User's manual of the PhoX library

Version 0.89

Christophe Raffalli

Contents

Chapter 1

Basic PhoX Library

Warning: This library is always loaded !

1.1 Propositionnal connective.

1.1.1 Conjunction.

Definition 1.1 Conjunction.

$$X \wedge Y := \forall K\,((X \to Y \to K) \to K)$$ `X ∧ Y`

Proposition 1.2 Conjunction rules.

- conjunction.intro*:*

$$\forall X, Y\,(X \to Y \to X \wedge Y)$$

conjunction.intro *added as introduction rule (abbrev:* `n` *, options:)*

- conjunction.left.elim*:*

$$\forall X, Y\,(X \wedge Y \to X)$$

- conjunction.right.elim*:*

$$\forall X, Y\,(X \wedge Y \to Y)$$

- conjunction.left*:*

$$\forall X, Y, Z\,((Y \to Z \to X) \to Y \wedge Z \to X)$$

[0]written by: Christophe Raffalli, Paul Rozière (Université de Savoie, université Paris VII)

conjunction.left added as elimination rule (abbrev: `s` , options: `-n -i`)
conjunction.left.elim added as elimination rule (abbrev: `l` , options:)
conjunction.right.elim added as elimination rule (abbrev: `r` , options:)

Definition 1.3 Equivalence.

`X ↔ Y`

$$X \leftrightarrow Y := (X \to Y) \wedge (Y \to X)$$

1.1.2 Disjunction.

Definition 1.4 Disjunction.

`X ∨ Y`

$$X \vee Y := \forall K\,((X \to K) \to (Y \to K) \to K)$$

Proposition 1.5 Disjunction rules.

- disjunction.left.intro*:*

$$\forall X, Y\,(X \to X \vee Y)$$

- disjunction.right.intro*:*

$$\forall X, Y\,(Y \to X \vee Y)$$

  disjunction.left.intro *added as introduction rule (abbrev:* `l` *, options:)*

  disjunction.right.intro *added as introduction rule (abbrev:* `r` *, options:)*

- disjunction.elim*:*

$$\forall X, Y, Z\,((Y \to X) \to (Z \to X) \to (Y \vee Z) \to X)$$

  disjunction.elim *added as elimination rule (abbrev:* `n` *, options:* `-i` *)*

1.1.3 Propositional constants and negation.

Definition 1.6 Propositional constants and negation.

`False`
- $(\bot) := \forall X\, X$

`True`
- $(\top) := \forall X\,(X \to X)$

`¬ X`
- $(\neg)\, X := X \to \bot$

Proposition 1.7 Propositional constants and negation rules.

- true.intro*:*

$$\top$$

true.intro *added as introduction rule (abbrev:* `n` *, options:* *)*

- true.elim*:*

$$\forall X\,(X \to (\top) \to X)$$

true.elim *added as elimination rule (abbrev:* `l` *, options:* `-i -n` *)*

- false.elim*:*

$$\forall X\,((\bot) \to X)$$

false.elim *added as elimination rule (abbrev:* `n` *, options:* `-i` *)*

- not.elim*:*

$$\forall X, Y\,(X \to (\neg)\, X \to Y)$$

1.1.4 Existential quantifiers.

Definition 1.8 Existential quantifiers definitions.

- $\exists x\; A\; x$ `:=` $\forall K\,(\forall x{:}A\; K \to K)$ `∃x A x`
- $\exists ! x\; A\; x$ `:=` $\exists z\,\forall w\,(A\; w \leftrightarrow w = z)$ `∃!x A x`

Proposition 1.9 Existential rules

- exists.intro*:*

$$\forall A\; \forall x{:}A\; \exists x\; A\; x$$

exists.intro *added as introduction rule (abbrev:* `n` *, options:* *)*

- exists.elim*:*

$$\forall X\,\forall A\,(\forall x{:}A\; X \to \exists x\; A\; x \to X)$$

exists.elim *added as elimination rule (abbrev:* `l` *, options:* `-i` *)*

equal.reflexive *added as introduction rule (abbrev:* `refl` *, options:* `-i` *)*

- exists.one.intro*:*

$$\forall A\; \forall x{:}A\; (\forall y{:}A\; y = x \to \exists ! x\; A\; x)$$

exists.one.intro *added as introduction rule (abbrev:* `n` *, options:* *)*

- exists.one.elim*:*

$$\forall X\,\forall A\ (\forall z{:}A\ (\forall w{:}A\ w = z \to X) \to \exists! x\ A\,x \to X)$$

exists.one.elim *added as elimination rule (abbrev:* `n` *, options:* `-i` *)*

Definition 1.10 The arrow type

`E ⇒ D`

$$E \Rightarrow D \ \texttt{:=}\ \lambda f \forall x{:}E\ D\,(f\,x)$$

The next definition is useful to get extra parenthesis.

Definition 1.11

`{{ e }}`

$$((e))\ \texttt{:=}\ e$$

1.1.5 Equality.

Axiom 1.12 equal.proposition

$$\forall X, Y\,((X \leftrightarrow Y) \to X = Y)$$

equal.proposition added as introduction rule (abbrev: `prop` , options:)

Axiom 1.13 equal.extensional

$$\forall X, Y\,(\forall x\ X\,x = Y\,x \to X = Y)$$

Proposition 1.14 equal.symmetric

$$\forall x,y\ (x = y \to y = x)$$

Proposition 1.15 equal.transitive

$$\forall x,y,z\ (x = y \to y = z \to x = z)$$

Definition 1.16

`x ≠ y`

$$x \neq y\ \texttt{:=}\ (\neg)\,(x = y)$$

Proposition 1.17 not_equal_refl

$$\forall x,y\ (x \neq y \to y \neq x)$$

Definition 1.18

`equal.decidable P`

$$\text{equal.decidable}\ P\ \texttt{:=}\ \forall x,y{:}P\ (x = y \vee x \neq y)$$

1.1.6 Some tautologies.

Proposition 1.19 int__contraposition__general

$$\forall A,B\,((A \to B) \to \forall X\,((B \to X) \to A \to X))$$

Proposition 1.20 int__contraposition

$$\forall A,B\,((A \to B) \to (\neg)\,B \to (\neg)\,A)$$

Proposition 1.21 equivalence.int__contraposition

$$\forall A,B\,((A \leftrightarrow B) \to ((\neg)\,A \leftrightarrow (\neg)\,B))$$

Proposition 1.22 equivalence.reflexive

$$\forall A\ (A \leftrightarrow A)$$

Proposition 1.23 equivalence.symmetrical

$$\forall A,B\,((A \leftrightarrow B) \to (B \leftrightarrow A))$$

Proposition 1.24 equivalence.transitive

$$\forall A,B,C\,((A \leftrightarrow B) \to (B \leftrightarrow C) \to (A \leftrightarrow C))$$

Proposition 1.25 disjunction.reflexive

$$\forall A\ (A \vee A \leftrightarrow A)$$

Proposition 1.26 disjunction.symmetrical

$$\forall A,B\,((A \vee B) \to B \vee A)$$

Proposition 1.27 disjunction.associative

$$\forall A,B,C\,((A \vee B \vee C) \to A \vee B \vee C)$$

Proposition 1.28 conjunction.reflexive

$$\forall A\ (A \wedge A \leftrightarrow A)$$

Proposition 1.29 conjunction.symmetrical

$$\forall A,B\,(A \wedge B \to B \wedge A)$$

Proposition 1.30 conjunction.associative

$$\forall A,B,C\,(A \wedge B \wedge C \to A \wedge B \wedge C)$$

Proposition 1.31 disj__conj.distributive

$$\forall A,B,C\,((A \wedge B \vee A \wedge C) \to A \wedge (B \vee C))$$

Proposition 1.32 conj__disj.distributive

$$\forall A,B,C\,((A \vee B) \wedge (A \vee C) \to A \vee B \wedge C)$$

1.1.7 Classical logic.

Axiom 1.33 peirce_law

$$\forall X, Y\, (((X \rightarrow Y) \rightarrow X) \rightarrow X)$$

If you want to do intuitionnistic logic only, do not use this axiom ! You can always use the command depend to see if a theorem uses the Peirce's law

Proposition 1.34 not_idempotent

$$\forall X\, ((\neg)\, ((\neg)\, X) \rightarrow X)$$

Proposition 1.35 absurd

$$\forall X\, (((\neg)\, X \rightarrow X) \rightarrow X)$$

Proposition 1.36 contradiction

$$\forall X\, ((\neg)\, ((\neg)\, X) \rightarrow X)$$

Proposition 1.37 excluded_middle

$$\forall X\, (X \vee (\neg)\, X)$$

Proposition 1.38 arrow_left

$$\forall A, B, X\, (((\neg)\, A \rightarrow X) \rightarrow (B \rightarrow X) \rightarrow (A \rightarrow B) \rightarrow X)$$

arrow_left added as elimination rule (abbrev: `3` , options: `-n $→`)

Proposition 1.39 forall_left

$$\forall A\ \forall X\ \forall x\, ((A\, x \rightarrow X) \rightarrow \forall x\, A\, x \rightarrow X)$$

forall_left added as elimination rule (abbrev: `n` , options: `-n`)

1.1.8 Definite description.

Constant 1.40

`Def`

$$\Delta : (\ \text{'a} \rightarrow \text{prop}\ ) \rightarrow \text{'a}$$

Axiom 1.41 Def.axiom *definite description axiom*

$$\forall P\, (\exists! z\, P\, z \rightarrow P\, (\Delta^{x}_{P\, x}))$$

Def.axiom added as introduction rule (abbrev: `Def` , options: `-o 10.0 -t`)

Proposition 1.42 Def.lemma

$$\forall P\, \left(\exists! z\, P\, z \rightarrow \forall x{:}P\, \left(\Delta^{y}_{P\, y}\right) = x\right)$$

1.1.9 Contraposition.

Proposition 1.43 contraposition

$$\forall A, B\, ((\neg)\, B \to (\neg)\, A) = (A \to B)$$

Proposition 1.44 equivalence.contraposition

$$\forall A, B\, ((\neg)\, B \leftrightarrow (\neg)\, A) = (A \leftrightarrow B)$$

Definition 1.45

List of theorems: contrapose := contraposition equivalence.contraposition

For reasoning by contraposition (classical reasoning) you can use: ”rewrite -p 0 -r contrapose.” For the intuitionnistic instance of reasoning by contraposition: rewrite contrapose.

1.1.10 De Morgan Laws.

Proposition 1.46 conjunction.demorgan

$$\forall X, Y\, (\neg)\, (X \wedge Y) = ((\neg)\, X \vee (\neg)\, Y)$$

Proposition 1.47 conjarrowleft.demorgan

$$\forall X, Y\, (\neg)\, (X \wedge Y) = (X \to (\neg)\, Y)$$

Proposition 1.48 conjarrowright.demorgan

$$\forall X, Y\, (\neg)\, (X \wedge Y) = (Y \to (\neg)\, X)$$

Proposition 1.49 disjunction.demorgan

$$\forall X, Y\, (\neg)\, (X \vee Y) = ((\neg)\, X \wedge (\neg)\, Y)$$

Proposition 1.50 arrow.demorgan

$$\forall X, Y\, (\neg)\, (X \to Y) = (X \wedge (\neg)\, Y)$$

Proposition 1.51 negation.demorgan

$$\forall X\, (\neg)\, ((\neg)\, X) = X$$

Proposition 1.52 forall.demorgan

$$\forall X\, (\neg)\, (\forall x\, X\, x) = \exists x\, (\neg)\, (X\, x)$$

Proposition 1.53 exists.demorgan

$$\forall X\,(\neg)\,(\exists x\; X\,x) = \forall x\,(\neg)\,(X\,x)$$

Definition 1.54

List of theorems: demorgan := disjunction.demorgan forall.demorgan arrow.demorgan exists.demorgan conjunction.demorgan negation.demorgan

Definition 1.55

List of theorems: demorganl := disjunction.demorgan forall.demorgan arrow.demorgan exists.demorgan conjarrowleft.demorgan negation.demorgan

Definition 1.56

List of theorems: demorganr := disjunction.demorgan forall.demorgan arrow.demorgan exists.demorgan conjarrowright.demorgan negation.demorgan

Definition 1.57

`Let x = e inside e'`

$$\mathrm{Let}\,x\texttt{=}\ e \text{ inside } e' \;\texttt{:=}\; e'$$

Proposition 1.58 and_arrow

$$\forall X, Y, Z\,((X \wedge\ Y \to\ Z) \to\ X \to\ Y \to\ Z)$$

Proposition 1.59 exists_arrow

$$\forall X\,\forall Z\,((\exists x\; X\,x \to\ Z) \to\ \forall x{:}X\ Z)$$

Chapter 2

Binary relations

2.1 Usual definitions on binary relations.

Definition 2.1

$$\text{transitive}\, D\, R \;:=\; \forall\, a, b, c{:}D\; (R\; a\; b \to\; R\; b\; c \to\; R\; a\; c)$$ `transitive D R`

Definition 2.2

$$\text{reflexive}\, D\, R \;:=\; \forall\, a{:}D\; R\; a\; a$$ `reflexive D R`

Definition 2.3

$$\text{anti.reflexive}\, D\, R \;:=\; \forall\, a{:}D\; (\neg)\; (R\; a\; a)$$ `anti.reflexive D R`

Definition 2.4

$$\text{symmetric}\, D\, R \;:=\; \forall\, a, b{:}D\; (R\; a\; b \to\; R\; b\; a)$$ `symmetric D R`

Definition 2.5

$$\text{anti.symmetric}\, D\, R \;:=\; \forall\, a, b{:}D\; (R\; a\; b \wedge\; R\; b\; a \to\; a = b)$$ `anti.symmetric D R`

Definition 2.6

$$\text{preorder}\, D\, R \;:=\; \text{transitive}\, D\, R \wedge \text{reflexive}\, D\, R$$ `preorder D R`

Definition 2.7

$$\text{strict.order}\, D\, R \;:=\; \text{transitive}\, D\, R \wedge \text{anti.reflexive}\, D\, R$$ `strict.order D R`

Definition 2.8

$$\text{order}\, D\, R \;:=\; \text{preorder}\, D\, R \wedge \text{anti.symmetric}\, D\, R$$ `order D R`

Definition 2.9

`equivalence D R` $\quad$ equivalence $D\,R$ `:=` preorder $D\,R \wedge$ symmetric $D\,R$

Definition 2.10

`total D R` $\quad$ total $D\,R$ `:=` $\forall x{,}y{:}D\ (R\,x\,y \vee\ R\,y\,x)$

Definition 2.11

`strict.total D R` $\quad$ strict.total $D\,R$ `:=` $\forall x{,}y{:}D\ (R\,x\,y \vee\ R\,y\,x \vee\ x = y)$

Definition 2.12

`well.founded D R` $\quad$ well.founded $D\,R$ `:=` $\forall X\ (\forall a{:}D\ (\forall b{:}D\ (R\,b\,a \to\ X\,b) \to\ X\,a) \to\ \forall a{:}D\ X\,a)$

Definition 2.13

`well.order D R` $\quad$ well.order $D\,R$ `:=` strict.order $D\,R \wedge$ strict.total $D\,R \wedge$ well.founded $D\,R$

Fact 2.14 Some properties of well founded relations.

- inf.well_founded*: Any subset has an inf element*

$$\forall D\,\forall R{:}(\text{well.founded } D)\ \forall X \\ (\exists x{:}D\ X\,x \to \exists x{:}D\ (X\,x \wedge \forall y{:}D\ (X\,y \to (\neg)\ (R\,y\,x))))$$

[0]written by: Christophe Raffalli, Paul Roziere (Equipe de Logique, Université Chambéry, Paris VII)

Chapter 3

The boolean

3.1 Properties of basic operations and predicates on the booleans.

3.1.1 Basic definitions.

To define the booleans, we extend the language with two contant symbols TT and FF. Then the booleans are defined by the following predicate $\mathbb{B}\,x$:

Definition 3.1 Booleans

- $\mathbb{B}\,x$ `:=` $\forall X\,(X(\top) \rightarrow X(\bot) \rightarrow X\,x)$ `B x`

3.1.2 The introduction rules for $\mathbb{B}$.

Fact 3.2 $\top$ and $\bot$ are booleans

- True.total.B*:*
$$\mathbb{B}\,(\top)$$
- False.total.B*:*
$$\mathbb{B}\,(\bot)$$

True.total.B added as introduction rule (abbrev: `True` , options: `-i -c`)
False.total.B added as introduction rule (abbrev: `False` , options: `-i -c`)

Fact 3.3 is_True.total.B

$$\forall b\,(b \rightarrow \mathbb{B}\,b)$$

is_True.total.B added as introduction rule (abbrev: `is_True` , options:)

[0]written by: Christophe Raffalli, Paul Roziere (Paris VII, Paris XII university)

Fact 3.4 is_False.total.B

$$\forall b{:}(\neg)\ \mathbb{B}\ b$$

is_False.total.B added as introduction rule (abbrev: `is_False` , options:)

3.1.3 Elimination rules for $\mathbb{B}$.

Fact 3.5 case.B *Case analysis on* $\mathbb{B}$

$$\forall X\ \forall b\ \begin{pmatrix}((\neg)\ b \to\ b = (\bot)\ \to\ X(\bot)) \to (b \to\ b = (\top)\ \to\ X(\top)) \to \\ \mathbb{B}\ b \to\ X\ b\end{pmatrix}$$

case.B added as elimination rule (abbrev: `case` , options:)

These theorems are added respectively as introduction and elimination rules for the predicate $\mathbb{B}$ with the given abbreviation (This implies for instance that `elim True.total.B` is equivalent to `intro True`). Moreover the last rule is invertible and the third rule is not necessary for completeness.

3.1.4 Left rules for $\mathbb{B}$.

Proposition 3.6 True_not_False.B $\top$ *and* $\bot$ *are distinct*

$$(\top) \neq \bot$$

Fact 3.7 True_not_False_left.B *The previous proposition as left rule*

$$\forall X\ ((\top) = (\bot)\ \to\ X)$$

True_not_False_left.B added as elimination rule (abbrev: `True_not_False` , options: `-i -n`)

Fact 3.8 False_not_True_left.B *The previous proposition as left rule*

$$\forall X\ ((\bot) = (\top)\ \to\ X)$$

False_not_True_left.B added as elimination rule (abbrev: `False_not_True` , options: `-i -n`)

Fact 3.9 equal_True_left.B *Left rule for True*

$$\forall X, b\ ((b \to\ X) \to\ b = (\top)\ \to\ X)$$

equal_True_left.B added as elimination rule (abbrev: `equal_True_left` , options: `-i -n`)

Fact 3.10 True__equal__left.B *Left rule for True*

$$\forall X, b\ ((b \to X) \to (\top) = b \to X)$$

True_equal_left.B added as elimination rule (abbrev: `left_True` , options: `-i -n`)

Fact 3.11 equal__False__left.B *Left rule for False*

$$\forall X, b\ (((\neg)\ b \to X) \to b = (\bot) \to X)$$

equal_False_left.B added as elimination rule (abbrev: `equal_False_left` , options: `-i -n`)

Fact 3.12 False__equal__left.B *Left rule for False*

$$\forall X, b\ (((\neg)\ b \to X) \to (\bot) = b \to X)$$

False_equal_left.B added as elimination rule (abbrev: `left_False` , options: `-i -n`)

Fact 3.13 elim.B *Left rule for* $\mathbb{B}$

$$\forall X, b\ ((b \to b = (\top) \to X) \to ((\neg)\ b \to b = (\bot) \to X) \to \mathbb{B}\ b \to X)$$

elim.B added as elimination rule (abbrev: `elim` , options: `-n`)

Theorem 3.14 B__is__excluded__middle.B

$$\forall x\ (\mathbb{B}\ x \leftrightarrow x \lor (\neg)\ x)$$

3.1.5 Boolean equality.

Using the previous axiom, we can prove (instuitionistically) the decidability of the equality on booleans.

Fact 3.15 eq__dec.B

$$\text{equal.decidable}\ \mathbb{B}$$

eq_dec.B added as introduction rule (abbrev: `B` , options: `-i -t`)

3.2 Definition of functions on booleans.

3.2.1 if ... then ... else ...

Definition 3.16 graph of a function defined by a test

`ifP b x y z`

$$\text{ifP } b\, x\, y\, z \;\texttt{:=}\; b \wedge z = x \vee (\neg)\, b \wedge z = y$$

Definition 3.17 function defined by a test

`if b then x else y`

$$\text{if } b \text{ then } x \text{ else } y \;\texttt{:=}\; \Delta^{z}_{\text{ifP } b\, x\, y\, z}$$

Using the definite description operator, we can introduce a new function symbol if b then x else y.

Fact 3.18 Basic properties of \$if

- True.if.B*:*

  $$\forall X\, \forall c_1, c_2\; (X \rightarrow \text{if } X \text{ then } c_1 \text{ else } c_2 = c_1)$$

  True.if.B *added as equation*

- False.if.B*:*

  $$\forall X\, \forall c_1, c_2\; ((\neg)\, X \rightarrow \text{if } X \text{ then } c_1 \text{ else } c_2 = c_2)$$

  False.if.B *added as equation*

An alternative would be to add if b then x else y as a constant and replace the previous theorems by axioms. We prefer to limit the number of axioms because this should help to detect a contradiction in our assumptions. Moreover, we are not replacing two axioms by a more powerful one, because the definite description principle is a conservative axiom.

Fact 3.19 total.if.B *Totality of* \$if

$$\forall X\, \forall b{:}\mathbb{B}\; \forall c_1, c_2{:}X\; X(\text{if } b \text{ then } c_1 \text{ else } c_2)$$

Fact 3.20 case.if.B *Totality of* \$if *a better version*

$$\forall X\, \forall b{:}\mathbb{B}\; \forall c_1, c_2\; ((b \rightarrow X\, c_1) \rightarrow ((\neg)\, b \rightarrow X\, c_2) \rightarrow X(\text{if } b \text{ then } c_1 \text{ else } c_2))$$

case.if.B added as introduction rule (abbrev: `if` , options: `-t`)

The case.if.B theorem can not be added as an introduction rule because it would be an introduction rule for any predicate ! Nevertheless it is added as a ”totality rule” (using the command `new_intro -t`. This tells the `trivial` tactic to use it when the goal is of the form $P(\text{if } b \text{ then } c_1 \text{ else } c_2)$ with P atomic. This is useful to prove that functions using \$if are total.

3.2.2 Boolean functions.

Fact 3.21 We prove the totality of these functions

- and.total.B*:*
$$\forall x,y{:}\mathbb{B}\ \mathbb{B}\,(x \wedge y)$$
- or.total.B*:*
$$\forall x,y{:}\mathbb{B}\ \mathbb{B}\,(x \vee y)$$
- neg.total.B*:*
$$\forall x{:}\mathbb{B}\ \mathbb{B}\,((\neg)\,x)$$

and.total.B *added as introduction rule (abbrev:* `and` *, options:* `-i -t` *)*

or.total.B *added as introduction rule (abbrev:* `or` *, options:* `-i -t` *)*

neg.total.B *added as introduction rule (abbrev:* `neg` *, options:* `-i -t` *)*

Fact 3.22 The equation for $\wedge$

- and.lTrue.B*:*
$$\forall x\ ((\top) \wedge x) = x$$
- and.rTrue.B*:*
$$\forall x{:}\mathbb{B}\ (x \wedge \top) = x$$
- and.lFalse.B*:*
$$\forall x\ ((\bot) \wedge x) = \bot$$
- and.rFalse.B*:*
$$\forall x{:}\mathbb{B}\ (x \wedge \bot) = \bot$$

and.rFalse.B *added as equation*

Fact 3.23 The equation for $\vee$

- or.lFalse.B*:*
$$\forall x\ ((\bot) \vee x) = x$$
- or.rFalse.B*:*
$$\forall x{:}\mathbb{B}\ (x \vee \bot) = x$$
- or.lTrue.B*:*
$$\forall x\ ((\top) \vee x) = \top$$
- or.rTrue.B*:*
$$\forall x{:}\mathbb{B}\ (x \vee \top) = \top$$

or.rFalse.B *added as equation*

Fact 3.24 The equation for $\neg$

- neg.True.B*:*

$$(\neg)\,(\top) = \bot$$

- neg.False.B*:*

$$(\neg)\,(\bot) = \top$$

  neg.False.B *added as equation*

Chapter 4

Natural numbers : second order definition

4.1 Basic definitions.

To define the natural numbers, we extend the language with one contant symbol 0 and one unary function symbol S x. Then the natural numbers are defined by the following predicate $\mathbb{N}\, x$

Definition 4.1 Church integers

- nat
- 0 : nat `N0`
- S x : nat $\to$ nat `S x`
- $\mathbb{N}\, x$ `:=` $\forall X\, (X\, 0 \to \forall y{:}X\; X\, (\mathrm{S}\; y) \to X\, x)$ `N x`

4.1.1 Introduction rules for $\mathbb{N}$.

Fact 4.2 N0.total.N 0 *is an integer*

$$\mathbb{N}\, 0$$

Fact 4.3 S.total.N *The successor function* S *is total*

$$\forall x{:}\mathbb{N}\; \mathbb{N}\, (\mathrm{S}\; x)$$

N0.total.N added as introduction rule (abbrev: `N0` , options: `-i -c`)
S.total.N added as introduction rule (abbrev: `S` , options: `-i -c`)

[0]written by: Christophe Raffalli, Paul Roziere (Paris VII, Paris XII university)

4.1.2 Elimination rules for $\mathbb{N}$.

Induction on natural number as an elimination rule.

Fact 4.4 rec.N *Induction on* $\mathbb{N}$

$$\forall X\,(X\,0 \to \forall y{:}\mathbb{N}\ (X\,y \to X\,(\mathrm{S}\ y)) \to \forall x{:}\mathbb{N}\ X\,x)$$

Elimination by case on $\mathbb{N}$.

Fact 4.5 case.N *case on* $\mathbb{N}$

$$\forall x{:}\mathbb{N}\ (x = 0 \lor \exists z{:}\mathbb{N}\ x = \mathrm{S}\ z)$$

Fact 4.6 case_left.N

$$\forall X\,\forall x\,((x = 0 \to X\,0) \to \forall y{:}\mathbb{N}\ (x = \mathrm{S}\ y \to X\,(\mathrm{S}\ y)) \to \mathbb{N}\,x \to X\,x)$$

case_left.N added as elimination rule (abbrev: `case` , options: `-n`)
rec.N added as elimination rule (abbrev: `rec` , options:)

The introduction rules are added with the command `new_intro -c`. The option `-c` indicates that 0 and S are constructors and the trivial tactic will try to use these rules when the goal is of the form $P\,0 \lor P\,(\mathrm{S}\ t)$ even if P is a unification variable.

The elimination rules are added with the command `new_elim` using the option `-n` for case.N. This tells PhoX that this second rule is not necessary for completeness.

The abbreviations are given with the rules. For instance, `elim case.N with H` is equivalent to `elim H with [case]` and `elim N0.total.N` is equivalent to `intro N0`.

4.1.3 left rules for = on $\mathbb{N}$

We add usual axioms of Peano Arithmetic.

Axiom 4.7

- N0_not_S.N*: zero and successor distinct*

$$\forall x{:}\mathbb{N}\ 0 \neq \mathrm{S}\ x$$

- S_inj.N*: successor injective*

$$\forall x,y{:}\mathbb{N}\ (\mathrm{S}\ x = \mathrm{S}\ y \to x = y)$$

We also prove the following left rules for natural numbers (the first one is an axiom).

Fact 4.8 Showing that integers are distincts.

- S_not_N0.N*:*

$$\forall x{:}\mathbb{N}\ \mathrm{S}\ x \neq 0$$

S_not_N0.N *added as elimination rule (abbrev:* `S_not_N0` *, options:* `-i -n` *)*

N0_not_S.N *added as elimination rule (abbrev:* `N0_not_S` *, options:* `-i -n` *)*

- S_inj_left.N*:*

$$\forall X\ \forall x,y{:}\mathbb{N}\ ((x = y \to X) \to \mathrm{S}\ x = \mathrm{S}\ y \to X)$$

S_inj_left.N *added as elimination rule (abbrev:* `S_inj` *, options:* `-i -n` *)*

- x_not_Sx.N*:*

$$\forall x{:}\mathbb{N}\ x \neq \mathrm{S}\ x$$

x_not_Sx.N *added as elimination rule (abbrev:* `x_not_Sx` *, options:* `-i -n` *)*

- Sx_not_x.N*:*

$$\forall x{:}\mathbb{N}\ \mathrm{S}\ x \neq x$$

Sx_not_x.N *added as elimination rule (abbrev:* `Sx_not_x` *, options:* `-i -n` *)*

4.2 Definition of functions on natural numbers.

Warning: In this section we define basic functions on natural numbers. These definitions are included in the module `nat.phx`. You can also use the module `nat_ax.phx` where they are replaced by axioms. In general, you should use the first module. But if you want to use theorems on natural numbers on a structure isomorphic to natural numbers (like the positive integers!), then you should use the module `nat_ax.phx`.

4.2.1 Definition by induction.

Definition 4.9 graph of a function defined by induction

$$\mathrm{DEF}^{rec}_{\mathbf{N}}\ a\,f\,n\,z \ := \forall X\ (X\,0\,a \to \forall y{:}\mathbb{N}\ \forall r{:}(X\,y)\ X\,(\mathrm{S}\ y)\,(f\,y\,r) \to X\,n\,z)$$ `def_rec_P.N a f n z`

The predicate $\mathrm{DEF}^{rec}_{\mathbf{N}}\ a\,f\,x\,z$ defines by induction the graph of the function which to x associate z using a as base case (when $x = 0$) and f for the successor case.

Definition 4.10 function defined by induction

`def_rec.N a f n`

$$\mathrm{def}^{rec}_{\mathbf{N}}\ a\,f\,n \ \texttt{:=}\ \Delta^{z}_{\mathrm{DEF}^{rec}_{\mathbf{N}}\ a\,f\,n\,z}$$

Using the definite description, we can introduce a function symbol $\mathrm{def}^{rec}_{\mathbf{N}}\ a\,f$ for any definition by induction on natural numbers.

Fact 4.11 Basic properties of $\mathrm{def}^{rec}_{\mathbf{N}}$

- def_rec.N0.N*:*

  $$\forall f\,\forall a\ \mathrm{def}^{rec}_{\mathbf{N}}\ a\,f\,0 = a$$

  def_rec.N0.N *added as equation*

- def_rec.S.N*:*

  $$\forall f\,\forall a\ \forall n{:}\mathbb{N}\ \mathrm{def}^{rec}_{\mathbf{N}}\ a\,f\,(\mathrm{S}\ n) = f\,n\,(\mathrm{def}^{rec}_{\mathbf{N}}\ a\,f\,n)$$

  def_rec.S.N *added as equation*

The previous proposition is proved using properties of the definite description operator.

Fact 4.12 def_rec.total.N *Totality of a function defined by induction*

$$\forall X\,\forall f{:}(\mathbb{N} \Rightarrow X \Rightarrow X)\ \forall a{:}X\ \forall n{:}\mathbb{N}\ X\,(\mathrm{def}^{rec}_{\mathbf{N}}\ a\,f\,n)$$

def_rec.total.N added as introduction rule (abbrev: `rec_def` , options: `-t`)

Application : In the following section we will define by induction addition, multiplication and exponentiation

4.3 Some very usual functions.

4.3.1 Addition.

Definition 4.13 addition

`x + y`

$$x + y \ \texttt{:=}\ \mathrm{def}^{rec}_{\mathbf{N}}\ y\,\lambda n, r\,(\mathrm{S}\ r)\,x$$

Fact 4.14 Basic properties

- add.lN0.N*:*

  $$\forall y{:}\mathbb{N}\ 0 + y = y$$

  add.lN0.N *added as equation*

- add.lS.N*:*

  $$\forall x, y{:}\mathbb{N}\ \mathrm{S}\ x + y = \mathrm{S}\ (x + y)$$

  add.lS.N *added as equation*

4.3.2 Multiplication.

Definition 4.15 multiplication

$$x.y \ := \ \mathrm{def}^{rec}_{\mathbf{N}}\, 0\, \lambda n, r\, (r + y) \qquad\qquad \texttt{x * y}$$

Fact 4.16 Basic properties

- mul.lN0.N*:*

$$\forall y{:}\mathbb{N}\ 0.y = 0$$

  mul.lN0.N *added as equation*

- mul.lS.N*:*

$$\forall x, y{:}\mathbb{N}\ \mathrm{S}\ x.y = x.y + y$$

  mul.lS.N *added as equation*

4.3.3 Exponentiation.

Definition 4.17

$$1 \ := \ \mathrm{S}\, 0 \qquad\qquad \texttt{N1}$$

N1.total.N added as introduction rule (abbrev: `N1` , options: `-i -t`)

Definition 4.18 exponentiation

$$x^y \ := \ \mathrm{def}^{rec}_{\mathbf{N}}\, 1\, \lambda n, r\, (x.r)\, y \qquad\qquad \texttt{x ^ y}$$

Fact 4.19 Basic properties

- exp.rN0.N*:*

$$\forall x{:}\mathbb{N}\ x^0 = 1$$

  exp.rN0.N *added as equation*

- exp.rS.N*:*

$$\forall x, y{:}\mathbb{N}\ x^{\mathrm{S}\ y} = x^y.x$$

  exp.rS.N *added as equation*

4.3.4 Predecessor.

We will define the predecessor as a partial function (if we define it as a total function, we can not make it coincide with the predecessor on integer when we consider natural numbers as a $\subset$ of integers).

Definition 4.20 graph of predecessor

`predP x z`

$$\text{predP}\, x\, z \;:=\; \mathbb{N}\, z \wedge x = \text{S}\, z$$

Lemma 4.21 *predP$_{unique}$* predP *defines a partial function*

$$\forall x{:}\mathbb{N}\ \exists!z\ \text{predP}\ (\text{S}\ x)\ z$$

Using the definite description operator, we can define the predecessor as follows.

Definition 4.22 predecessor

`P n`

$$\text{P}\ n \;:=\; \Delta^{z}_{\text{predP}\ n\ z}$$

Fact 4.23 pred.rS.N *Basic property of predecessor*

$$\forall n{:}\mathbb{N}\ \text{P S}\ n = n$$

pred.rS.N added as equation

4.3.5 Subtraction.

Definition 4.24 subtraction

`x - y`

$$x - y \;:=\; \text{def}^{rec}_{\mathbf{N}}\ x\, \lambda n, r\, (\text{P}\ r)\ y$$

Definition 4.25 inferior $\vee$ equal

`x ≤ y`

$$x \leq y \;:=\; \forall X\ (X\, x \to \forall z{:}X\ X\, (\text{S}\ z) \to X\, y)$$

see module `nat_ax.phx`, section ordering, for definitions of orders on natural numbers

Fact 4.26 Basic properties

- sub.rN0.N*:*
$$\forall x{:}\mathbb{N}\ x - 0 = x$$
sub.rN0.N *added as equation*

- sub.S.N*:*
$$\forall x, y{:}\mathbb{N}\ (y \leq x \to \text{S}\ x - \text{S}\ y = x - y)$$
sub.S.N *added as equation*

4.4 Properties of basic operations and predicates on $\mathbb{N}$.

4.4.1 Some constants.

Definition 4.27

$$1 := \mathrm{S}\,0 \qquad \texttt{N1}$$

Fact 4.28 N1.total.N

$$\mathbb{N}\,1$$

N1.total.N added as introduction rule (abbrev: `N1` , options: `-i -t`)

Definition 4.29

$$2 := \mathrm{S}\,1 \qquad \texttt{N2}$$

Fact 4.30 N2.total.N

$$\mathbb{N}\,2$$

N2.total.N added as introduction rule (abbrev: `N2` , options: `-i -t`)

Definition 4.31

$$3 := \mathrm{S}\,2 \qquad \texttt{N3}$$

Fact 4.32 N3.total.N

$$\mathbb{N}\,3$$

N3.total.N added as introduction rule (abbrev: `N3` , options: `-i -t`)

Definition 4.33

$$4 := \mathrm{S}\,3 \qquad \texttt{N4}$$

Fact 4.34 N4.total.N

$$\mathbb{N}\,4$$

N4.total.N added as introduction rule (abbrev: `N4` , options: `-i -t`)

Definition 4.35

$$5 := \mathrm{S}\,4 \qquad \texttt{N5}$$

Fact 4.36 N5.total.N

$$\mathbb{N}\,5$$

N5.total.N added as introduction rule (abbrev: `N5` , options: `-i -t`)

Definition 4.37

N6 $$6 := \mathrm{S}\,5$$

Fact 4.38 N6.total.N

$$\mathbb{N}\,6$$

N6.total.N added as introduction rule (abbrev: `N6` , options: `-i -t`)

Definition 4.39

N7 $$7 := \mathrm{S}\,6$$

Fact 4.40 N7.total.N

$$\mathbb{N}\,7$$

N7.total.N added as introduction rule (abbrev: `N7` , options: `-i -t`)

Definition 4.41

N8 $$8 := \mathrm{S}\,7$$

Fact 4.42 N8.total.N

$$\mathbb{N}\,8$$

N8.total.N added as introduction rule (abbrev: `N8` , options: `-i -t`)

Definition 4.43

N9 $$9 := \mathrm{S}\,8$$

Fact 4.44 N9.total.N

$$\mathbb{N}\,9$$

N9.total.N added as introduction rule (abbrev: `N9` , options: `-i -t`)

Definition 4.45

N10 $$10 := \mathrm{S}\,9$$

Fact 4.46 N10.total.N

$$\mathbb{N}\,10$$

N10.total.N added as introduction rule (abbrev: `N10` , options: `-i -t`)

Fact 4.47 some case elimination rules

- case2.N*:*

$$\forall X\,\forall x \begin{pmatrix}(x = 0 \to X\,0) \to (x = 1 \to X\,1) \to \\ \forall y{:}\mathbb{N}\ (x = \mathrm{S}\,\mathrm{S}\,y \to X\,(\mathrm{S}\,\mathrm{S}\,y)) \to \mathbb{N}\,x \to X\,x\end{pmatrix}$$

  case2.N *added as elimination rule (abbrev:* `case2` *, options:* `-n` *)*

  case_left.N *added as elimination rule (abbrev:* `case1` *, options:* *)*

- case3.N*:*

$$\forall X\,\forall x \begin{pmatrix}(x = 0 \to X\,0) \to (x = 1 \to X\,1) \to \\ (x = 2 \to X\,2) \to \\ \forall y{:}\mathbb{N}\ (x = \mathrm{S}\,\mathrm{S}\,\mathrm{S}\,y \to X\,(\mathrm{S}\,\mathrm{S}\,\mathrm{S}\,y)) \to \\ \mathbb{N}\,x \to X\,x\end{pmatrix}$$

  case3.N *added as elimination rule (abbrev:* `case3` *, options:* `-n` *)*

4.4.2 Properties of addition on $\mathbb{N}$.

Constant 4.48

$$x + y : \text{nat} \to \text{nat} \to \text{nat}$$ `x + y`

Axiom 4.49 axioms defining addition

- add.lN0.N*:*

$$\forall y{:}\mathbb{N}\ 0 + y = y$$

- add.lS.N*:*

$$\forall x,y{:}\mathbb{N}\ \mathrm{S}\,x + y = \mathrm{S}\,(x + y)$$

  add.lN0.N *added as equation*

  add.lS.N *added as equation*

These axioms are needed for axiomatic version of $\mathbb{N}$, and proved in `nat.phx`.

Fact 4.50 add.total.N *Totality of addition*

$$\forall x,y{:}\mathbb{N}\ \mathbb{N}\,(x + y)$$

add.total.N added as introduction rule (abbrev: `add` , options: `-i -t`)

We add this theorem as a totality rule (using the command `new_intro -t`) with the given abbreviation. Therefore we can use `intro add` instead of `elim add.total.N`.

Fact 4.51 exchanging sides in the properties defining addition

- add.rN0.N*:*

$$\forall x{:}\mathbb{N}\ x + 0 = x$$

  add.rN0.N *added as equation*

- add.rS.N*:*

$$\forall x, y{:}\mathbb{N}\ x + \mathrm{S}\ y = \mathrm{S}\ (x + y)$$

  add.rS.N *added as equation*

these facts are used to prove commutativity of addition.

Fact 4.52 add.commutative.N *commutativity of addition*

$$\forall x, y{:}\mathbb{N}\ x + y = y + x$$

add.commutative.N added as equation

Fact 4.53 add.associative.N *Associativity of addition*

$$\forall x, y, z{:}\mathbb{N}\ x + (y + z) = x + y + z$$

add.associative.N added as equation
added in both direction !
Following facts are useful for the performance of rewriting.

Fact 4.54 More on associativity and commutativity of addition

- add.ass_com_1.N*:*

$$\forall x, y, z{:}\mathbb{N}\ x + (y + z) = y + (x + z)$$

  add.ass_com_1.N *added as equation*

- add.ass_com_2.N*:*

$$\forall x, y, z{:}\mathbb{N}\ x + (y + z) = z + (y + x)$$

  add.ass_com_2.N *added as equation*

- add.ass_com_3.N*:*

$$\forall x, y, z{:}\mathbb{N}\ x + y + z = x + z + y$$

  add.ass_com_3.N *added as equation*

- add.ass_com_4.N*:*

$$\forall x,y,z{:}\mathbb{N}\ x+y+z=z+y+x$$

add.ass_com_4.N *added as equation*

Fact 4.55 Addition and constants

- add.rN1.N*:*

$$\forall x{:}\mathbb{N}\ x+1=\mathrm{S}\,x$$

add.rN1.N *added as equation*

- add.lN1.N*:*

$$\forall x{:}\mathbb{N}\ 1+x=\mathrm{S}\,x$$

add.lN1.N *added as equation*

- add.rN2.N*:*

$$\forall x{:}\mathbb{N}\ x+2=\mathrm{S}\,\mathrm{S}\,x$$

add.rN2.N *added as equation*

- add.lN2.N*:*

$$\forall x{:}\mathbb{N}\ 2+x=\mathrm{S}\,\mathrm{S}\,x$$

add.lN2.N *added as equation*

Fact 4.56 regularity of addition

- add.leq.N*: left regularity of addition*

$$\forall x,y,y'{:}\mathbb{N}\ (x+y=x+y'\ \to\ y=y')$$

- add.leq_left.N*: left regularity of addition, left side*

$$\forall X\ \forall x,y,y'{:}\mathbb{N}\ ((y=y'\ \to\ X)\ \to\ x+y=x+y'\ \to\ X)$$

add.leq_left.N *added as elimination rule (abbrev:* `add.leq` *, options:* `-i -rm -n` *)*

- add.req.N*: right regularity of addition*

$$\forall x,y,y'{:}\mathbb{N}\ (y+x=y'+x\to\ y=y')$$

- add.req_left.N*: right regularity of addition, left side*

$$\forall X\ \forall x,y,y'{:}\mathbb{N}\ ((y=y'\ \to\ X)\ \to\ y+x=y'+x\to\ X)$$

add.req_left.N *added as elimination rule (abbrev:* `add.req` *, options:* `-i -rm -n` *)*

added as invertible left rule.

4.4.3 Properties of multiplication.

Constant 4.57

`x × y`

$$x.y : \text{nat} \to \text{nat} \to \text{nat}$$

Axiom 4.58 Axioms defining multiplication.

- mul.lN0.N*:*

$$\forall y{:}\mathbb{N}\ 0.y = 0$$

- mul.lS.N*:*

$$\forall x, y{:}\mathbb{N}\ \text{S}\ x.y = x.y + y$$

  mul.lN0.N *added as equation*

  mul.lS.N *added as equation*

These axioms are needed for axiomatic version of $\mathbb{N}$, and proved in `nat.phx`

Fact 4.59 mul.total.N *Totality of multiplication*

$$\forall x, y{:}\mathbb{N}\ \mathbb{N}\,(x.y)$$

mul.total.N added as introduction rule (abbrev: `mul` , options: `-i -t`)

We add this theorem as a totality rule (using the command `new_intro -t`) with the given abbreviation. Therefore we can use `intro mul` instead of `elim mul.total.N`.

Fact 4.60 exchanging sides in the properties defining multiplication

- mul.rN0.N*:*

$$\forall x{:}\mathbb{N}\ x.0 = 0$$

  mul.rN0.N *added as equation*

- mul.rS.N*:*

$$\forall x, y{:}\mathbb{N}\ x.\text{S}\ y = x.y + x$$

  mul.rS.N *added as equation*

These facts are used to prove commutativity.

Fact 4.61 mul.commutative.N *Commutativity of multiplication*

$$\forall x, y{:}\mathbb{N}\ x.y = y.x$$

mul.commutative.N added as equation

distributivity has to be proved before associativity

Fact 4.62 mul.left.distributive.N *Left distributivity of multiplication on addition*

$$\forall x, y, z{:}\mathbb{N}\ x.(y + z) = x.y + x.z$$

mul.left.distributive.N added as equation

Fact 4.63 mul.right.distributive.N *Right distributivity of multiplication on addition*

$$\forall x, y, z{:}\mathbb{N}\ (y + z).x = y.x + z.x$$

mul.right.distributive.N added as equation

Fact 4.64 mul.associative.N *Associativity of multiplication*

$$\forall x, y, z{:}\mathbb{N}\ x.(y.z) = x.y.z$$

mul.associative.N added as equation

Following facts are useful for the performance of rewriting.

Fact 4.65 More on associativity and commutativity of multiplication

- mul.ass_com_1.N*:*

  $$\forall x, y, z{:}\mathbb{N}\ x.(y.z) = y.(x.z)$$

  mul.ass_com_1.N *added as equation*

- mul.ass_com_2.N*:*

  $$\forall x, y, z{:}\mathbb{N}\ x.(y.z) = z.(y.x)$$

  mul.ass_com_2.N *added as equation*

- mul.ass_com_3.N*:*

  $$\forall x, y, z{:}\mathbb{N}\ x.y.z = x.z.y$$

  mul.ass_com_3.N *added as equation*

- mul.ass_com_4.N*:*

  $$\forall x, y, z{:}\mathbb{N}\ x.y.z = z.y.x$$

  mul.ass_com_4.N *added as equation*

Fact 4.66 Multiplication and constants

- mul.rN1.N*: N1 right neutral for multiplication*

$$\forall x{:}\mathbb{N}\ x.1 = x$$

  mul.rN1.N *added as equation*

- mul.lN1.N*: N1 left neutral for multiplication*

$$\forall x{:}\mathbb{N}\ 1.x = x$$

  mul.lN1.N *added as equation*

- mul.rN2.N*:*

$$\forall x{:}\mathbb{N}\ x.2 = x + x$$

  mul.rN2.N *added as equation*

- mul.lN2.N*:*

$$\forall x{:}\mathbb{N}\ 2.x = x + x$$

  mul.lN2.N *added as equation*

Fact 4.67 mul.integr.N *Integrity for multiplication in* $\mathbb{N}$

$$\forall x,y{:}\mathbb{N}\ (x.y = 0 \to x = 0 \lor y = 0)$$

Fact 4.68 mul.lintegr.N *Integrity for multiplication in* $\mathbb{N}$

$$\forall x,y{:}\mathbb{N}\ (x.y = 0 \to y \neq 0 \to x = 0)$$

Fact 4.69 mul.rintegr.N *Integrity for multiplication in* $\mathbb{N}$

$$\forall x,y{:}\mathbb{N}\ (x.y = 0 \to x \neq 0 \to y = 0)$$

Fact 4.70 mul.integr_left.N *Integrity for multiplication in* $\mathbb{N}$ *as left rule*

$$\forall X\ \forall x,y{:}\mathbb{N}\ ((x = 0 \to X) \to (y = 0 \to X) \to x.y = 0 \to X)$$

mul.integr_left.N added as elimination rule (abbrev: `mul.integr` , options: `-i -n`)

Fact 4.71 mul.integr_left$'$ *Integrity for multiplication in* $\mathbb{N}$ *as left rule*

$$\forall X\ \forall x,y{:}\mathbb{N}\ ((x = 0 \to X) \to (y = 0 \to X) \to 0 = x.y \to X)$$

mul.integr_left$'$ added as elimination rule (abbrev: `mul.integr'` , options: `-i -n`)

Fact 4.72 regularity of multiplication

- mul.leq.N*: left regularity of multiplication*

$$\forall y,y',x{:}\mathbb{N}\ (x \neq 0 \to x.y = x.y' \to y = y')$$

- mul.leq_left.N*: left regularity of multiplication on left side*

$$\forall X\ \forall x,y,y'{:}\mathbb{N}\ ((y = y' \to X) \to x \neq 0 \to x.y = x.y' \to X)$$

  mul.leq_left.N *added as elimination rule (abbrev:* `mul.leq` *, options:* `-i -rm -n` *)*

  added as invertible left rule.

- mul.req.N*: right regularity of multiplication*

$$\forall x,y,y'{:}\mathbb{N}\ (x \neq 0 \to y.x = y'.x \to y = y')$$

- mul.req_left.N*: right regularity of multiplication on left side*

$$\forall X\ \forall x,y,y'{:}\mathbb{N}\ ((y = y' \to X) \to x \neq 0 \to y.x = y'.x \to X)$$

  mul.req_left.N *added as elimination rule (abbrev:* `mul.req` *, options:* `-i -rm -n` *)*

  added as invertible left rule.

4.4.4 Properties of exponentiation.

Constant 4.73

$$x^y : \text{nat} \to \text{nat} \to \text{nat}$$ `x ^ y`

Axiom 4.74 Axioms defining exponentiation

- exp.rN0.N*:*

$$\forall x{:}\mathbb{N}\ x^0 = 1$$

- exp.rS.N*:*

$$\forall x,y{:}\mathbb{N}\ x^{\mathrm{S}\ y} = x^y.x$$

  exp.rN0.N *added as equation*

  exp.rS.N *added as equation*

These axioms are needed for axiomatic version of $\mathbb{N}$, and proved in `nat.phx`

Fact 4.75 exp.total.N *totality of the exponentiation*

$$\forall x, y{:}\mathbb{N}\ \mathbb{N}\,(x^y)$$

exp.total.N added as introduction rule (abbrev: `exp` , options: `-i -t`)

We add this theorem as a totality rule (using the command `new_intro -t`) with the given abbreviation. Therefore we can use `intro exp` instead of `elim exp.total.N`.

Fact 4.76 exp.left.distributive.N *left "distributivity" of exponentiation*

$$\forall x, y, z{:}\mathbb{N}\ x^{y\,+\,z} = x^y.x^z$$

exp.left.distributive.N added as equation

Fact 4.77 properties of exponentiation on multiplication

- exp.composition.N*: product in exposant*

  $$\forall x, y, z{:}\mathbb{N}\ x^{y.z} = (x^y)^z$$

  exp.composition.N *added as equation*

- exp.right.distributive.N*: exponentiation of a product*

  $$\forall x, y, z{:}\mathbb{N}\ (x.y)^z = x^z.y^z$$

  exp.right.distributive.N *added as equation*

Fact 4.78 properties of exponentiation with 1

- exp.rN1.N*:* 1 *in exposant*

  $$\forall x{:}\mathbb{N}\ x^1 = x$$

  exp.rN1.N *added as equation*

- exp.lN1.N*: exponentiation of* 1

  $$\forall x{:}\mathbb{N}\ 1^x = 1$$

  exp.lN1.N *added as equation*

4.4.5 Some more constants

Definition 4.79

$$20 \ \texttt{:=} \ 10 + 10 \qquad \texttt{N20}$$

Fact 4.80 N20.total.N

$$\mathbb{N}\, 20$$

N20.total.N added as introduction rule (abbrev: `N20` , options: `-i -t`)

Definition 4.81

$$30 \ \texttt{:=} \ 10 + 20 \qquad \texttt{N30}$$

Fact 4.82 N30.total.N

$$\mathbb{N}\, 30$$

N30.total.N added as introduction rule (abbrev: `N30` , options: `-i -t`)

Definition 4.83

$$40 \ \texttt{:=} \ 10 + 30 \qquad \texttt{N40}$$

Fact 4.84 N40.total.N

$$\mathbb{N}\, 40$$

N40.total.N added as introduction rule (abbrev: `N40` , options: `-i -t`)

Definition 4.85

$$50 \ \texttt{:=} \ 10 + 40 \qquad \texttt{N50}$$

Fact 4.86 N50.total.N

$$\mathbb{N}\, 50$$

N50.total.N added as introduction rule (abbrev: `N50` , options: `-i -t`)

Definition 4.87

$$60 \ \texttt{:=} \ 10 + 50 \qquad \texttt{N60}$$

Fact 4.88 N60.total.N

$$\mathbb{N}\, 60$$

N60.total.N added as introduction rule (abbrev: `N60` , options: `-i -t`)

Definition 4.89

`N70` $$70 \; \texttt{:=} \; 10 + 60$$

Fact 4.90 N70.total.N

$$\mathbb{N}\, 70$$

N70.total.N added as introduction rule (abbrev: `N70` , options: `-i -t`)

Definition 4.91

`N80` $$80 \; \texttt{:=} \; 10 + 70$$

Fact 4.92 N80.total.N

$$\mathbb{N}\, 80$$

N80.total.N added as introduction rule (abbrev: `N80` , options: `-i -t`)

Definition 4.93

`N90` $$90 \; \texttt{:=} \; 10 + 80$$

Fact 4.94 N90.total.N

$$\mathbb{N}\, 90$$

N90.total.N added as introduction rule (abbrev: `N90` , options: `-i -t`)

Definition 4.95

`N100` $$100 \; \texttt{:=} \; 10 + 90$$

Fact 4.96 N100.total.N

$$\mathbb{N}\, 100$$

N100.total.N added as introduction rule (abbrev: `N100` , options: `-i -t`)

Definition 4.97

`N200` $$200 \; \texttt{:=} \; 100 + 100$$

Fact 4.98 N200.total.N

$$\mathbb{N}\, 200$$

N200.total.N added as introduction rule (abbrev: `N200` , options: `-i -t`)

Definition 4.99

`N300` $$300 \; \texttt{:=} \; 100 + 200$$

Fact 4.100 N300.total.N

$$\mathbb{N}\, 300$$

N300.total.N added as introduction rule (abbrev: `N300` , options: `-i -t`)

Definition 4.101

$$400 := 100 + 300 \qquad \texttt{N400}$$

Fact 4.102 N400.total.N

$$\mathbb{N}\,400$$

N400.total.N added as introduction rule (abbrev: `N400` , options: `-i -t`)

Definition 4.103

$$500 := 100 + 400 \qquad \texttt{N500}$$

Fact 4.104 N500.total.N

$$\mathbb{N}\,500$$

N500.total.N added as introduction rule (abbrev: `N500` , options: `-i -t`)

Definition 4.105

$$600 := 100 + 500 \qquad \texttt{N600}$$

Fact 4.106 N600.total.N

$$\mathbb{N}\,600$$

N600.total.N added as introduction rule (abbrev: `N600` , options: `-i -t`)

Definition 4.107

$$700 := 100 + 600 \qquad \texttt{N700}$$

Fact 4.108 N700.total.N

$$\mathbb{N}\,700$$

N700.total.N added as introduction rule (abbrev: `N700` , options: `-i -t`)

Definition 4.109

$$800 := 100 + 700 \qquad \texttt{N800}$$

Fact 4.110 N800.total.N

$$\mathbb{N}\,800$$

N800.total.N added as introduction rule (abbrev: `N800` , options: `-i -t`)

Definition 4.111

N900 $$900 := 100 + 800$$

Fact 4.112 N900.total.N
$$\mathbb{N}\, 900$$

N900.total.N added as introduction rule (abbrev: `N900` , options: `-i -t`)

Definition 4.113

N1000 $$1000 := 100 + 900$$

Fact 4.114 N1000.total.N
$$\mathbb{N}\, 1000$$

N1000.total.N added as introduction rule (abbrev: `N1000` , options: `-i -t`)

4.4.6 Ordering on N.

Definition 4.115 ordering relations on natural numbers

- x ≤ y — $x \leq y := \forall X\, (X\, x \to \forall z{:}X\; X\, (\mathrm{S}\; z) \to X\, y)$
- x < y — $x < y := \mathrm{S}\; x \leq y$
- x ≥ y — $x \geq y := y \leq x$
- x > y — $x > y := y < x$

Now we will prove some properties of these ordering relations. In fact we will only need to prove properties about [$\leq$] and a few properties about [$<$] as this is enough for reasonning.

Properties of $\leq$

Fact 4.116 introduction rules for $\leq$

- lesseq.refl.N*:*
$$\forall x{:}\mathbb{N}\;\, x \leq x$$
- lesseq.lN0.N*:*
$$\forall x{:}\mathbb{N}\;\, 0 \leq x$$
- lesseq.lS.N*:*
$$\forall x{:}\mathbb{N}\;\, \forall y\; (x \leq y \to \mathrm{S}\; x \leq \mathrm{S}\; y)$$
- lesseq.rS.N*:*
$$\forall x{:}\mathbb{N}\;\, \forall y\; (x \leq y \to x \leq \mathrm{S}\; y)$$

- lesseq.Sl.N*:*
$$\forall x,y{:}\mathbb{N}\ (\mathrm{S}\ x \leq y \to x \leq y)$$

lesseq.rS.N *added as introduction rule (abbrev:* `rS` *, options:* *)*

lesseq.lS.N *added as introduction rule (abbrev:* `lS` *, options:* `-i` *)*

lesseq.lN0.N *added as introduction rule (abbrev:* `lN0` *, options:* `-i` *)*

lesseq.refl.N *added as introduction rule (abbrev:* `refl` *, options:* `-i` *)*

Fact 4.117 elimination rules for $\leq$

- lesseq.rec.N*:*

$$\forall X\ \forall x,y{:}\mathbb{N}\ \begin{pmatrix} X\,x \to \forall z{:}\mathbb{N}\ (x \leq z \to X\,z \to X\,(\mathrm{S}\ z)) \to \\ x \leq y \to X\,y \end{pmatrix}$$

- lesseq.ltrans.N*:*

$$\forall x{:}\mathbb{N}\ \forall y,z\ (x \leq y \to y \leq z \to x \leq z)$$

- lesseq.rtrans.N*:*

$$\forall x{:}\mathbb{N}\ \forall y,z\ (y \leq z \to x \leq y \to x \leq z)$$

lesseq.rec.N *added as elimination rule (abbrev:* `rec` *, options:* `-n` *)*

lesseq.ltrans.N *added as elimination rule (abbrev:* `2` *, options:* `-t $≤` *)*

lesseq.rtrans.N *added as elimination rule (abbrev:* `2` *, options:* `-t $≤` *)*

Fact 4.118 Eliminating S both sides of $\leq$ in hypothesis

- lesseq.S_inj.N*:*
$$\forall x,y{:}\mathbb{N}\ (\mathrm{S}\ x \leq \mathrm{S}\ y \to x \leq y)$$

- lesseq.S_inj_left.N*:*

$$\forall X\ \forall x,y{:}\mathbb{N}\ ((x \leq y \to X) \to \mathrm{S}\ x \leq \mathrm{S}\ y \to X)$$

lesseq.S_inj_left.N *added as elimination rule (abbrev:* `S_inj` *, options:* `-i -rm -n` *)*

added as invertible left rule.

Fact 4.119 Eliminating $\leq$ in hypothesis

- lesseq.rN0.N*:*
$$\forall x{:}\mathbb{N}\ (x \leq 0 \to x = 0)$$

- lesseq.rN0_left.N*:*

$$\forall X\ \forall x{:}\mathbb{N}\ ((x = 0 \to X) \to x \leq 0 \to X)$$

lesseq.rN0_left.N *added as elimination rule (abbrev:* `rN0` *, options:* `-i -rm -n` *)*

added as invertible left rule.

- lesseq.or_eq_S.N*:*

$$\forall x,y{:}\mathbb{N}\ (x \leq \mathrm{S}\ y \to x \leq y \lor x = \mathrm{S}\ y)$$

- lesseq.or_eq_S_left.N*:*

$$\forall X\ \forall x,y{:}\mathbb{N}\ \begin{pmatrix}(x \leq y \to X) \to (x = \mathrm{S}\ y \to X) \to \\ x \leq \mathrm{S}\ y \to X\end{pmatrix}$$

lesseq.or_eq_S_left.N *added as elimination rule (abbrev:* `or_eq_S` *, options:* `-i -n` *)*

added as invertible left rule.

The last properties allows to replace [x ≤ Nn] where n is some integer by x= N0 ∨ ... x = Nn. With the new invertible left rules, properties like [∀ x :N (x ≤ N2 → x= N0 ∨ x= N1 ∨ x =N2)] become provable with tactic trivial.

Fact 4.120 lesseq.anti_sym.N *antisymmetry of lesseq on N*

$$\forall x,y{:}\mathbb{N}\ (x \leq y \to y \leq x \to x = y)$$

Fact 4.121 some other left rules for ≤

- lesseq.Sx_x.N*:*

$$\forall x{:}\mathbb{N}\ (\neg)\ (\mathrm{S}\ x \leq x)$$

lesseq.Sx_x.N *added as elimination rule (abbrev:* `Sx_x` *, options:* `-i -n` *)*

added as invertible left rule.

- lesseq.rN1.N*:*

$$\forall x{:}\mathbb{N}\ (\neg)\ (\mathrm{S}\ x \leq 0)$$

lesseq.rN1.N *added as elimination rule (abbrev:* `rN1` *, options:* `-i -n` *)*

added as invertible left rule.

- lesseq.S_is_S.N*:*

$$\forall x,y{:}\mathbb{N}\ (\mathrm{S}\ x \leq y \to \exists z{:}\mathbb{N}\ (y = \mathrm{S}\ z \land x \leq z))$$

- lesseq.S_is_S_left.N*:*

$$\forall X\ \forall x,y{:}\mathbb{N}\ \begin{pmatrix}\forall z{:}\mathbb{N}\ (y = \mathrm{S}\ z \to x \leq z \to X) \to \mathrm{S}\ x \leq y \to \\ X\end{pmatrix}$$

  lesseq.S_is_S_left.N *added as elimination rule (abbrev:* `S_is_S` *, options:* `-i -o 2.0 -rm -n` *)*

  added as invertible left rule.

Fact 4.122 Variations about the totality of lesseq

- lesseq.case1.N*:*

$$\forall x,y{:}\mathbb{N}\ (x \leq y \lor y < x)$$

- lesseq.case2.N*:*

$$\forall x,y{:}\mathbb{N}\ (x \leq y \to x = y \lor x < y)$$

- lesseq.case3.N*:*

$$\forall x,y{:}\mathbb{N}\ (x < y \lor x = y \lor y < x)$$

- lesseq.total.N*:*

$$\forall x,y{:}\mathbb{N}\ (x \leq y \lor y \leq x)$$

- rlesseq.total.N*:*

$$\forall x,y{:}\mathbb{N}\ (x < y \lor y \leq x)$$

- less.case.N*:*

$$\forall Q\ \forall x,y{:}\mathbb{N}\ \begin{pmatrix}(x < y \to Q) \to (x = y \to Q) \to \\ (y < x \to Q) \to Q\end{pmatrix}$$

- lesseq.case.N*:*

$$\forall Q\ \forall x,y{:}\mathbb{N}\ ((x = y \to Q) \to (x < y \to Q) \to x \leq y \to Q)$$

lesseq.case.N added as elimination rule (abbrev: `case` , options: `-n`)

less.case.N is nothing more than a version of lesseq.case3.N with a ternary disjunction

Fact 4.123 relationships between <, ≤, >, ≥

- less.imply.lesseq.N*:*

$$\forall x,y{:}\mathbb{N}\ (x < y \to x \leq y)$$

- lesseq.contradiction.N*:*

$$\forall x,y{:}\mathbb{N}\ (\neg)\ (x < y \land y \leq x)$$

- lesseq.imply.not.greater.N*:*

$$\forall x,y{:}\mathbb{N}\ (x \leq y \to (\neg)\ (y < x))$$

- not.greater.imply.lesseq.N*:*

$$\forall x,y{:}\mathbb{N}\ ((\neg)\ (x < y) \to y \leq x)$$

- less.imply.not.lesseq.N*:*

$$\forall x,y{:}\mathbb{N}\ (x < y \to (\neg)\ (y \leq x))$$

  less.imply.not.lesseq.N *added as elimination rule (abbrev:* `less.imply.not.lesseq.N` *, options:* `-i -o 1.0 -n` *)*

- not.lesseq.imply.less.N*:*

$$\forall x,y{:}\mathbb{N}\ ((\neg)\ (x \leq y) \to y < x)$$

- less_S.imply.lesseq.N*:*

$$\forall x,y{:}\mathbb{N}\ (x < \mathrm{S}\ y \to x \leq y)$$

- lesseq.imply.less_S.N*:*

$$\forall x,y{:}\mathbb{N}\ (x \leq y \to x < \mathrm{S}\ y)$$

Fact 4.124 A slightly more powerful induction rule on $\leq$

- lesseq.rec2.N*:*

$$\forall X\ \forall x,y{:}\mathbb{N} \begin{pmatrix} X\,x \to \forall z{:}\mathbb{N}\ (x \leq z \to z < y \to X\,z \to X\,(\mathrm{S}\ z)) \to \\ x \leq y \to X\,y \end{pmatrix}$$

Fact 4.125 well_founded.N $\leq$ *is well-founded on N, this is an induction principle on N*

$$\text{well.founded}\ \mathbb{N} <$$

well_founded.N added as elimination rule (abbrev: `wf` , options: `-n`)

Ordering and equality

Definition 4.126

$$x \lambda mathrel{<>}\ y := x < y \vee y < x$$ `x <> y`

Fact 4.127 less_or_sup.neq.N <> *implies* $\neq$

$$\forall x,y{:}\mathbb{N}\ (x \lambda mathrel{<>}\ y \to x \neq y)$$

Totality of order become :

Fact 4.128 neq.less_or_sup.N $\neq$ *implies* $\lambda mathrel{<>}$

$$\forall x,y{:}\mathbb{N}\ (x \neq y \to x \lambda mathrel{<>}\ y)$$

Ordering and addition

Fact 4.129 Introducing operation + in a relation using $\leq$

- lesseq.ladd.N*:*
$$\forall x,y{:}\mathbb{N}\ x \leq x + y$$
- lesseq.radd.N*:*
$$\forall x,y{:}\mathbb{N}\ x \leq y + x$$
- lesseq.add.N*:*
$$\forall x,y,x',y'{:}\mathbb{N}\ (x \leq x' \to y \leq y' \to x + y \leq x' + y')$$

these three facts added as introduction rules
lesseq.add.N added as introduction rule (abbrev: `lesseq.add` , options:)
lesseq.ladd.N added as introduction rule (abbrev: `lesseq.ladd` , options: `-i`)
lesseq.radd.N added as introduction rule (abbrev: `lesseq.radd` , options: `-i`)

Fact 4.130 Eliminating operation + in a relation using $\leq$

- lesseq.ladd_left.N*:*
$$\forall x,y,y'{:}\mathbb{N}\ (x + y \leq x + y' \to y \leq y')$$
- lesseq.ladd_rleft.N*:*
$$\forall X\ \forall x,y,y'{:}\mathbb{N}\ ((y \leq y' \to X) \to x + y \leq x + y' \to X)$$

lesseq.ladd_left.N *added as elimination rule (abbrev:* `ladd` *, options:)*

lesseq.ladd_rleft.N *added as elimination rule (abbrev:* `laddi` *, options:* `-i -n` *)*

added as invertible elimination rule.

- lesseq.radd_left.N*:*

$$\forall x, y, y' \colon \mathbb{N}\ (y + x \leq y' + x \to y \leq y')$$

- lesseq.radd_rleft.N*:*

$$\forall X\ \forall x, y, y' \colon \mathbb{N}\ ((y \leq y' \to X) \to y + x \leq y' + x \to X)$$

lesseq.radd_left.N *added as elimination rule (abbrev:* `radd` *, options:)*

lesseq.radd_rleft.N *added as elimination rule (abbrev:* `raddi` *, options:* `-i -n` *)*

added as invertible elimination rule.

Fact 4.131 From a relation with + to ≤

- ladd.lesseq.N*:*

$$\forall x, y, z \colon \mathbb{N}\ (x + y \leq z \to x \leq z)$$

ladd.lesseq.N *added as elimination rule (abbrev:* `laddo` *, options:)*

- radd.lesseq.N*:*

$$\forall x, y, z \colon \mathbb{N}\ (x + y \leq z \to y \leq z)$$

radd.lesseq.N *added as elimination rule (abbrev:* `raddo` *, options:)*

Ordering and multiplication

Fact 4.132 Introducing operation . in a relation using ≤

- lesseq.lmul.N*:*

$$\forall x, y \colon \mathbb{N}\ (y \neq 0 \to x \leq x.y)$$

- lesseq.rmul.N*:*

$$\forall x, y \colon \mathbb{N}\ (y \neq 0 \to x \leq y.x)$$

- lesseq.mul.N*:*

$$\forall x, y, x', y' \colon \mathbb{N}\ (x \leq x' \to y \leq y' \to x.y \leq x'.y')$$

lesseq.mul.N *added as introduction rule (abbrev:* `lesseq.mul` *, options:)*

lesseq.lmul.N *added as introduction rule (abbrev:* `lesseq.lmul` *, options:* `-i` *)*

lesseq.rmul.N *added as introduction rule (abbrev:* `lesseq.rmul` *, options:* `-i` *)*

these three facts are addes as introduction rules

Fact 4.133 Eliminating operation . in a relation using *≤

- lesseq.lmul_left.N*:*

$$\forall y', y, x{:}\mathbb{N}\ (x \neq 0 \rightarrow x.y \leq x.y' \rightarrow y \leq y')$$

- lesseq.lmul_rleft.N*:*

$$\forall X\ \forall y', y, x{:}\mathbb{N}\ ((y \leq y' \rightarrow X) \rightarrow x \neq 0 \rightarrow x.y \leq x.y' \rightarrow X)$$

  lesseq.lmul_left.N *added as elimination rule (abbrev:* `lmul` *, options:)*

  lesseq.lmul_rleft.N *added as elimination rule (abbrev:* `lmuli` *, options:* `-i -n` *)*

- lesseq.rmul_left.N*:*

$$\forall y', y, x{:}\mathbb{N}\ (x \neq 0 \rightarrow y.x \leq y'.x \rightarrow y \leq y')$$

  added as invertible elimination rule.

- lesseq.rmul_rleft.N*:*

$$\forall X\ \forall y', y, x{:}\mathbb{N}\ ((y \leq y' \rightarrow X) \rightarrow x \neq 0 \rightarrow y.x \leq y'.x \rightarrow X)$$

  lesseq.rmul_left.N *added as elimination rule (abbrev:* `rmul` *, options:)*

  lesseq.rmul_rleft.N *added as elimination rule (abbrev:* `rmuli` *, options:* `-i -n` *)*

  added as invertible elimination rule.

4.4.7 Predecessor, defined as a partial function on $\mathbb{N}$

Constant 4.134

$$\mathrm{P}\ x : \mathrm{nat} \rightarrow \mathrm{nat}$$ `P x`

Axiom 4.135 axioms defining predecessor

- pred.rS.N*:*

$$\forall x{:}\mathbb{N}\ \mathrm{P}\ \mathrm{S}\ x = x$$

  pred.rS.N *added as equation*

Fact 4.136 pred.total.N *"Totality" of predecessor (on its definition set)*

$$\forall x{:}\mathbb{N}\ (0 < x \rightarrow \mathbb{N}\,(\mathrm{P}\ x))$$

pred.total.N added as introduction rule (abbrev: `P` , options: `-t`)

This property is added as a totality rule (using the command `new_intro -t`) with the given abbreviation. Therefore we can use `intro pred` instead of `elim pred.total.N`

Fact 4.137 pred.lS.N

$$\forall x{:}\mathbb{N}\ (x \neq 0 \to \mathrm{S}\ \mathrm{P}\ x = x)$$

pred.lS.N added as equation

This property is added as a rewriting rule.

4.4.8 Subtraction, defined as a partial function on $\mathbb{N}$

Constant 4.138

`x - y`

$$x - y : \text{nat} \to \text{nat} \to \text{nat}$$

Axiom 4.139 axioms defining predecessor

- sub.rN0.N*:*
$$\forall x{:}\mathbb{N}\ x - 0 = x$$
- sub.S.N*:*
$$\forall x,y{:}\mathbb{N}\ (y \leq x \to \mathrm{S}\ x - \mathrm{S}\ y = x - y)$$

  sub.rN0.N *added as equation*

  sub.S.N *added as equation*

Fact 4.140 sub.total.N *"Totality" of subtraction, on its definition set*

$$\forall y,x{:}\mathbb{N}\ (y \leq x \to \mathbb{N}\,(x - y))$$

sub.total.N added as introduction rule (abbrev: `sub` , options: `-i -t`)

This property is added as a totality rule (using the command `new_intro -t`) with the given abbreviation. Therefore we can use `intro sub` instead of `elim sub`

Fact 4.141 Some useful rewriting properties on subtraction

- sub.inv.N*:*
$$\forall a{:}\mathbb{N}\ a - a = 0$$

  sub.inv.N *added as equation*

- sub.lS.N*:*
$$\forall a,b{:}\mathbb{N}\ (b \leq a \to \mathrm{S}\ a - b = \mathrm{S}\ (a - b))$$

  sub.lS.N *added as equation*

- sub.rS.N*:*
$$\forall a, b{:}\mathbb{N}\ (b < a \to a - \mathrm{S}\ b = \mathrm{P}\ (a - b))$$
sub.rS.N *added as equation*
- sub.lP.N*:*
$$\forall a, b{:}\mathbb{N}\ (b < a \to \mathrm{P}\ a - b = \mathrm{P}\ (a - b))$$
sub.lP.N *added as equation*
- sub.rP.N*:*
$$\forall a, b{:}\mathbb{N}\ (0 < b \to b \leq a \to a - \mathrm{P}\ b = \mathrm{S}\ (a - b))$$
sub.rP.N *added as equation*
- add.rsub.N*:*
$$\forall b, a{:}\mathbb{N}\ (b \leq a \to a - b + b = a)$$
add.rsub.N *added as equation*
- add.lsub.N*:*
$$\forall b, a{:}\mathbb{N}\ (b \leq a \to b + a - b = a)$$
add.lsub.N *added as equation*
- sub.radd.N*:*
$$\forall b, a{:}\mathbb{N}\ a + b - b = a$$
sub.radd.N *added as equation*
- sub.ladd.N*:*
$$\forall b, a{:}\mathbb{N}\ b + a - b = a$$
sub.ladd.N *added as equation*
- sub.less.inv.N*:*
$$\forall a, b{:}\mathbb{N}\ (a \leq b \to b - a \leq b)$$
sub.less.inv.N *added as introduction rule (abbrev:* `sub.inv` *, options:* `-i` *)*
- sub.rsub.N*:*
$$\forall b, a{:}\mathbb{N}\ (b \leq a \to a - (a - b) = b)$$
sub.rsub.N *added as equation*

all the last properties are added as rewriting rules.

Fact 4.142 Properties on $-$ and $\leq$

- lesseq.rsub.N*:*
$$\forall a, b{:}\mathbb{N}\ (b \leq a \to a - b \leq a)$$
lesseq.rsub.N *added as introduction rule (abbrev:* `rsub` *, options:* `-i` *)*

- lesseq.S_rsub.N*:*
$$\forall a, b{:}\mathbb{N}\ (b > 0 \to b \leq a \to \mathrm{S}\,(a - b) \leq a)$$
lesseq.S_rsub.N *added as introduction rule (abbrev:* `S_rsub` *, options:* `-i` *)*

- lesseq.rrsub.N*:*
$$\forall x, y, z{:}\mathbb{N}\ (x \leq y \to z \leq x \to x - z \leq y - z)$$
lesseq.rrsub.N *added as introduction rule (abbrev:* `rrsub` *, options:* `-i` *)*

- lesseq.llsub.N*:*
$$\forall x, y, z{:}\mathbb{N}\ (y \leq x \to z \leq y \to x - y \leq x - z)$$
lesseq.llsub.N *added as introduction rule (abbrev:* `llsub` *, options:* `-i` *)*

- lesseq.sub_inc.N*:*
$$\forall x, y, x', y'{:}\mathbb{N}\ (y \leq x \to x \leq x' \to y' \leq y \to x - y \leq x' - y')$$
lesseq.sub_inc.N *added as introduction rule (abbrev:* `sub_inc` *, options:)*

- lesseq.sub_radd.N*:*
$$\forall x, y, z{:}\mathbb{N}\ (y \leq x \to x \leq z + y \to x - y \leq z)$$
lesseq.sub_radd.N *added as elimination rule (abbrev:* `2` *, options:* `$≤` *)*

- lesseq.sub_ladd.N*:*
$$\forall x, y, z{:}\mathbb{N}\ (y \leq x \to z + y \leq x \to z \leq x - y)$$
lesseq.sub_ladd.N *added as elimination rule (abbrev:* `2` *, options:* `$≤` *)*

These three properties are added as introduction rules.

4.4.9 Some more properties on addition and subtraction in $\mathbb{N}$

Fact 4.143 From addition to subtraction and converse

- add_to_sub.N*:*

$$\forall a, b, c{:}\mathbb{N}\ (a + b = c \to a = c - b)$$

- sub_to_add.N*:*

$$\forall a, b, c{:}\mathbb{N}\ (b \leq a \to a - b = c \to a = c + b)$$

Fact 4.144 Permutations in expressions using + and −

- sub.rass.N*:*

$$\forall x, y, z{:}\mathbb{N}\ (z \leq y \to x + (y - z) = x + y - z)$$

  sub.rass.N *added as equation*

- sub.lass.N*:*

$$\forall x, y, z{:}\mathbb{N}\ (y + z \leq x \to x - (y + z) = x - y - z)$$

  sub.lass.N *added as equation*

- sub.comm.N*:*

$$\forall x, y, z{:}\mathbb{N}\ (z \leq x \to x + y - z = x - z + y)$$

  sub.comm.N *added as equation*

- sub.add.N*:*

$$\forall x, y, z{:}\mathbb{N}\ (y \leq x + z \to z \leq y \to x - (y - z) = x + z - y)$$

  sub.add.N *added as equation*

All these properties are added as rewriting rules.

Subtraction and multiplication

Fact 4.145 Distributivity of multiplication on subtraction

- mul.lsub.dist.N*:*

$$\forall x, y, z{:}\mathbb{N}\ (x \leq y \to (y - x).z = y.z - x.z)$$

  mul.lsub.dist.N *added as equation*

  mul.rsub.dist.N *added as equation*

These two properties are added as rewriting rules

4.4.10 Two intuitionnistic properties

Fact 4.146 odd_or_even.N *All naturals are even* $\vee$ *odd*

$$\forall x{:}\mathbb{N}\ \exists y{:}\mathbb{N}\ (x = 2.y \vee x = 1 + 2.y)$$

Fact 4.147 eq_dec.N *equality on natural numbers is decidable*

$$\text{equal.decidable}\ \mathbb{N}$$

eq_dec.N added as introduction rule (abbrev: `N` , options: `-i -t`)

4.4.11 Some more properties on multiplication and equality

Fact 4.148 rmul.neq_N1.N *Product and 1*

$$\forall x,y{:}\mathbb{N}\ (x\ \lambda mathrel{<>}\ 1 \to y.x\ \lambda mathrel{<>}\ 1)$$

Fact 4.149 rmul.eq_N1.N *Product and* 1

$$\forall x,y{:}\mathbb{N}\ (y.x = 1 \to x = 1)$$

Fact 4.150 lmul.eq_N1.N *Product and* 1

$$\forall x,y{:}\mathbb{N}\ (x.y = 1 \to x = 1)$$

Fact 4.151 mul.eq_N1.N *Product and* 1

$$\forall X\ \forall x,y{:}\mathbb{N}\ ((x = 1 \to y = 1 \to X) \to x.y = 1 \to X)$$

mul.eq_N1.N added as elimination rule (abbrev: `mul.eq_N1` , options: `-i -n`)
this property is added as invertible left rule.

Definition 4.152

List of theorems: calcul.N := add.lN0.N add.lS.N add.rN0.N add.rS.N mul.lN0.N mul.lS.N mul.rN0.N mul.rS.N exp.rN0.N exp.rS.N pred.rS.N

Chapter 5

Products

5.1 Properties of basic operations and predicates on the product.

5.1.1 Basic definitions

To define the product of two predicates, we extend the language with one binary function symbol [x, y] . Then the product of two unary predicates is defined by the following predicate $A \times B$

Definition 5.1 Product

- product['a,'b]
- x,y : 'a $\to$ 'b $\to$ 'a * 'b `x , y`
- $(A \times B)\, p$:= $\forall X\, (\forall a{:}A\ \forall b{:}B\ X(a,b) \to X\,p)$ `Product A B p`

5.1.2 The introduction rule for ×

Fact 5.2 intro.Product *Product introduction*

$$\forall A\ \forall B\ \forall x{:}A\ \forall y{:}B\ (A \times B)\,(x,y)$$

intro.Product added as introduction rule (abbrev: `i` , options: `-i -c`)

5.1.3 The elimination rules for ×

Fact 5.3 elim.Product *Product elimination*

$$\forall X\ \forall A\ \forall B\ \forall z\ (\forall x{:}A\ \forall y{:}B\ (z = x,y \to X) \to (A \times B)\, z \to X)$$

[0]written by: Christophe Raffalli, Paul Roziere (Equipe de Logique, Université Chambéry, Paris VII)

elim.Product added as elimination rule (abbrev: `r` , options: `-i`)

Axiom 5.4 injective.Product *Product injective*

$$\forall x\,\forall y\,\forall x'\,\forall y'\,(x{,}y = x'{,}y' \to x = x' \wedge y = y')$$

Fact 5.5 injective_left.Product *Product injective as left rule*

$$\forall X\,\forall x\,\forall y\,\forall x'\,\forall y'\,((x = x' \to y = y' \to X) \to x{,}y = x'{,}y' \to X)$$

injective_left.Product added as elimination rule (abbrev: `Product` , options: `-i -n`)

5.2 Projections

5.2.1 Definitions

Projections are introduced using the definite description operator on these predicates :

Definition 5.6

`fstP z x`

$$\mathrm{fstP}\,z\,x \;\texttt{:=}\; \exists y\; z = x{,}y$$

Definition 5.7

`sndP z y`

$$\mathrm{sndP}\,z\,y \;\texttt{:=}\; \exists x\; z = x{,}y$$

Definition 5.8 projections defined as functions

`fst z`
- first projection fst z `:=` $\Delta^{x}_{\mathrm{fstP}\;z\;x}$

`snd z`
- second projection snd z `:=` $\Delta^{y}_{\mathrm{sndP}\;z\;y}$

Then using the properties of the definite description, we prove the following facts.

Fact 5.9 fst.Product *property defining first projection*

$$\forall x\,\forall y\;\mathrm{fst}\,(x{,}y) = x$$

Fact 5.10 snd.Product *property defining second projection*

$$\forall x\,\forall y\;\mathrm{snd}\,(x{,}y) = y$$

We add these propositions as rewriting rules and we close the definition of fst and snd
fst.Product added as equation
snd.Product added as equation

Definition 5.11

List of theorems: calcul.Product := fst.Product snd.Product

5.2.2 Very basic facts

We also prove the following propositions :

Fact 5.12 fst.total.Product *first projection is always defined on a product*

$$\forall A\ \forall B\ \forall p{:}(A \times B)\ A\,(\mathrm{fst}\ p)$$

fst.total.Product added as introduction rule (abbrev: `fst` , options: `-t`)

Fact 5.13 snd.total.Product *second projection is always defined on a product*

$$\forall A\ \forall B\ \forall p{:}(A \times B)\ B\,(\mathrm{snd}\ p)$$

snd.total.Product added as introduction rule (abbrev: `snd` , options: `-t`)

Fact 5.14 surjective.Product *Reconstruction of a product from its projections*

$$\forall A\ \forall B\ \forall x{:}(A \times B)\ \mathrm{fst}\ x, \mathrm{snd}\ x = x$$

surjective.Product added as equation

The two first are added as totality rule and the last one is added as rewriting rule

5.3 Lexicographic ordering

Definition 5.15

lex $R_1\ R_2\ c_1\ c_2$ `:=` $R_1\,(\mathrm{fst}\ c_1)\,(\mathrm{fst}\ c_2) \vee \mathrm{fst}\ c_1 = \mathrm{fst}\ c_2 \wedge R_2\,(\mathrm{snd}\ c_1)\,(\mathrm{snd}\ c_2)$ `lex R1 R2 c1 c2`

Chapter 6

Sums

6.1 Basic definitions

To define the sums (disjoint $\cup$) of two predicates, we extend the language with two unary function symbols $\mathrm{inl}\, x$ and $\mathrm{inr}\, x$.

Sort 6.1

$$\mathrm{sum}['a,'b]$$

Constant 6.2

`inl`

$$\mathrm{inl} : \ 'a \to \mathrm{sum}['a,\ 'b]$$

Constant 6.3

`inr`

$$\mathrm{inr} : \ 'b \to \mathrm{sum}['a,\ 'b]$$

Definition 6.4 Sum of predicates

`Sum A B z`

$$(A \oplus B)\, z \ \texttt{:=}\ \forall X\, (\forall x{:}A\ X(\mathrm{inl}\, x) \to \forall y{:}B\ X(\mathrm{inr}\, y) \to X\, z)$$

Axiom 6.5

- inl.injective.Sum*: inl is injective*

$$\forall x,y\ (\mathrm{inl}\, x = \mathrm{inl}\, y \to x = y)$$

- inr.injective.Sum*: inr is injective*

$$\forall x,y\ (\mathrm{inr}\, x = \mathrm{inr}\, y \to x = y)$$

[0]written by: Christophe Raffalli, Paul Roziere (Equipe de Logique, Université Chambéry, Paris VII)

- inl_not_inr.Sum*: inl x is not inr y*

$$\forall x\,\forall y\ \mathrm{inl}\ x \neq \mathrm{inr}\ y$$

inl_not_inr.Sum added as elimination rule (abbrev: `inl_not_inr` , options: `-i -n`)

The last claim is added as invertible elimination rule.

Fact 6.6 Introduction rules for sums

- intro_left.Sum*:*

$$\forall A\ \forall B\ \forall x{:}A\ (A \oplus B)\ (\mathrm{inl}\ x)$$

- intro_right.Sum*:*

$$\forall A\ \forall B\ \forall y{:}B\ (A \oplus B)\ (\mathrm{inr}\ y)$$

intro_left.Sum *added as introduction rule (abbrev:* `l` *, options:* `-c` *)*

intro_right.Sum *added as introduction rule (abbrev:* `r` *, options:* `-c` *)*

Fact 6.7 elimination rules for sums

- elim.Sum*:*

$$\forall X\,\forall A\ \forall B\ \forall z \begin{pmatrix}\forall x{:}A\ (z = \mathrm{inl}\ x \to X) \to \\ \forall y{:}B\ (z = \mathrm{inr}\ y \to X) \to (A \oplus B)\ z \to X\end{pmatrix}$$

- inl.injective_left.Sum*:*

$$\forall X\,\forall x{,}y\ ((x = y \to X) \to \mathrm{inl}\ x = \mathrm{inl}\ y \to X)$$

- inr.injective_left.Sum*:*

$$\forall X\,\forall x{,}y\ ((x = y \to X) \to \mathrm{inr}\ x = \mathrm{inr}\ y \to X)$$

- inr_not_inl.Sum*:*

$$\forall x\,\forall y\ \mathrm{inr}\ x \neq \mathrm{inl}\ y$$

elim.Sum *added as elimination rule (abbrev:* `e` *, options:* `-i` *)*

inr_not_inl.Sum *added as elimination rule (abbrev:* `inr_not_inl` *, options:* `-i -n` *)*

inl.injective_left.Sum *added as elimination rule (abbrev:* `inl.injective` *, options:* `-i -n` *)*

inr.injective_left.Sum *added as elimination rule (abbrev:* `inr.injective` *, options:* `-i -n` *)*

These four rules and are added as invertible elimination rules.

6.2 Matching

We define

Definition 6.8

`caseP f g z r`

$$\mathrm{caseP}\, f\, g\, z\, r \ \mathtt{:=}\ \forall x\ (z = \mathrm{inl}\, x \to\ r = f\, x) \wedge \forall y\ (z = \mathrm{inr}\, y \to\ r = g\, y)$$

and we prove the following:
Using the definite description, we define:

Definition 6.9 match function on sums

`case f g z`

$$\mathrm{case}\, f\, g\, z\ \mathtt{:=}\ \Delta^{r}_{\mathrm{caseP}\ f\ g\ z\ r}$$

Then using the properties of the definite description, we prove the following propositions.

Fact 6.10 Characteristic properties of case

- case.left.Sum*: match left part of the sum*

$$\forall f\, \forall g\, \forall x\ \mathrm{case}\, f\, g\, (\mathrm{inl}\, x) = f\, x$$

- case.right.Sum*: match right part of the sum*

$$\forall f\, \forall g\, \forall y\ \mathrm{case}\, f\, g\, (\mathrm{inr}\, y) = g\, y$$

case.left.Sum *added as equation*

case.right.Sum *added as equation*

we add these facts as rewriting rules and we close the definition of case .
We also prove :

Fact 6.11 case.total.Sum *case is well defined on sums*

$$\forall A\ \forall B\ \forall C\ \forall f{:}(A\ \Rightarrow\ C)\ \forall g{:}(B \Rightarrow\ C)\ \forall z{:}(A \oplus B)\ \ C\,(\mathrm{case}\, f\, g\, z)$$

case.total.Sum added as introduction rule (abbrev: `case` , options: `-t`)

Chapter 7

Lists

7.1 Basic definitions and properties

7.1.1 Definitions and axioms

To define lists, we extend the language with one constant symbols [nil] and one binary function symbol [x :: l].

Sort 7.1

$$\mathrm{list}[\text{'a}]$$

Constant 7.2 empty list

$$\emptyset : \mathrm{list}[\text{'a}]$$ `nil`

Constant 7.3 cons

$$x{::}y : \text{'a} \to \mathrm{list}[\text{'a}] \to \mathrm{list}[\text{'a}]$$ `x :: y`

Then the list predicate is defined as follows:

Definition 7.4

$$(\mathbb{L}_D)\, x \;\texttt{:=}\; \forall X\, (X\,\emptyset \to \forall a{:}D\, \forall y{:}X\; X\,(a{::}y) \to X\,x)$$ `List D x`

We assume the following.

Axiom 7.5

- nil_not_cons.List*: empty list* $\emptyset$ *is not a* $::$ *(cons)*

$$\forall x\, \forall l\; \emptyset \neq x{::}l$$

- cons.injective.List*: injectivity of list constructor* $::$ *(cons)*

$$\forall x_1\, \forall l_1\, \forall x_2\, \forall l_2\, (x_1{::}l_1 = x_2{::}l_2 \to x_1 = x_2 \wedge l_1 = l_2)$$

[0]written by: Christophe Raffalli, Paul Roziere (Equipe de Logique, Université Chambéry, Paris VII)

7.1.2 Rules on lists

We prove the introduction and elimination rules for lists.

These rules are added respectively as introdution and elimination rules, with the given abbreviations.

Introduction rules

Fact 7.6 nil.total.List *nil is a list*

$$\forall D\ (\mathbb{L}_D)\ \emptyset$$

Fact 7.7 cons.total.List :: *(cons) is well defined*

$$\forall D\ \forall a{:}D\ \forall l{:}(\mathbb{L}_D)\ (\mathbb{L}_D)\ (a{::}l)$$

nil.total.List added as introduction rule (abbrev: `nil` , options: `-i -c`)
cons.total.List added as introduction rule (abbrev: `cons` , options: `-i -c`)

Elimination rules

Fact 7.8 rec.List *structural induction on lists*

$$\forall D\ \forall X\ \big(X\,\emptyset \to \forall a{:}D\ \forall l'{:}(\mathbb{L}_D)\ \big(X\,l' \to X\,\big(a{::}l'\big)\big) \to \forall l{:}(\mathbb{L}_D)\ X\,l\big)$$

Fact 7.9 case.List *reasoning by cases on the structure of lists*

$$\forall D\ \forall l{:}(\mathbb{L}_D)\ \big(l = \emptyset \vee \exists d{:}D\ \exists l'{:}(\mathbb{L}_D)\ l = d{::}l'\big)$$

Fact 7.10 case_left.List

$$\forall D\ \forall X\ \forall l\ \begin{pmatrix}(l = \emptyset \to X\,\emptyset) \to \forall d{:}D\ \forall l'{:}(\mathbb{L}_D)\ \big(l = d{::}l' \to X\,\big(d{::}l'\big)\big) \to \\ (\mathbb{L}_D)\ l \to X\,l\end{pmatrix}$$

rec.List added as elimination rule (abbrev: `rec` , options:)
case_left.List added as elimination rule (abbrev: `case` , options: `-n`)
case.List added as elimination rule (abbrev: `ocase` , options: `-n`)

Left rules (eliminating list constructors)

Fact 7.11 cons_not_nil.List :: *(cons) is not* $\emptyset$

$$\forall x\ \forall l\ x{::}l \neq \emptyset$$

Fact 7.12 cons.injective_left.List *injectivity of list constructor (rule form)*

$$\forall X\ \forall x_1,x_2\ \forall l_1,l_2\ ((x_1 = x_2 \to l_1 = l_2 \to X) \to x_1{::}l_1 = x_2{::}l_2 \to X)$$

These two facts and the first claim are added as invertible left rules. We close then definition of lists.
nil_not_cons.List added as elimination rule (abbrev: `nil_not_cons` , options: `-i -n`)
cons_not_nil.List added as elimination rule (abbrev: `cons_not_nil` , options: `-i -n`)
cons.injective_left.List added as elimination rule (abbrev: `cons.injective_left` , options: `-i -n`)

Fact 7.13 cons.left.List

$$\forall X\, \forall A\, \forall a\, \forall l\, ((A\, a \to (\mathbb{L}_A)\, l \to X) \to (\mathbb{L}_A)\, (a{::}l) \to X)$$

cons.left.List added as elimination rule (abbrev: `cl` , options: `-i -n`)

7.1.3 Decidability of equality

Fact 7.14 eq_dec.List *equality is decidable on lists*

$$\forall D{:}\text{equal.decidable}\ \text{equal.decidable}\ (\mathbb{L}_D)$$

eq_dec.List added as introduction rule (abbrev: `List` , options: `-i -t`)

7.2 Defining functions by induction on lists

7.2.1 Definition

In order to introduce definition of functions by structural induction on list we will use the operator Δ of definite description. We then first introduce the following predicate.

The predicate $\mathrm{DEF}^{rec}_{\mathbf{L}}\, a\, f\, l\, z$ defines a function which maps the list l to z using structural induction on the list l with a as base case (when $l = \emptyset$) and f for the cons case.

Definition 7.15 definition by induction on lists : predicate version

$$\mathrm{DEF}^{rec}_{\mathbf{L}}\, a\, f\, l\, z \;:=\; \forall X\, (X\, \emptyset\, a \to \forall l_0{:}(\mathbb{L}_{\,x\top})\; \forall x\, \forall r{:}(X\, l_0)\;\; X\, (x{::}l_0)\, (f\, x\, l_0\, r) \to X\, l\, z) \qquad \texttt{def\_rec\_P.List a f l z}$$

Note: you should remark the use of $\lambda x\top$ to use lists of *anything* !

We prove then that $\mathrm{DEF}^{rec}_{\mathbf{L}}\, a\, f\, l\, z$ effectively defines a function.

The main theorem about untyped list is the following.

Fact 7.16 True.List *Untyped list*

$$\forall D\, \forall l{:}(\mathbb{L}_D)\; (\mathbb{L}_{\lambda x\top})\, l$$

True.List added as introduction rule (abbrev: `True` , options: `-o 1.0`)

We add it as an introduction rule.

Using the definite description, we can now define an operator $\mathrm{def}^{rec}_{\mathbf{L}}$ that introduces a function symbol $\mathrm{def}^{rec}_{\mathbf{L}}\ a\,f$ for any definition by induction on lists !

Definition 7.17 definition by induction on list

`def_rec.List a f l`

$$\mathrm{def}^{rec}_{\mathbf{L}}\ a\,f\,l\ \mathtt{:=}\ \Delta^{z}_{\mathrm{DEF}^{rec}_{\mathbf{L}}\ a\,f\,l\,z}$$

Using the properties of the definite description, we can prove the following.

Fact 7.18 Characteristic properties of definitions by induction

- def_rec.nil.List*: characteristic property of definition by induction on list : base case*

$$\forall f\,\forall a\ \mathrm{def}^{rec}_{\mathbf{L}}\ a\,f\,\emptyset = a$$

- def_rec.cons.List*: characteristic property of definition by induction on list : recurrence step*

$$\forall f\,\forall a\ \forall x\ \forall l{:}(\mathbb{L}_{\lambda x\,\top})\ \mathrm{def}^{rec}_{\mathbf{L}}\ a\,f\,(x{::}l) = f\,x\,l\,(\mathrm{def}^{rec}_{\mathbf{L}}\ a\,f\,l)$$

def_rec.nil.List *added as equation*

def_rec.cons.List *added as equation*

These theorems are added as rewriting rules and then the definition of $\mathrm{def}^{rec}_{\mathbf{L}}$ is closed

We can now prove the totality of any definition by induction:

Fact 7.19 def_rec.total.List *Totality of a function defined by induction on lists*

$$\forall X\,\forall D\,\forall f{:}(D \Rightarrow\ (\mathbb{L}_D)\ \Rightarrow\ X \Rightarrow\ X)\ \forall a{:}X\,\forall l{:}(\mathbb{L}_D)\ X\,(\mathrm{def}^{rec}_{\mathbf{L}}\ a\,f\,l)$$

def_rec.total.List added as introduction rule (abbrev: `def_rec` , options: `-t`)

7.2.2 Application : operations on lists

The append function

Definition 7.20

`l @ l'`

$$l\,@\,l'\ \mathtt{:=}\ \mathrm{def}^{rec}_{\mathbf{L}}\ l'\ \lambda d,l_0,r\,(d{::}r)\ l$$

We prove the following properties of $l \,@\, l'$.

Fact 7.21 Characteristic properties of @

- append.lnil.List*:*
$$\forall l\; \emptyset \,@\, l = l$$
append.lnil.List *added as equation*

- append.lcons.List*:*
$$\forall a\; \forall l{:}(\mathbb{L}_{\lambda x\,\top})\; \forall l'\; a{::}l \,@\, l' = a{::}\big(l \,@\, l'\big)$$
append.lcons.List *added as equation*

Fact 7.22 append.total.List *totality of* @

$$\forall D\; \forall l,l'{:}(\mathbb{L}_D)\; (\mathbb{L}_D)\,\big(l \,@\, l'\big)$$

append.total.List added as introduction rule (abbrev: `append` , options: `-i -t`)

Fact 7.23 append.rnil.List

$$\forall l{:}(\mathbb{L}_{\lambda x\,\top})\; l \,@\, \emptyset = l$$

append.rnil.List added as equation

Fact 7.24 append.associative.List *associativity of* @

$$\forall x,y,z{:}(\mathbb{L}_{\lambda x\,\top})\; (x \,@\, y) \,@\, z = x \,@\, y \,@\, z$$

append.associative.List added as equation

The map functional

We define :

Definition 7.25

$$\mathrm{map}\, f\, l \;:=\; \mathrm{def}^{rec}_{\mathbf{L}}\; \emptyset\; \lambda a,l_0,r\,(f\,a{::}r)\; l$$ `map f l`

Fact 7.26 Characteristics properties of map**.**

- map.nil.List*:*
$$\forall f\, \mathrm{map}\, f \emptyset = \emptyset$$

- map.cons.List*:*
$$\forall f\, \forall a\; \forall l{:}(\mathbb{L}_{\lambda x\,\top})\; \mathrm{map}\, f\,(a{::}l) = f\,a{::}\mathrm{map}\, f\, l$$

map.nil.List added as equation
map.cons.List added as equation

Fact 7.27 map.total.List *totality of* map

$$\forall D\,\forall D'\,\forall f{:}\big(D \Rightarrow D'\big)\,\forall l{:}(\mathbb{L}_D)\,(\mathbb{L}_{D'})\,(\mathrm{map}\,f\,l)$$

map.total.List added as introduction rule (abbrev: `map` , options: `-i -t`)

Fact 7.28 map.append.List map *on* @

$$\forall f\,\forall l_1,l_2{:}(\mathbb{L}_{\lambda x\,\top})\ \mathrm{map}\,f\,(l_1\ @\ l_2) = \mathrm{map}\,f\,l_1\ @\ \mathrm{map}\,f\,l_2$$

map.append.List added as equation

7.2.3 Head and tail of a list as partial functions

Definitions

Definition 7.29 graph of head

`headP l a`

$$\mathrm{headP}\,l\,a\ := \exists l'\ l = a{::}l'$$

Definition 7.30 graph of tail

`tailP l l'`

$$\mathrm{tailP}\,l\,l'\ := \exists a\ l = a{::}l'$$

Definition 7.31 head

`head l`

$$\mathrm{head}\,l\ := \Delta^{z}_{\mathrm{headP}\,l\,z}$$

Definition 7.32 tail

`tail l`

$$\mathrm{tail}\,l\ := \Delta^{z}_{\mathrm{tailP}\,l\,z}$$

Basic facts

Fact 7.33 head.cons.List *Characteristic property of head*

$$\forall D\,\forall a{:}D\,\forall l{:}(\mathbb{L}_D)\ \mathrm{head}\,(a{::}l) = a$$

head.cons.List added as equation

Fact 7.34 tail.cons.List *Characteristic property of tail*

$$\forall D\,\forall a{:}D\,\forall l{:}(\mathbb{L}_D)\ \mathrm{tail}\,(a{::}l) = l$$

tail.cons.List added as equation

Fact 7.35 head.total.List *totality of head on its definition set*

$$\forall D\,\forall l{:}(\mathbb{L}_D)\;(l \neq \emptyset \to D\,(\text{head}\, l))$$

Fact 7.36 tail.total.List *totality of tail on its definition set*

$$\forall D\,\forall l{:}(\mathbb{L}_D)\;(l \neq \emptyset \to (\mathbb{L}_D)\,(\text{tail}\, l))$$

head.total.List added as introduction rule (abbrev: `head` , options: `-t`)
tail.total.List added as introduction rule (abbrev: `tail` , options: `-i -t`)

Fact 7.37 cons_head_tail.List

$$\forall D\,\forall l{:}(\mathbb{L}_D)\;(l \neq \emptyset \to \text{head}\, l{::}\text{tail}\, l = l)$$

cons_head_tail.List added as equation

7.2.4 Quantifiers bounded on lists.

Existence in a list

Definition

Definition 7.38 there exists x:D in l

Exists $D\,l$ `:=` $\forall X\,(\forall a\,\forall l_0\,(D\,a \to X\,(a{::}l_0)) \to \forall a\,\forall l_0{:}X\;X\,(a{::}l_0) \to X\,l)$ `Exists D l`

Introduction rules

Fact 7.39 Exists.lcons.List *left introducing* Exists

$$\forall D\,\forall a\,\forall l\,(D\,a \to \text{Exists}\, D\,(a{::}l))$$

Fact 7.40 Exists.rcons.List *right introducing* Exists

$$\forall D\,\forall a\,\forall l{:}(\text{Exists}\, D)\;\text{Exists}\, D\,(a{::}l)$$

Exists.lcons.List added as introduction rule (abbrev: `Exists.lcons` , options:)
Exists.rcons.List added as introduction rule (abbrev: `Exists.rcons` , options:)

Elimination rules

Fact 7.41 Exists.nil.List *Nothing in* $\emptyset$

$$\forall D\,(\neg)\,(\text{Exists}\, D\,\emptyset)$$

Fact 7.42 Exists.elim_cons.List *eliminating* Exists *in cons*

$$\forall D\,\forall a\,\forall l\,(\text{Exists}\, D\,(a{::}l) \to D\,a \vee \text{Exists}\, D\,l)$$

Exists.nil.List added as elimination rule (abbrev: `Exists.nil` , options:)
Exists.elim_cons.List added as elimination rule (abbrev: `Exists_cons` , options:)

Existence in append

Fact 7.43 Exists.lappend.List *left introducing* Exists *in* @

$$\forall D\ \forall l{:}(\mathbb{L}_{\lambda x\,\top})\ \forall l'\ \big(\text{Exists}\ D\ l \to \text{Exists}\ D\,\big(l\,@\,l'\big)\big)$$

Fact 7.44 Exists.rappend.List *right introducing* Exists *in* @

$$\forall D\ \forall l{:}(\mathbb{L}_{\lambda x\,\top})\ \forall l'{:}(\text{Exists}\ D)\ \text{Exists}\ D\,\big(l\,@\,l'\big)$$

Exists.lappend.List added as introduction rule (abbrev: `Exists.lappend` , options:)
Exists.rappend.List added as introduction rule (abbrev: `Exists.rappend` , options:)

Fact 7.45 Exists.elim_append.List *eliminating* Exists *in* @

$$\forall D\ \forall l{:}(\mathbb{L}_{\lambda x\,\top})\ \forall l'\ \big(\text{Exists}\ D\,\big(l\,@\,l'\big) \to \text{Exists}\ D\ l \lor \text{Exists}\ D\ l'\big)$$

Exists.elim_append.List added as elimination rule (abbrev: `Exists.elim_append` , options:)

Universal quantifer bounded on a list

Universal closure of the predicate D on the list l is exactly List D l

Definition 7.46 Forall x such that D in l

`Forall`

$$\text{Forall}\ := \mathbb{L}$$

Results on list can then be reinterpreted, for instance $\forall D\ (\mathbb{L}_D)\ \emptyset$ is $\forall D\ \mathit{forall}\ D\,\emptyset$.

It is also the case of the following facts.

Fact 7.47 List_increasing *introducing list of a type stronger*

$$\forall A, B\ (\forall x{:}A\ \ B\,x \to \forall l{:}(\mathbb{L}_A)\ (\mathbb{L}_B)\ l)$$

List_increasing added as elimination rule (abbrev: `inc` , options: `-t`)

Fact 7.48 List_conjunction *list of objects of type $A \land B$*

$$\forall A, B\ \forall l{:}(\mathbb{L}_A)\ \Big((\mathbb{L}_B)\ l \to \Big(\mathbb{L}_{\lambda x\,(A\,x\ \land\ B\,x)}\Big)\ l\Big)$$

7.2.5 Membership in a list

All facts are trivially derived as particular cases of analogous ones with Exists .

Introduction rules

Definition 7.49 membership in list

$$\text{Mem}\, x\, l \;\texttt{:=}\; \text{Exists}\,(=x)\; l$$ `Mem x l`

Fact 7.50 Mem.lcons.List *left introducing* Mem *in cons*

$$\forall\, a\; \forall\, l\; \text{Mem}\; a\,(a{::}l)$$

Fact 7.51 Mem.rcons.List *right introducing* Mem *in cons*

$$\forall\, b, a\; \forall\, l{:}(\text{Mem}\; b)\;\text{Mem}\; b\,(a{::}l)$$

Mem.lcons.List added as introduction rule (abbrev: `Mem.lcons` , options:)
Mem.rcons.List added as introduction rule (abbrev: `Mem.rcons` , options:)

Elimination rules

Fact 7.52 Mem.nil.List *no member of nil*

$$\forall\, x\;(\neg)\;(\text{Mem}\; x\,\emptyset)$$

Fact 7.53 Mem.elim__cons.List *eliminating* Mem *in cons*

$$\forall\, b, a\; \forall\, l\;(\text{Mem}\; b\,(a{::}l) \;\to\; b = a \;\vee\; \text{Mem}\; b\; l)$$

Mem.nil.List added as elimination rule (abbrev: `Mem.nil` , options:)
Mem.elim__cons.List added as elimination rule (abbrev: `Mem_cons` , options:)

Membership in append

Fact 7.54 Mem.lappend.List *left introducing* Mem *in* @

$$\forall\, b\; \forall\, l{:}(\mathbb{L}_{\lambda x\,\top})\; \forall\, l'\; \left(\text{Mem}\; b\; l \to \text{Mem}\; b\,\left(l \mathbin{@} l'\right)\right)$$

Fact 7.55 Mem.rappend.List *right introducing* Mem *in* @

$$\forall\, b\; \forall\, l{:}(\mathbb{L}_{\lambda x\,\top})\; \forall\, l'{:}(\text{Mem}\; b)\;\text{Mem}\; b\,\left(l \mathbin{@} l'\right)$$

Fact 7.56 Mem.elim__append.List *eliminating* Mem *in* @

$$\forall\, b\; \forall\, l{:}(\mathbb{L}_{\lambda x\,\top})\; \forall\, l'\; \left(\text{Mem}\; b\,\left(l \mathbin{@} l'\right) \to \text{Mem}\; b\; l \vee \text{Mem}\; b\; l'\right)$$

7.3 Data type of list of length n

7.4 Some functions using integers

we will define a partial function that return the (n-1)-th element of a given list (the first element is at position 0).

Definition 7.57 l truncated after n-th element

`nthl`

$$\text{nthl} \;\texttt{:=}\; \lambda l\,(\text{def}^{rec}_{\mathbf{N}}\; l\,\lambda n, l\,(\text{tail}\, l))$$

Definition 7.58 (n-1)-th element of l

`nth`

$$\text{nth} \;\texttt{:=}\; \lambda l, n\,(\text{head}\,(\text{nthl}\, l\, n))$$

Fact 7.59 characteristic properties of nthl and nth

- nthl.N0.List*:*
$$\forall D\, \forall l{:}(\mathbb{L}_D)\, \forall n{:}\mathbb{N}\; \text{nthl}\, l\, 0 = l$$
- nthl.S.List*:*
$$\forall D\, \forall l{:}(\mathbb{L}_D)\, \forall a{:}D\, \forall n{:}\mathbb{N}\; \text{nthl}\,(a{::}l)\,(\text{S}\; n) = \text{nthl}\, l\, n$$
- nth.N0.List*:*
$$\forall D\, \forall l{:}(\mathbb{L}_D)\, \forall a{:}D\, \forall n{:}\mathbb{N}\; \text{nth}\,(a{::}l)\, 0 = a$$
- nth.S.List*:*
$$\forall D\, \forall l{:}(\mathbb{L}_D)\, \forall a{:}D\, \forall n{:}\mathbb{N}\; \text{nth}\,(a{::}l)\,(\text{S}\; n) = \text{nth}\, l\, n$$

7.5 Some functions using integers

7.5.1 Length of a list

The length of a list is defined by :

Definition 7.60

`length l`

$$\text{length}\; l \;\texttt{:=}\; \text{def}^{rec}_{\mathbf{L}}\; 0\, \lambda x, l_0, r\,(\text{S}\; r)\; l$$

Fact 7.61 Characteristic properties of length

- length.nil.List*:*
$$\text{length}\, \emptyset = 0$$

  length.nil.List *added as equation*

- length.cons.List*:*

$$\forall a\ \forall l{:}(\mathbb{L}_{\lambda x\,\top})\ \text{length}\,(a{::}l) = \text{S length}\,l$$

length.cons.List *added as equation*

Fact 7.62 length.total.List *totality of* length

$$\forall l{:}(\mathbb{L}_{\lambda x\,\top})\ \mathbb{N}\,(\text{length}\,l)$$

length.total.List added as introduction rule (abbrev: `length` , options: `-i -t`)

Fact 7.63 length.append.List length *on* @ *is* +

$$\forall l,l'{:}(\mathbb{L}_{\lambda x\,\top})\ \text{length}\left(l\ @\ l'\right) = \text{length}\,l + \text{length}\,l'$$

length.append.List added as equation

Fact 7.64 length.map.List

$$\forall D\ \forall f\ \forall l{:}(\mathbb{L}_D)\ \text{length}\,(\text{map}\,f\,l) = \text{length}\,l$$

length.map.List added as equation

Fact 7.65 length_elim.N0.List *l with length 0 is nil*

$$\forall X\ \forall l{:}(\mathbb{L}_{\lambda x\,\top})\ ((l = \emptyset \to X) \to \text{length}\,l = 0 \to X)$$

Fact 7.66 length_elim.S.List *l with length > 0 is a cons*

$$\forall X\ \forall D\ \forall l{:}(\mathbb{L}_D)\ \left(\forall l'{:}(\mathbb{L}_D)\ \forall a{:}D\ \left(l = a{::}l' \to X\right) \to 0 < \text{length}\,l \to X\right)$$

length_elim.N0.List added as elimination rule (abbrev: `length_elim.N0` , options:)
length_elim.S.List added as elimination rule (abbrev: `length_elim.S` , options:)

7.5.2 n-th element of a list as partial function

we will define a partial function that return the (n-1)-th element of a given list (the first element is at position 0).

Constant 7.67 l truncated before n-th element

$$\text{nthl} : \text{list['a]} \to \text{nat} \to \text{list['a]}$$ `nthl`

Constant 7.68 (n-1)-th element of l

`nth` $$\text{nth} : \text{list['a]} \to \text{nat} \to \text{'a}$$

Axiom 7.69 characteristic properties of nthl and nth

- nthl.N0.List*:*
$$\forall D\ \forall l{:}(\mathbb{L}_D)\ \forall n{:}\mathbb{N}\ \text{nthl}\, l\, 0 = l$$
- nthl.S.List*:*
$$\forall D\ \forall l{:}(\mathbb{L}_D)\ \forall a{:}D\ \forall n{:}\mathbb{N}\ \text{nthl}\,(a{::}l)\,(\text{S}\ n) = \text{nthl}\, l\, n$$
- nth.N0.List*:*
$$\forall D\ \forall l{:}(\mathbb{L}_D)\ \forall a{:}D\ \forall n{:}\mathbb{N}\ \text{nth}\,(a{::}l)\ 0 = a$$
- nth.S.List*:*
$$\forall D\ \forall l{:}(\mathbb{L}_D)\ \forall a{:}D\ \forall n{:}\mathbb{N}\ \text{nth}\,(a{::}l)\,(\text{S}\ n) = \text{nth}\, l\, n$$

nthl.N0.List *added as equation*
nthl.S.List *added as equation*
nth.N0.List *added as equation*
nth.S.List *added as equation*

Fact 7.70 *nthl.tail.List*
$$\forall D\ \forall l{:}(\mathbb{L}_D)\ \forall a{:}D\ \forall n{:}\mathbb{N}\ \text{tail}\,(\text{nthl}\,(a{::}l)\ n) = \text{nthl}\, l\, n$$

Fact 7.71 nthl.total.List *totality of nthl on its definition set*
$$\forall D\ \forall l{:}(\mathbb{L}_D)\ \forall n{:}\mathbb{N}\ (n \le \text{length}\ l \to (\mathbb{L}_D)\,(\text{nthl}\, l\, n))$$

length.total.List added as introduction rule (abbrev: `nthl` , options: `-t -i`)

Fact 7.72 length.nthl.List *length of nthl l n*
$$\forall D\ \forall n{:}\mathbb{N}\ \forall l{:}(\mathbb{L}_D)\ (n \le \text{length}\ l \to \text{length}\,(\text{nthl}\, l\, n) = \text{length}\ l - n)$$

length.nthl.List added as equation

Fact 7.73 head.nthl.List *nth is head of nthl*
$$\forall D\ \forall l{:}(\mathbb{L}_D)\ \forall n{:}\mathbb{N}\ (n < \text{length}\ l \to \text{nth}\, l\, n = \text{head}\,(\text{nthl}\, l\, n))$$

length.nthl.List added as equation

Fact 7.74 nth.total.List *totality of nth on its definition set*
$$\forall D\ \forall l{:}(\mathbb{L}_D)\ \forall n{:}\mathbb{N}\ (n < \text{length}\ l \to D\,(\text{nth}\, l\, n))$$

nth.total.List added as introduction rule (abbrev: `nth` , options: `-t -i`)

Fact 7.75 lenght_induction.List
$$\forall A\ \forall X \begin{pmatrix} \forall l{:}(\mathbb{L}_A)\ \left(\forall l'{:}(\mathbb{L}_A)\ \left(\text{length}\ l' < \text{length}\ l \to X\, l'\right) \to X\, l\right) \to \\ \forall l{:}(\mathbb{L}_A)\ X\, l \end{pmatrix}$$

Chapter 8

Quotient

8.1 Basic definitions

Sort 8.1

$$\mathrm{set}$$

Constant 8.2

$$\mathrm{D} : \mathrm{set} \to \mathrm{prop}$$ `D`

Constant 8.3

$$\mathrm{R} : \mathrm{set} \to \mathrm{set} \to \mathrm{prop}$$ `R`

Axiom 8.4 refl.Q

$$\mathrm{reflexive}\,\mathrm{D}\,\mathrm{R}$$

Axiom 8.5 sym.Q

$$\mathrm{symmetric}\,\mathrm{D}\,\mathrm{R}$$

Axiom 8.6 trans.Q

$$\mathrm{transitive}\,\mathrm{D}\,\mathrm{R}$$

Definition 8.7

$$\mathrm{Q}\,X \;\texttt{:=}\; \exists x{:}\mathrm{D}\; X\,x \wedge \forall x{:}X\; \mathrm{D}\,x \wedge \forall x,y{:}\mathrm{D}\; (\mathrm{R}\,x\,y \to X\,x \to X\,y) \wedge \forall x,y\; (X\,x \to X\,y \to \mathrm{R}\,x\,y)$$ `Q X`

Definition 8.8

$$\mathrm{class}\,x\,y \;\texttt{:=}\; \mathrm{D}\,y \wedge \mathrm{R}\,x\,y$$ `class x y`

[0]written by: Christophe Raffalli (Paris VII & Paris XII university)

Proposition 8.9 class.Q

$$\forall x{:}\mathrm{D}\ \mathrm{Q}\,(\mathrm{class}\,x)$$

class.Q added as introduction rule (abbrev: `class` , options: `-c`)

Proposition 8.10 equal.class.Q

$$\forall x,y{:}\mathrm{D}\ (\mathrm{R}\,x\,y \to \mathrm{class}\,x = \mathrm{class}\,y)$$

equal.class.Q added as equation

Proposition 8.11 class.inj.Q

$$\forall x,y{:}\mathrm{D}\ (\mathrm{class}\,x = \mathrm{class}\,y \to \mathrm{R}\,x\,y)$$

Proposition 8.12 class.elim

$$\forall X\,\forall x\begin{pmatrix}\forall z{:}\mathrm{D}\ (\forall z'{:}x\ \mathrm{D}\,z' \to \forall z'{:}x\ \mathrm{R}\,z\,z' \to x\,z \to x = \mathrm{class}\,z \to X) \to \\ \mathrm{Q}\,x \to X\end{pmatrix}$$

class.elim added as elimination rule (abbrev: `rec` , options: `-i`)

Proposition 8.13 equal.Q

$$\forall x,y{:}\mathrm{Q}\ (\forall x',y'{:}\mathrm{D}\ (x\,x' \to y\,y' \to \mathrm{R}\,x'\,y') \to x = y)$$

equal.Q added as introduction rule (abbrev: `equal` , options: `-i`)

8.2 Compatible fonctions

Definition 8.14

`Compatible f R`

$$\mathrm{Compatible}\,f\,\mathrm{R}\ \texttt{:=}\ \forall x,y{:}\mathrm{D}\ (\mathrm{R}\,x\,y \to f\,x = f\,y)$$

Definition 8.15

`Lift f c z`

$$\mathrm{Lift}\,f\,c\,z\ \texttt{:=}\ \forall x{:}c\ z = f\,x$$

Proposition 8.16 lift.compatible.Q

$$\forall f\,(\mathrm{Compatible}\,f\,\mathrm{R} \to \forall c{:}\mathrm{Q}\ \exists!z\ \mathrm{Lift}\,f\,c\,z)$$

Definition 8.17

`lift f c`

$$\mathrm{lift}\,f\,c\ \texttt{:=}\ \Delta\,(\mathrm{Lift}\,f\,c)$$

Proposition 8.18 lift.total.Q

$$\forall D' \ \forall f{:}\big(\mathrm{D} \Rightarrow D'\big) \ \big(\mathrm{Compatible}\, f \mathrm{R} \to \forall c{:}\mathrm{Q} \ D' \ (\mathrm{lift}\, f\, c)\big)$$

lift.total.Q added as introduction rule (abbrev: `total` , options: `-c`)

Proposition 8.19 lift.prop

$$\forall f \,(\mathrm{Compatible}\, f \mathrm{R} \to \forall x{:}\mathrm{D} \ \mathrm{lift}\, f (\mathrm{class}\ x) = f\, x)$$

lift.prop added as equation

Proposition 8.20 class.eq.Q

$$\forall x{:}\mathrm{Q} \ \exists x'{:}x \ x = \mathrm{class}\ x'$$

Chapter 9

About the axiom of choice

Axiom 9.1 AC

$$\forall Q\ (\exists z\ Q\,z \to\ Q\,(\Delta\ Q))$$

Definition 9.2

`Def2_1 Q`

$$\mathrm{Def}_{2_1}\ Q\ := \Delta^{x}_{\exists y\ Q\,x\,y}$$

Definition 9.3

`Def2_2 Q`

$$\mathrm{Def}_{2_2}\ Q\ := \Delta^{y}_{Q\,(\mathrm{Def}_{2_1}\ Q)\ y}$$

Fact 9.4 AC2

$$\forall Q\ \left(\exists x\,\exists y\ Q\,x\,y \to\ Q\left(\mathrm{Def}_{2_1}\ Q\right)\left(\mathrm{Def}_{2_2}\ Q\right)\right)$$

Definition 9.5

`Chaine X R C`

$$\mathrm{Chaine}\,X\,R\,C\ := \exists x\ C\,x \wedge\ C\quad X \wedge\ \forall x,y{:}\,C\ (R\,x\,y \vee\ R\,y\,x)$$

Theorem 9.6 Zorn

$$\forall X\,\forall R\begin{pmatrix}\exists x\ X\,x \to \text{ order}\,X\,R \to\ \forall C{:}(\mathrm{Chaine}\,X\,R)\ \exists m{:}\,X\ \forall y{:}\,C\ R\,y\,m \to \\ \exists M{:}\,X\ \forall x{:}\,X\ (R\,M\,x \to\ M = x)\end{pmatrix}$$

Theorem 9.7 Zermelo

$$\forall X\,\exists R\ \text{well.order}\ X\,R$$

---

[0]written by: Christophe Raffalli (Paris VII $\lambda\wedge$ Paris XII university)