} :: \text{'trm set } \text{and } \text{fmla} :: \text{'fmla set}$
and $\text{Var } \text{FvarsT } \text{substT } \text{Fvars } \text{subst}$
and $\text{eql } \text{cnj } \text{imp } \text{all } \text{exi}$
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  $\varphi1 \Longrightarrow \text{nprv} (\text{insert } \varphi1 \text{ } F) \ \varphi2 \Longrightarrow \text{nprv} (\text{insert } \varphi2 (\text{insert } \varphi1 \text{ } F)) \ \psi \Longrightarrow$ 
 $F \subseteq \text{fmla} \Longrightarrow \text{finite } F \Longrightarrow \varphi1 \in \text{fmla} \Longrightarrow \varphi2 \in \text{fmla} \Longrightarrow \psi \in \text{fmla} \Longrightarrow$ 
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*: $nprv\ \{\varphi\}\ \varphi \implies \varphi \in fmla \implies prv\ \varphi$
 <proof>

4.4 More Structural Properties

lemma *nprv_clear*: $nprv\ \{\varphi\}\ \varphi \implies F \subseteq fmla \implies finite\ F \implies \varphi \in fmla \implies nprv\ F\ \varphi$
 <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 $\text{nprv } ((\lambda \varphi. \text{subst } \varphi \ t \ x) \ ' F) (\text{subst } \psi \ t \ x)$
 $\langle \text{proof} \rangle$

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 $\text{nprv } F (\text{subst } \psi \ t \ x)$
 $\langle \text{proof} \rangle$

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 $1: \text{nprv } ((\lambda \varphi. \text{subst } \varphi \ (Var \ y) \ x) \ ' F) (\text{subst } \psi \ (Var \ y) \ x)$
shows $\text{nprv } F \ \psi$
 $\langle \text{proof} \rangle$

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 $\text{nprv } ((\lambda \varphi. \text{psubst } \varphi \ \text{txs}) \ ' F) (\text{psubst } \psi \ \text{txs})$
 $\langle \text{proof} \rangle$

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

lemma *nprv_impI_rev2*:
assumes $\text{nprv } F (\text{imp } \varphi \ \psi)$ **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 $\text{nprv } G \ \psi$
 $\langle \text{proof} \rangle$

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 φ 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 φ)
<proof>

lemma *nprv_exiE*:
assumes *n: nprv F (exi x φ)*
and *nn: nprv (insert φ F) ψ*
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 ψ*
shows *nprv F ψ*
<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 φ nprv (insert φ F) ψ*
and *$\varphi \in$ fmla $\psi \in$ fmla and F \subseteq fmla and finite F*
shows *nprv F ψ*
<proof>

lemmas *nprv_addLemmaE1 = nprv_addLemmaE[OF _ nprv_hyp, simped]*

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

lemma *nprv_addImpLemmaE*:
assumes *prv (imp φ 1 φ 2) and nprv F φ 1 and nprv ((insert φ 2) F) ψ*
and *F \subseteq fmla finite F φ 1 \in fmla φ 2 \in fmla $\psi \in$ fmla*
shows *nprv F ψ*
<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 φ 1 (imp φ 2 φ 3))*
and *F \subseteq fmla finite F φ 1 \in fmla φ 2 \in fmla φ 3 \in fmla*
and *nprv F φ 1 nprv F φ 2*
shows *nprv F φ 3*
<proof>

lemma *nprv_addImp2LemmaE*:
assumes *prv (imp φ 1 (imp φ 2 φ 3)) and nprv F φ 1 and nprv F φ 2 and nprv ((insert φ 3) F) ψ*
and *F \subseteq fmla finite F φ 1 \in fmla φ 2 \in fmla φ 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*: $F \subseteq \text{fmla} \implies \text{finite } F \implies t \in \text{trm} \implies \text{nprv } F \text{ (eq } t \text{)}$
<proof>

lemma *nprv_eq_eqLI*: $t1 = t2 \implies F \subseteq \text{fmla} \implies \text{finite } F \implies t1 \in \text{trm} \implies \text{nprv } F \text{ (eq } t1 \text{)}$
<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) ψ
*<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 φ)} (exi y (exi x φ))*
<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** $n2: nprv\ (insert\ \varphi\ F)\ (eql\ (Var\ x)\ t)$
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** $n2: nprv\ (insert\ (subst\ \varphi\ (Var\ y)\ x)\ F)\ (eql\ (Var\ y)\ t)$
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** $n2: nprv\ (insert\ (subst\ \varphi\ (Var\ y)\ x)\ (insert\ \varphi\ F))\ (eql\ (Var\ y)\ (Var\ x))$
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 $prv (\text{imp} (\text{exu } x \ \varphi) (\text{imp} (\text{subst } \varphi \ t1 \ x) (\text{imp} (\text{subst } \varphi \ t2 \ x) (\text{eql } t1 \ t2))))$

<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** $r1 \ r2 =$

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

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

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

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

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

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

$(-, ((nrule \ r: \ r1)?) , (nrule \ r: \ r2)?) , (nrule \ r: \ r3)?) ,$
 $(nrule \ r: \ r4)?) , (nrule \ r: \ r5)?) ; \text{fail}))$

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

$(-, ((nrule \ r: \ r1)?) , (nrule \ r: \ r2)?) , (nrule \ r: \ r3)?) ,$
 $(nrule \ r: \ r4)?) , (nrule \ r: \ r5)?) , (nrule \ r: \ r6)?) ; \text{fail}))$

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 σ satisfies a property φ : (1) using \exists : there exists an "out" such that σ and φ hold for it; (2) using \forall : for all "out" such that σ holds for it, φ 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 φ 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 φ with a pseudo-term τ yields a formula that could be casually written $\varphi(\tau)$. It states the existence of an output zz of τ on which φ holds. Instead of $\varphi(\tau)$, we write $instInpP\ \varphi\ n\ \tau$ where n is the number of input variables of τ . In the name *instInpP*, *Inp* is as before a reminder that we instantiate φ on its variable "inp" and the suffix "P" stands for "Pseudo".

definition $instInpP :: 'fmla \Rightarrow nat \Rightarrow 'fmla \Rightarrow 'fmla$ **where**

$instInpP\ \varphi\ n\ \tau \equiv let\ zz = Variable\ (Suc\ (Suc\ n))\ in$

$exi\ zz\ (cnj\ (subst\ \tau\ (Var\ zz)\ out)\ (subst\ \varphi\ (Var\ zz)\ inp))$

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* χ (*Suc* 0) σ) 0 τ) (is ?B)
 <proof>

lemma *prv_instInpP_compose*:

assumes [*simp*]: $\chi \in \text{fmla}$ *Fvars* $\chi = \{\text{inp}\}$
and $\sigma[\text{simp}]: \sigma \in \text{ptrm}$ (*Suc* 0) **and** $\tau[\text{simp}]: \tau \in \text{ptrm}$ 0
shows *prv* (*imp* (*instInpP* (*instInpP* χ (*Suc* 0) σ) 0 τ)
 (*instInpP* χ 0 (*instInpP* σ 0 τ))) (is ?A)

and

prv (*imp* (*instInpP* χ 0 (*instInpP* σ 0 τ))
 (*instInpP* (*instInpP* χ (*Suc* 0) σ) 0 τ)) (is ?B)

and

prv (*eqv* (*instInpP* (*instInpP* χ (*Suc* 0) σ) 0 τ)
 (*instInpP* χ 0 (*instInpP* σ 0 τ))) (is ?C)

<proof>

5.4 Equality between Pseudo-Terms and Terms

Casually, the equality between a pseudo-term τ and a term t can be written as $\vdash \tau = t$. This is in fact the (provability of) the instantiation of τ with t on τ 's output variable out. Indeed, this formula says that the unique entity denoted by τ is exactly t .

definition *prveqlPT* :: 'fmla \Rightarrow 'trm \Rightarrow bool **where**
prveqlPT τ $t \equiv \text{prv}$ (*subst* τ 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*]: $\varphi \in \text{fmla}$ **and** *f*: *Fvars* $\varphi \subseteq \{\text{inp}\}$ **and** $\tau: \tau \in \text{ptrm}$ 0
and [*simp*]: $t \in \text{trm}$ *FvarsT* $t = \{\}$
and τt : *prveqlPT* τ t
shows *nprv* {*instInpP* φ 0 τ } (*instInpP* φ t)
 <proof>

lemma *prveqlPT_prv_instInp_instInpP*:

assumes $\varphi \in \text{fmla}$ **and** *f*: *Fvars* $\varphi \subseteq \{\text{inp}\}$ **and** $\tau: \tau \in \text{ptrm}$ 0
and $t \in \text{trm}$ *FvarsT* $t = \{\}$
and τt : *prveqlPT* τ t
shows *prv* (*imp* (*instInpP* φ 0 τ) (*instInpP* φ t))
 <proof>

lemma *prveqlPT_nprv_instInpP_instInp*:

assumes[*simp*]: $\varphi \in \text{fmla}$ **and** *f*: *Fvars* $\varphi \subseteq \{\text{inp}\}$ **and** $\tau: \tau \in \text{ptrm}$ 0
and [*simp*]: $t \in \text{trm}$ *FvarsT* $t = \{\}$
and τt : *prveqlPT* τ t
shows *nprv* {*instInpP* φ t } (*instInpP* φ 0 τ)
 <proof>

lemma *prveqlPT_prv_instInpP_instInp*:

assumes $\varphi \in \text{fmla}$ **and** *f*: *Fvars* $\varphi \subseteq \{\text{inp}\}$ **and** $\tau: \tau \in \text{ptrm}$ 0
and $t \in \text{trm}$ *FvarsT* $t = \{\}$
and τt : *prveqlPT* τ t
shows *prv* (*imp* (*instInpP* φ t) (*instInpP* φ 0 τ))
 <proof>

lemma *prveqlPT_prv_instInp_eqv_instInpP*:

assumes $\varphi \in \text{fmla}$ **and** *f*: *Fvars* $\varphi \subseteq \{\text{inp}\}$ **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  $\varphi$   $t$ ))  
<proof>
```

```
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
    egl 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 egl cnj imp all exi
and fls
and dsj
and num
+
— The notion of truth for sentences:
fixes isTrue :: 'fmla ⇒ bool
assumes
  not_isTrue_fls: ¬ isTrue fls
and
isTrue_imp:
 $\bigwedge \varphi \psi. \varphi \in \text{fmla} \implies \psi \in \text{fmla} \implies \text{Fvars } \varphi = \{\} \implies \text{Fvars } \psi = \{\} \implies$ 
 $\text{isTrue } \varphi \implies \text{isTrue } (\text{imp } \varphi \psi) \implies \text{isTrue } \psi$ 
and
isTrue_all:
 $\bigwedge x \varphi. x \in \text{var} \implies \varphi \in \text{fmla} \implies \text{Fvars } \varphi = \{x\} \implies$ 
 $(\forall n \in \text{num}. \text{isTrue } (\text{subst } \varphi n x)) \implies \text{isTrue } (\text{all } x \varphi)$ 
and
isTrue_exi:
 $\bigwedge x \varphi. x \in \text{var} \implies \varphi \in \text{fmla} \implies \text{Fvars } \varphi = \{x\} \implies$ 
 $\text{isTrue } (\text{exi } x \varphi) \implies (\exists n \in \text{num}. \text{isTrue } (\text{subst } \varphi n x))$ 
and
isTrue_neg:
 $\bigwedge \varphi. \varphi \in \text{fmla} \implies \text{Fvars } \varphi = \{\} \implies$ 
 $\text{isTrue } \varphi \vee \text{isTrue } (\text{neg } \varphi)$ 
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 ω -consistent (in the strong, intuitionistic sense):

lemma ω consistent: ω consistent
<proof>

lemma ω consistentStd1: ω consistentStd1
<proof>

lemma ω consistentStd2: ω 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]:  $\bigwedge t x. t \in \text{trm} \implies x \in \text{var} \implies$   
  substT zer t x = zer  
and  
suc[simp]:  $\bigwedge t. t \in \text{trm} \implies \text{suc } t \in \text{trm}$   
and  
FvarsT_suc[simp]:  $\bigwedge t. t \in \text{trm} \implies$   
  FvarsT (suc t) = FvarsT t  
and  
substT_suc[simp]:  $\bigwedge t1 t x. t1 \in \text{trm} \implies t \in \text{trm} \implies x \in \text{var} \implies$ 
```

```

  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 $\text{us_facts: set us } \subseteq \text{var}$
 $\text{set us } \cap \text{FvarsT } r = \{\}$
 $\text{set us } \cap \bigcup (\text{FvarsT} \text{ ' (fst ' (set txs))}) = \{\}$
 $\text{set us } \cap \text{snd} \text{ ' (set txs) } = \{\}$
 $\text{length us} = \text{length txs}$
 distinct us
shows $\text{rawpsubstT (rawpsubstT } r \text{ (zip (map Var us) (map snd txs))) (zip (map fst txs) us) = rawpsubstT } r \text{ txs}$
<proof>

lemma *rawpsubstT_compose_freshVar2_aux*:
assumes $r[\text{simp}] : r \in \text{atrm}$
and $\text{ts: set ts } \subseteq \text{atrm}$
and $\text{xs: set xs } \subseteq \text{var}$ distinct xs
and $\text{us_facts: set us } \subseteq \text{var}$ 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 $\text{vs_facts: set vs } \subseteq \text{var}$ 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 $\text{rawpsubstT (rawpsubstT } r \text{ (zip (map Var us) xs)) (zip ts us) = rawpsubstT (rawpsubstT } r \text{ (zip (map Var vs) xs)) (zip ts vs)}$
<proof>

lemma *rawpsubstT_compose_freshVar2*:
assumes $r[\text{simp}] : r \in \text{atrm}$
and $\text{ts: set ts } \subseteq \text{atrm}$
and $\text{xs: set xs } \subseteq \text{var}$ distinct xs
and $\text{us_facts: set us } \subseteq \text{var}$ 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 $\text{vs_facts: set vs } \subseteq \text{var}$ 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 $\text{rawpsubstT (rawpsubstT } r \text{ (zip (map Var us) xs)) (zip ts us) = rawpsubstT (rawpsubstT } r \text{ (zip (map Var vs) xs)) (zip ts vs) (is ?L = ?R)}$
<proof>

lemma *in_fst_image*: $a \in \text{fst} \text{ ' } AB \iff (\exists b. (a,b) \in AB)$ *<proof>*

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 $\bigwedge i j. i < j \implies j < \text{length } txs \implies \text{snd } (txs!j) \notin \text{FvarsT } (\text{fst } (txs!i))$
shows $\text{psubstT } r \ txs = \text{rawpsubstT } r \ txs$
<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 $r \in \text{atrm } \text{set } xs \subseteq \text{var}$
shows $\text{rawpsubstT } r \ (\text{map } (\lambda x. (\text{Var } x,x)) \ xs) = r$
<proof>

lemma *psubstT_same_Var[simp]*:
assumes $r \in \text{atrm } \text{set } xs \subseteq \text{var}$ **and** *distinct* xs
shows $\text{psubstT } r \ (\text{map } (\lambda x. (\text{Var } x,x)) \ xs) = r$
<proof>

thm *psubstT_notIn*

lemma *rawpsubst_eq1*:
assumes $t1 \in \text{trm } t2 \in \text{trm}$
and $\text{snd } '(\text{set } txs) \subseteq \text{var } \text{fst } '(\text{set } txs) \subseteq \text{trm}$
shows $\text{rawpsubst } (\text{eql } t1 \ t2) \ txs = \text{eql } (\text{rawpsubstT } t1 \ txs) \ (\text{rawpsubstT } t2 \ txs)$
<proof>

lemma *psubst_eq1[simp]*:
assumes $t1 \in \text{atrm } t2 \in \text{atrm}$
and $\text{snd } '(\text{set } txs) \subseteq \text{var } \text{fst } '(\text{set } txs) \subseteq \text{atrm}$
and *distinct* $(\text{map } \text{snd } \ txs)$

shows $psubst (eql t1 t2) txs = eql (psubstT t1 txs) (psubstT t2 txs)$
 ⟨proof⟩

lemma *psubst_exu[simp]*:
assumes $\varphi \in fmla$ $x \in var$ $snd \text{ ' } (set\ txs \subseteq var\ fst \text{ ' } (set\ txs \subseteq atrm$
 $x \notin snd \text{ ' } (set\ txs$ $x \notin (\bigcup t \in fst \text{ ' } (set\ txs. FvarsT\ t))\ distinct\ (map\ snd\ txs)$
shows $psubst (exu\ x\ \varphi)\ txs = exu\ x\ (psubst\ \varphi\ txs)$
 ⟨proof⟩

thm *psubstT_Var_not[no_vars]*

lemma *rawpsubstT_Var_in*:
assumes $snd \text{ ' } (set\ txs) \subseteq var\ fst \text{ ' } (set\ txs) \subseteq trm$
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 $y \in var$ $snd \text{ ' } (set\ txs) \subseteq var\ fst \text{ ' } (set\ txs) \subseteq trm$
and $distinct\ (map\ snd\ txs)$ **and** $(s,y) \in set\ txs$
shows $psubstT\ (Var\ y)\ txs = s$
 ⟨proof⟩

lemma *psubstT_Var_Cons_aux*:
assumes $y \in var$ $x \in var$ $t \in atrm$
 $snd \text{ ' } (set\ txs) \subseteq var\ fst \text{ ' } (set\ txs) \subseteq atrm$ $x \notin snd \text{ ' } (set\ txs)$
 $distinct\ (map\ snd\ txs)$ $y \neq x$
shows $psubstT\ (Var\ y)\ ((t,x) \# txs) = psubstT\ (Var\ y)\ txs$
 ⟨proof⟩

Simplification rules for parallel substitution:

lemma *psubstT_Var_Cons[simp]*:
 $y \in var \implies x \in var \implies t \in atrm \implies$
 $snd \text{ ' } (set\ txs) \subseteq var \implies fst \text{ ' } (set\ txs) \subseteq atrm \implies distinct\ (map\ snd\ txs) \implies x \notin snd \text{ ' } (set\ txs) \implies$
 $psubstT\ (Var\ y)\ ((t,x) \# txs) = (if\ y = x\ then\ t\ else\ psubstT\ (Var\ y)\ txs)$
 ⟨proof⟩

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

lemma *rawpsubstT_suc*:
assumes $r \in trm$ **and** $snd \text{ ' } (set\ txs) \subseteq var$ **and** $fst \text{ ' } (set\ txs) \subseteq trm$
shows $rawpsubstT\ (suc\ r)\ txs = suc\ (rawpsubstT\ r\ txs)$
 ⟨proof⟩

lemma *psubstT_suc[simp]*:
assumes $r \in atrm$ **and** $snd \text{ ' } (set\ txs) \subseteq var$ **and** $fst \text{ ' } (set\ txs) \subseteq atrm$
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 $Lq :: 'fmla$ **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 **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 $LLq \ t1 \ t2 = exi \ z \ (eql \ t2 \ (pls \ (Var \ z) \ t1))$

<proof>

lemma *LLq_pls_zz*:

assumes $t1 \in atrm \ t2 \in atrm \ zz \notin FvarsT \ t1 \ zz \notin FvarsT \ t2$

shows $LLq \ t1 \ t2 = exi \ zz \ (eql \ t2 \ (pls \ (Var \ zz) \ t1))$

<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**

$Ls \equiv cnj \ Lq \ (neg \ (eql \ (Var \ xx) \ (Var \ yy)))$

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*:

$t1 \in atrm \implies t2 \in atrm \implies$

$LLs \ t1 \ t2 = cnj \ (LLq \ t1 \ t2) \ (neg \ (eql \ t1 \ t2))$

<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 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 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 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 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 φ (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 φ) **and** $yx[simp]: y = x \vee y \notin$ Fvars φ*

shows *nprv F (all x φ)*

<proof>

lemma *nprv_exiE_var:*

assumes *n: nprv F (exi x φ)*

and *nn: nprv (insert (subst φ (Var y) x) F) ψ*

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 φ y \notin (\bigcup (Fvars ' F) y \notin Fvars ψ*

shows *nprv F ψ*

<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* =
 $\text{nprv_flsE}[\text{OF } \text{nprv_addImpLemmaE}[\text{OF } \text{prv_neg_zer_suc}[\text{unfolded } \text{neg_def}], \text{OF } _ _ \text{nprv_hyp}, \text{simped}, \text{rotated}]$

lemmas *nprv_zer_suc_contrE0* = $\text{nprv_zer_suc_contrE}[\text{OF } \text{nprv_hyp}, \text{simped}]$

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* = $\text{nprv_zer_suc_2contrE}[\text{OF } \text{nprv_hyp } _, \text{simped}]$

lemmas *nprv_zer_suc_2contrE1* = $\text{nprv_zer_suc_2contrE}[\text{OF } _ \text{nprv_hyp}, \text{simped}]$

lemmas *nprv_zer_suc_2contrE01* = $\text{nprv_zer_suc_2contrE}[\text{OF } \text{nprv_hyp } \text{nprv_hyp}, \text{simped}]$

lemma *prv_inj_suc*:
 $t \in \text{atrm} \implies t' \in \text{atrm} \implies$
 $\text{prv} (\text{imp} (\text{eql} (\text{suc } t) (\text{suc } t'))$
 $(\text{eql } t t'))$
 ⟨*proof*⟩

lemmas *nprv_eql_sucI* = $\text{nprv_addImpLemmaI}[\text{OF } \text{prv_inj_suc}, \text{simped}, \text{rotated } 4]$

lemmas *nprv_eql_sucE* = $\text{nprv_addImpLemmaE}[\text{OF } \text{prv_inj_suc}, \text{simped}, \text{rotated } 2]$

lemmas *nprv_eql_sucE0* = $\text{nprv_eql_sucE}[\text{OF } \text{nprv_hyp } _, \text{simped}]$

lemmas *nprv_eql_sucE1* = $\text{nprv_eql_sucE}[\text{OF } _ \text{nprv_hyp}, \text{simped}]$

lemmas *nprv_eql_sucE01* = $\text{nprv_eql_sucE}[\text{OF } \text{nprv_hyp } \text{nprv_hyp}, \text{simped}]$

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** $prv\ (eq\ (pls\ t\ zer)\ t)$
 $\langle proof \rangle$

lemma prv_pls_suc :
 $t \in atrm \implies t' \in atrm \implies$
 $prv\ (eq\ (pls\ t\ (suc\ t'))\ (suc\ (pls\ t\ t')))$
 $\langle proof \rangle$

lemma prv_tms_zer :
assumes $[simp]: t \in atrm$ **shows** $prv\ (eq\ (tms\ t\ zer)\ zer)$
 $\langle proof \rangle$

lemma prv_tms_suc :
 $t \in atrm \implies t' \in atrm \implies$
 $prv\ (eq\ (tms\ t\ (suc\ t'))\ (pls\ (tms\ t\ t')\ t))$
 $\langle proof \rangle$

lemma $prv_suc_imp_cong$:
assumes $t1[simp]: t1 \in atrm$ **and** $t2[simp]: t2 \in atrm$
shows $prv\ (imp\ (eq\ t1\ t2)\ (eq\ (suc\ t1)\ (suc\ t2)))$
 $\langle proof \rangle$

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 $prv\ (eq\ t1\ t2)$
shows $prv\ (eq\ (suc\ t1)\ (suc\ t2))$
 $\langle proof \rangle$

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 $prv\ (imp\ (eq\ t1\ t1')\ (imp\ (eq\ t2\ t2')\ (eq\ (pls\ t1\ t2)\ (pls\ t1'\ t2'))))$
 $\langle proof \rangle$

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 $[simp]: t \in atrm$

shows $prv\ (LLq\ zer\ t)$

<proof>

lemmas $Q1 = prv_LLq_zer$

lemma $prv_LLq_zer_imp_eql:$

assumes $[simp]: t \in atrm$

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 $[simp]: t \in atrm$

shows $prv\ (imp\ (lds\j\ (map\ (\lambda i. eql\ t\ (Num\ i))\ (toN\ n)))\ (LLq\ t\ (Num\ n)))$

<proof>

declare *subset_eq*[simp]
lemmas *nprv_sdsj_eq_LLqI* = *nprv_addImpLemmaI*[OF *prv_sdsj_eq_imp_LLq*, *simped*, *rotated 3*]
lemmas *nprv_sdsj_eq_LLqE* = *nprv_addImpLemmaE*[OF *prv_sdsj_eq_imp_LLq*, *simped*, *rotated 1*]
declare *subset_eq*[simp del]

lemmas *nprv_sdsj_eq_LLqE0* = *nprv_sdsj_eq_LLqE*[OF *nprv_hyp _*, *simped*]
lemmas *nprv_sdsj_eq_LLqE1* = *nprv_sdsj_eq_LLqE*[OF *_ nprv_hyp*, *simped*]
lemmas *nprv_sdsj_eq_LLqE01* = *nprv_sdsj_eq_LLqE*[OF *nprv_hyp nprv_hyp*, *simped*]

lemmas *O2I* = *nprv_sdsj_eq_LLqI*
lemmas *O2E* = *nprv_sdsj_eq_LLqE*
lemmas *O2E0* = *nprv_sdsj_eq_LLqE0*
lemmas *O2E1* = *nprv_sdsj_eq_LLqE1*
lemmas *O2E01* = *nprv_sdsj_eq_LLqE01*

lemma *prv_LLq_imp_sdsj_eqI*: — O3
assumes [simp]: $t \in \text{atrm}$
shows *prv (imp (LLq t (Num n)) (lds (map ($\lambda i. \text{eq } t \text{ (Num } i\text{)} (toN n))))$)*
<proof>

declare *subset_eq*[simp]
lemmas *prv_LLq_sdsj_eqII* = *nprv_addImpLemmaI*[OF *prv_LLq_imp_sdsj_eqI*, *simped*, *rotated 3*]
lemmas *prv_LLq_sdsj_eqIE* = *nprv_addImpLemmaE*[OF *prv_LLq_imp_sdsj_eqI*, *simped*, *rotated 1*]
declare *subset_eq*[simp del]

lemmas *prv_LLq_sdsj_eqIE0* = *prv_LLq_sdsj_eqIE*[OF *nprv_hyp _*, *simped*]
lemmas *prv_LLq_sdsj_eqIE1* = *prv_LLq_sdsj_eqIE*[OF *_ nprv_hyp*, *simped*]
lemmas *prv_LLq_sdsj_eqIE01* = *prv_LLq_sdsj_eqIE*[OF *nprv_hyp nprv_hyp*, *simped*]

lemmas *O3I* = *prv_LLq_sdsj_eqII*
lemmas *O3E* = *prv_LLq_sdsj_eqIE*
lemmas *O3E0* = *prv_LLq_sdsj_eqIE0*
lemmas *O3E1* = *prv_LLq_sdsj_eqIE1*
lemmas *O3E01* = *prv_LLq_sdsj_eqIE01*

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]: $x \in \text{var } \varphi \in \text{fmIa}$
and [simp]: $\bigwedge i. i \leq n \implies \text{prv (subst } \varphi \text{ (Num } i\text{)} x)$
shows *prv (ball x (Num n) φ)*
<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 $\text{prv_Num_LLq_eql_sucE}$:
 $\text{nprv} \ F \ (LLq \ (Num \ n) \ t) \ \Longrightarrow$
 $\text{nprv} \ (insert \ (eql \ (Num \ n) \ t) \ F) \ \psi \ \Longrightarrow$
 $\text{nprv} \ (insert \ (LLq \ (suc \ (Num \ n)) \ t) \ F) \ \psi \ \Longrightarrow$
 $t \in atrm \ \Longrightarrow \ F \subseteq fmla \ \Longrightarrow \ finite \ F \ \Longrightarrow \ \psi \in fmla \ \Longrightarrow$
 $\text{nprv} \ F \ \psi$
 $\langle proof \rangle$

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.