Infinite Lists

David Trachtenherz

September 13, 2023

Abstract

We introduce a theory of infinite lists in HOL formalized as functions over naturals (folder ListInf, theories ListInf and ListInf_Prefix). It also provides additional results for finite lists (theory ListInf/List2), natural numbers (folder CommonArith, esp. division/modulo, naturals with infinity), sets (folder CommonSet, esp. cutting/truncating sets, traversing sets of naturals).

Contents

1 Convenience results for set quantifiers 3
- 1.1 Some auxiliary results for HOL rules . . . 3
  - 1.1.1 Some auxiliary results for *Let* . . . 3
  - 1.1.2 Some auxiliary *if*-rules . . . 3
  - 1.1.3 Some auxiliary rules for function composition . . . 3
- 1.2 Some auxiliary lemmata for quantifiers . . . 3
  - 1.2.1 Auxiliary results for universal and existential quantifiers 3
  - 1.2.2 Auxiliary results for *empty* sets . . . 4
  - 1.2.3 Some auxiliary results for subset and membership relation . . . 4

2 Order and linear order: min and max 4
- 2.1 Additional lemmata about *min* and *max* . . . 4

3 Results for natural arithmetics with infinity 5
- 3.1 Arithmetic operations with *enat* . . . 5
  - 3.1.1 Additional definitions . . . 5
  - 3.1.2 Addition, difference, order . . . 6
  - 3.1.3 Multiplication and division . . . 7

4 Results for natural arithmetics 10
- 4.1 Some convenience arithmetic lemmata . . . 10
- 4.2 Additional facts about inequalities . . . 12
- 4.3 Inequalities for Suc and pred . . . 13
- 4.4 Additional facts about cancellation in (in-)equalities . . . 13

5 Results for division and modulo operators on integers **16**
5.1 Additional (in-)equalities with *div* and *mod* 16
5.2 Additional results for addition and subtraction with *mod* . . . 17
5.2.1 Divisor subtraction with *div* and *mod* 19
5.2.2 Modulo equality and modulo of difference 20
5.3 Some additional lemmata about integer *div* and *mod* 20
5.4 Some further (in-)equality results for *div* and *mod* 22
5.5 Additional multiplication results for *mod* and *div* 23
5.6 Some factor distribution facts for *mod* 23
5.7 More results about quotient *div* with addition and subtraction 24
5.8 Further results about *div* and *mod* 25
5.8.1 Some auxiliary facts about *mod* 25
5.8.2 Some auxiliary facts about *div* 26

6 Sets of natural numbers **30**
6.1 Auxiliary results for monotonic, injective and surjective functions over sets 30
6.1.1 Monotonicity 30
6.1.2 Injectivity 31
6.1.3 Surjectivity 32
6.1.4 Induction over natural sets 33
6.1.5 Monotonicity and injectivity of artithmetic operators . 34
6.2 *Min* and *Max* elements of a set 35
6.2.1 Basic results, as for *Least* 35
6.2.2 *Max* for sets over *enat* 39
6.2.3 *Min* and *Max* for set operations 40
6.3 Some auxiliary results for set operations 42
6.3.1 Some additional abbreviations for relations 42
6.3.2 Auxiliary results for *singletons* 43
6.3.3 Auxiliary results for *finite* and *infinite* sets 44
6.3.4 Some auxiliary results for disjoint sets 45
6.3.5 Some auxiliary results for subset relation 46
6.3.6 Auxiliary results for intervals from *SetInterval* 46
6.3.7 Auxiliary results for *card* 49

7 Cutting linearly ordered and natural sets **50**
7.1 Set restriction 50
7.2 Cut operators for sets/intervals 52
7.2.1 Definitions and basic lemmata for cut operators 52
7.2.2 Basic results for cut operators 54
7.2.3 Relations between cut operators 61
7.2.4 Function images with cut operators 61
7.2.5 Finiteness and cardinality with cut operators 62
7.2.6 Cutting a set at *Min* or *Max* element 63

7.2.7 Cut operators with intervals from SetInterval 65
7.2.8 Mirroring finite natural sets between their *Min* and *Max* element 66

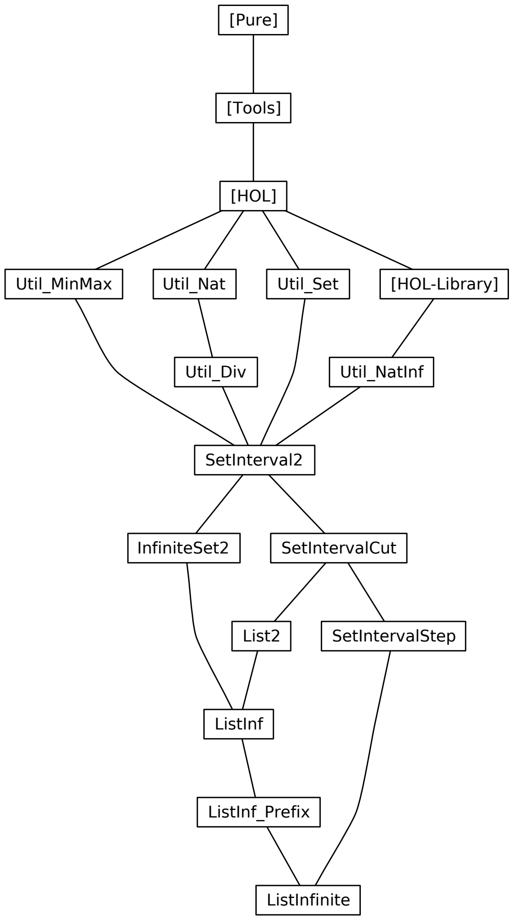
8 Stepping through sets of natural numbers 71
8.1 Function *inext* and *iprev* for stepping through natural sets . . 71
8.2 *inext-nth* and *iprev-nth* – nth element of a natural set 83
8.3 Induction over arbitrary natural sets using the functions *inext* and *iprev* 87
8.4 Natural intervals with *inext* and *iprev* 88
8.5 Further result for *inext-nth* and *iprev-nth* 89

9 Additional definitions and results for lists 90
9.1 Additional definitions and results for lists 90
9.1.1 Additional lemmata about list emptiness 91
9.1.2 Additional lemmata about *take*, *drop*, *hd*, *last*, *nth* and *filter* 91
9.1.3 Ordered lists 94
9.1.4 Additional definitions and results for sublists 97
9.1.5 Natural set images with lists 100
9.1.6 Mapping lists of functions to lists 102
9.1.7 Mapping functions with two arguments to lists 104

10 Set operations with results of type enat 107
10.1 Set operations with *enat* 107
10.1.1 Basic definitions 107
10.2 Results for *icard* 107

11 Additional definitions and results for lists 110
11.1 Infinite lists 111
11.1.1 Appending a functions to a list 111
11.1.2 *take* and *drop* for infinite lists 115
11.1.3 *zip* for infinite lists 121
11.1.4 Mapping functions with two arguments to infinite lists 122
11.2 Generalised lists as combination of finite and infinite lists . . 123
11.2.1 Basic definitions 123
11.2.2 *glength* 124
11.2.3 $@_g$ – gappend 125
11.2.4 *gmap* 125
11.2.5 *gset* 126
11.2.6 $!_g$ – gnth 126
11.2.7 *gtake* and *gdrop* 127

12 Prefices on finite and infinite lists **128**
12.1 Additional list prefix results 128
12.2 Counting equal pairs 129
12.3 Prefix length 131
12.4 Prefix infimum 133
12.5 Prefices for infinite lists 135



1 Convenience results for set quantifiers

theory *Util-Set*
imports *Main*
begin

1.1 Some auxiliary results for HOL rules

lemma *conj-disj-absorb*: $(P \wedge Q \vee Q) = Q$ ⟨*proof*⟩
lemma *disj-eq-distribL*: $((a \vee b) = (a \vee c)) = (a \vee (b = c))$ ⟨*proof*⟩
lemma *disj-eq-distribR*: $((a \vee c) = (b \vee c)) = ((a = b) \vee c)$ ⟨*proof*⟩

1.1.1 Some auxiliary results for *Let*

lemma *Let-swap*: $f\ (\mathit{let}\ x{=}a\ \mathit{in}\ g\ x) = (\mathit{let}\ x{=}a\ \mathit{in}\ f\ (g\ x))$ ⟨*proof*⟩

1.1.2 Some auxiliary *if*-rules

lemma *if-P'*: $[\![\ P;\ x = z\ ]\!] \Longrightarrow (\mathit{if}\ P\ \mathit{then}\ x\ \mathit{else}\ y) = z$ ⟨*proof*⟩
lemma *if-not-P'*: $[\![\ \neg\ P;\ y = z\ ]\!] \Longrightarrow (\mathit{if}\ P\ \mathit{then}\ x\ \mathit{else}\ y) = z$ ⟨*proof*⟩

lemma *if-P-both*: $[\![\ Q\ x;\ Q\ y\ ]\!] \Longrightarrow Q\ (\mathit{if}\ P\ \mathit{then}\ x\ \mathit{else}\ y)$ ⟨*proof*⟩
lemma *if-P-both-in-set*: $[\![\ x \in s;\ y \in s\ ]\!] \Longrightarrow (\mathit{if}\ P\ \mathit{then}\ x\ \mathit{else}\ y) \in s$ ⟨*proof*⟩

1.1.3 Some auxiliary rules for function composition

lemma *comp2-conv*: $f1 \circ f2 = (\lambda x.\ f1\ (f2\ x))$ ⟨*proof*⟩
lemma *comp3-conv*: $f1 \circ f2 \circ f3 = (\lambda x.\ f1\ (f2\ (f3\ x)))$ ⟨*proof*⟩

1.2 Some auxiliary lemmata for quantifiers

1.2.1 Auxiliary results for universal and existential quantifiers

lemma *ball-cong2*:
$[\![\ I \subseteq A;\ \forall x{\in}A.\ f\ x = g\ x\ ]\!] \Longrightarrow (\forall x{\in}I.\ P\ (f\ x)) = (\forall x{\in}I.\ P\ (g\ x))$ ⟨*proof*⟩
lemma *bex-cong2*:
$[\![\ I \subseteq A;\ \forall x{\in}I.\ f\ x = g\ x\ ]\!] \Longrightarrow (\exists x{\in}I.\ P\ (f\ x)) = (\exists x{\in}I.\ P\ (g\ x))$ ⟨*proof*⟩
lemma *ball-all-cong*:
$\forall x.\ f\ x = g\ x \Longrightarrow (\forall x{\in}I.\ P\ (f\ x)) = (\forall x{\in}I.\ P\ (g\ x))$ ⟨*proof*⟩
lemma *bex-all-cong*:
$\forall x.\ f\ x = g\ x \Longrightarrow (\exists x{\in}I.\ P\ (f\ x)) = (\exists x{\in}I.\ P\ (g\ x))$ ⟨*proof*⟩
lemma *all-cong*:
$\forall x.\ f\ x = g\ x \Longrightarrow (\forall x.\ P\ (f\ x)) = (\forall x.\ P\ (g\ x))$ ⟨*proof*⟩
lemma *ex-cong*:
$\forall x.\ f\ x = g\ x \Longrightarrow (\exists x.\ P\ (f\ x)) = (\exists x.\ P\ (g\ x))$ ⟨*proof*⟩

lemmas *all-eqI* = *iff-allI*

lemmas *ex-eqI* = *iff-exI*

lemma *all-imp-eqI*:
$[\![P = P'; \bigwedge x.\ P\ x \Longrightarrow Q\ x = Q'\ x]\!] \Longrightarrow$
$(\forall x.\ P\ x \longrightarrow Q\ x) = (\forall x.\ P'\ x \longrightarrow Q'\ x)$
$\langle proof \rangle$
lemma *ex-imp-eqI*:
$[\![P = P'; \bigwedge x.\ P\ x \Longrightarrow Q\ x = Q'\ x]\!] \Longrightarrow$
$(\exists x.\ P\ x \wedge Q\ x) = (\exists x.\ P'\ x \wedge Q'\ x)$
$\langle proof \rangle$

1.2.2 Auxiliary results for *empty* sets

lemma *empty-imp-not-in*: $x \notin \{\}$ $\langle proof \rangle$
lemma *ex-imp-not-empty*: $\exists x.\ x \in A \Longrightarrow A \neq \{\}$ $\langle proof \rangle$
lemma *in-imp-not-empty*: $x \in A \Longrightarrow A \neq \{\}$ $\langle proof \rangle$
lemma *not-empty-imp-ex*: $A \neq \{\} \Longrightarrow \exists x.\ x \in A$ $\langle proof \rangle$
lemma *not-ex-in-conv*: $(\neg (\exists x.\ x \in A)) = (A = \{\})$ $\langle proof \rangle$

1.2.3 Some auxiliary results for subset and membership relation

lemma *bex-subset-imp-bex*: $[\![\exists x{\in}A.\ P\ x; A \subseteq B]\!] \Longrightarrow \exists x{\in}B.\ P\ x$ $\langle proof \rangle$
lemma *bex-imp-ex*: $\exists x{\in}A.\ P\ x \Longrightarrow \exists x.\ P\ x$ $\langle proof \rangle$
lemma *ball-subset-imp-ball*: $[\![\forall x{\in}B.\ P\ x; A \subseteq B]\!] \Longrightarrow \forall x{\in}A.\ P\ x$ $\langle proof \rangle$
lemma *all-imp-ball*: $\forall x.\ P\ x \Longrightarrow \forall x{\in}A.\ P\ x$ $\langle proof \rangle$

lemma *mem-Collect-eq-not*: $(a \notin \{x.\ P\ x\}) = (\neg\ P\ a)$ $\langle proof \rangle$
lemma *Collect-not-in-imp-not*: $a \notin \{x.\ P\ x\} \Longrightarrow \neg\ P\ a$ $\langle proof \rangle$
lemma *Collect-not-imp-not-in*: $\neg\ P\ a \Longrightarrow a \notin \{x.\ P\ x\}$ $\langle proof \rangle$
lemma *Collect-is-subset*: $\{x \in A.\ P\ x\} \subseteq A$ $\langle proof \rangle$

end

2 Order and linear order: min and max

theory *Util-MinMax*
imports *Main*
begin

2.1 Additional lemmata about *min* and *max*

lemma *min-less-imp-conj*: $(z::'a::linorder) < min\ x\ y \Longrightarrow z < x \wedge z < y$ $\langle proof \rangle$
lemma *conj-less-imp-min*: $[\![z < x; z < y]\!] \Longrightarrow (z::'a::linorder) < min\ x\ y$ $\langle proof \rangle$

lemmas *min-le-iff-conj* = *min.bounded-iff*
lemma *min-le-imp-conj*: $(z::'a::linorder) \leq min\ x\ y \Longrightarrow z \leq x \wedge z \leq y$ $\langle proof \rangle$

lemmas *conj-le-imp-min* = *min.boundedI*

lemmas *min-eqL* = *min.absorb1*

lemmas *min-eqR* = *min.absorb2*
lemmas *min-eq* = *min-eqL min-eqR*

lemma *max-less-imp-conj*: $max\ x\ y < b \Longrightarrow x < (b::('a::linorder)) \wedge y < b$ ⟨*proof*⟩
lemma *conj-less-imp-max*: $[\![x < (b::('a::linorder)); y < b]\!] \Longrightarrow max\ x\ y < b$ ⟨*proof*⟩

lemmas *max-le-iff-conj* = *max.bounded-iff*
lemma *max-le-imp-conj*: $max\ x\ y \le b \Longrightarrow x \le (b::('a::linorder)) \wedge y \le b$ ⟨*proof*⟩

lemmas *conj-le-imp-max* = *max.boundedI*

lemmas *max-eqL* = *max.absorb1*

lemmas *max-eqR* = *max.absorb2*
lemmas *max-eq* = *max-eqL max-eqR*

lemmas *le-minI1* = *min.cobounded1*

lemmas *le-minI2* = *min.cobounded2*

lemma
min-le-monoR: $(a::'a::linorder) \le b \Longrightarrow min\ x\ a \le min\ x\ b$ **and**
min-le-monoL: $(a::'a::linorder) \le b \Longrightarrow min\ a\ x \le min\ b\ x$
⟨*proof*⟩
lemma
max-le-monoR: $(a::'a::linorder) \le b \Longrightarrow max\ x\ a \le max\ x\ b$ **and**
max-le-monoL: $(a::'a::linorder) \le b \Longrightarrow max\ a\ x \le max\ b\ x$
⟨*proof*⟩

end

3 Results for natural arithmetics with infinity

theory *Util-NatInf*
imports *HOL−Library.Extended-Nat*
begin

3.1 Arithmetic operations with *enat*

3.1.1 Additional definitions

instantiation *enat* :: *modulo*
begin

definition
div-enat-def [*code del*]:
$a\ div\ b \equiv (case\ a\ of$
$(enat\ x) \Rightarrow (case\ b\ of\ (enat\ y) \Rightarrow enat\ (x\ div\ y) \mid \infty \Rightarrow 0) \mid$
$\infty \Rightarrow (case\ b\ of\ (enat\ y) \Rightarrow ((case\ y\ of\ 0 \Rightarrow 0 \mid Suc\ n \Rightarrow \infty)) \mid \infty \Rightarrow \infty\ ))$
definition
mod-enat-def [*code del*]:
$a\ mod\ b \equiv (case\ a\ of$
$(enat\ x) \Rightarrow (case\ b\ of\ (enat\ y) \Rightarrow enat\ (x\ mod\ y) \mid \infty \Rightarrow a) \mid$
$\infty \Rightarrow \infty)$

instance ⟨*proof*⟩

end

lemmas *enat-arith-defs* =
zero-enat-def one-enat-def
plus-enat-def diff-enat-def times-enat-def div-enat-def mod-enat-def
declare *zero-enat-def*[*simp*]

lemmas *ineq0-conv-enat*[*simp*] = *i0-less*[*symmetric*, *unfolded zero-enat-def*]

lemmas *iless-eSuc0-enat*[*simp*] = *iless-eSuc0*[*unfolded zero-enat-def*]

3.1.2 Addition, difference, order

lemma *diff-eq-conv-nat*: $(x - y = (z{::}nat)) = (if\ y < x\ then\ x = y + z\ else\ z = 0)$
⟨*proof*⟩
lemma *idiff-eq-conv*:
$(x - y = (z{::}enat)) =$
$(if\ y < x\ then\ x = y + z\ else\ if\ x \neq \infty\ then\ z = 0\ else\ z = \infty)$
⟨*proof*⟩
lemmas *idiff-eq-conv-enat* = *idiff-eq-conv*[*unfolded zero-enat-def*]

lemma *less-eq-idiff-eq-sum*: $y \leq (x{::}enat) \Longrightarrow (z \leq x - y) = (z + y \leq x)$
⟨*proof*⟩

lemma *eSuc-pred*: $0 < n \Longrightarrow eSuc\ (n - eSuc\ 0) = n$
⟨*proof*⟩
lemmas *eSuc-pred-enat* = *eSuc-pred*[*unfolded zero-enat-def*]
lemmas *iadd-0-enat*[*simp*] = *add-0-left*[**where** *'a = enat*, *unfolded zero-enat-def*]
lemmas *iadd-0-right-enat*[*simp*] = *add-0-right*[**where** *'a=enat*, *unfolded zero-enat-def*]

lemma *ile-add1*: $(n{::}enat) \leq n + m$

⟨*proof*⟩
lemma *ile-add2*: $(n::enat) \le m + n$
⟨*proof*⟩

lemma *iadd-iless-mono*: $[\![(i::enat) < j; k < l]\!] \Longrightarrow i + k < j + l$
⟨*proof*⟩

lemma *trans-ile-iadd1*: $i \le (j::enat) \Longrightarrow i \le j + m$
⟨*proof*⟩
lemma *trans-ile-iadd2*: $i \le (j::enat) \Longrightarrow i \le m + j$
⟨*proof*⟩

lemma *trans-iless-iadd1*: $i < (j::enat) \Longrightarrow i < j + m$
⟨*proof*⟩
lemma *trans-iless-iadd2*: $i < (j::enat) \Longrightarrow i < m + j$
⟨*proof*⟩

lemma *iadd-ileD1*: $m + k \le (n::enat) \Longrightarrow m \le n$
⟨*proof*⟩

lemma *iadd-ileD2*: $m + k \le (n::enat) \Longrightarrow k \le n$
⟨*proof*⟩

lemma *idiff-ile-mono*: $m \le (n::enat) \Longrightarrow m - l \le n - l$
⟨*proof*⟩

lemma *idiff-ile-mono2*: $m \le (n::enat) \Longrightarrow l - n \le l - m$
⟨*proof*⟩

lemma *idiff-iless-mono*: $[\![m < (n::enat); l \le m]\!] \Longrightarrow m - l < n - l$
⟨*proof*⟩

lemma *idiff-iless-mono2*: $[\![m < (n::enat); m < l]\!] \Longrightarrow l - n \le l - m$
⟨*proof*⟩

3.1.3 Multiplication and division

lemmas *imult-infinity-enat*[*simp*] = *imult-infinity*[*unfolded zero-enat-def*]
lemmas *imult-infinity-right-enat*[*simp*] = *imult-infinity-right*[*unfolded zero-enat-def*]

lemma *idiv-enat-enat*[*simp*, *code*]: *enat a div enat b* = *enat* (*a div b*)
⟨*proof*⟩

lemma *idiv-infinity*: $0 < n \Longrightarrow (\infty::enat)$ *div* $n = \infty$
⟨*proof*⟩

lemmas *idiv-infinity-enat*[*simp*] = *idiv-infinity*[*unfolded zero-enat-def*]

lemma *idiv-infinity-right*[*simp*]: $n \neq \infty \Longrightarrow n\ div\ (\infty{::}enat) = 0$
⟨*proof*⟩

lemma *idiv-infinity-if*: $n\ div\ \infty = (if\ n = \infty\ then\ \infty\ else\ 0{::}enat)$
⟨*proof*⟩

lemmas *idiv-infinity-if-enat* = *idiv-infinity-if*[*unfolded zero-enat-def*]

lemmas *imult-0-enat*[*simp*] = *mult-zero-left*[**where** *'a=enat*,*unfolded zero-enat-def*]
lemmas *imult-0-right-enat*[*simp*] = *mult-zero-right*[**where** *'a=enat*,*unfolded zero-enat-def*]

lemmas *imult-is-0-enat* = *imult-is-0*[*unfolded zero-enat-def*]
lemmas *enat-0-less-mult-iff-enat* = *enat-0-less-mult-iff*[*unfolded zero-enat-def*]

lemma *imult-infinity-if*: $\infty * n = (if\ n = 0\ then\ 0\ else\ \infty{::}enat)$
⟨*proof*⟩
lemma *imult-infinity-right-if*: $n * \infty = (if\ n = 0\ then\ 0\ else\ \infty{::}enat)$
⟨*proof*⟩
lemmas *imult-infinity-if-enat* = *imult-infinity-if*[*unfolded zero-enat-def*]
lemmas *imult-infinity-right-if-enat* = *imult-infinity-right-if*[*unfolded zero-enat-def*]

lemmas *imult-is-infinity-enat* = *imult-is-infinity*[*unfolded zero-enat-def*]

lemma *idiv-by-0*: $(a{::}enat)\ div\ 0 = 0$
⟨*proof*⟩
lemmas *idiv-by-0-enat*[*simp*, *code*] = *idiv-by-0*[*unfolded zero-enat-def*]

lemma *idiv-0*: $0\ div\ (a{::}enat) = 0$
⟨*proof*⟩
lemmas *idiv-0-enat*[*simp*, *code*] = *idiv-0*[*unfolded zero-enat-def*]

lemma *imod-by-0*: $(a{::}enat)\ mod\ 0 = a$
⟨*proof*⟩
lemmas *imod-by-0-enat*[*simp*, *code*] = *imod-by-0*[*unfolded zero-enat-def*]

lemma *imod-0*: $0\ mod\ (a{::}enat) = 0$
⟨*proof*⟩
lemmas *imod-0-enat*[*simp*, *code*] = *imod-0*[*unfolded zero-enat-def*]

lemma *imod-enat-enat*[*simp*, *code*]: $enat\ a\ mod\ enat\ b = enat\ (a\ mod\ b)$
⟨*proof*⟩
lemma *imod-infinity*[*simp*, *code*]: $\infty\ mod\ n = (\infty{::}enat)$
⟨*proof*⟩
lemma *imod-infinity-right*[*simp*, *code*]: $n\ mod\ (\infty{::}enat) = n$
⟨*proof*⟩

lemma *idiv-self*: $[\![\ 0 < (n{::}enat);\ n \neq \infty\ ]\!] \Longrightarrow n\ div\ n = 1$
⟨*proof*⟩
lemma *imod-self*: $n \neq \infty \Longrightarrow (n{::}enat)\ mod\ n = 0$

⟨*proof*⟩

lemma *idiv-iless*: $m < (n{::}enat) \Longrightarrow m\ div\ n = 0$
⟨*proof*⟩
lemma *imod-iless*: $m < (n{::}enat) \Longrightarrow m\ mod\ n = m$
⟨*proof*⟩

lemma *imod-iless-divisor*: $[\![\ 0 < (n{::}enat);\ m \neq \infty\ ]\!] \Longrightarrow m\ mod\ n < n$
⟨*proof*⟩
lemma *imod-ile-dividend*: $(m{::}enat)\ mod\ n \le m$
⟨*proof*⟩
lemma *idiv-ile-dividend*: $(m{::}enat)\ div\ n \le m$
⟨*proof*⟩

lemma *idiv-imult2-eq*: $(a{::}enat)\ div\ (b * c) = a\ div\ b\ div\ c$
⟨*proof*⟩

lemma *imult-ile-mono*: $[\![\ (i{::}enat) \le j;\ k \le l\ ]\!] \Longrightarrow i * k \le j * l$
⟨*proof*⟩

lemma *imult-ile-mono1*: $(i{::}enat) \le j \Longrightarrow i * k \le j * k$
⟨*proof*⟩

lemma *imult-ile-mono2*: $(i{::}enat) \le j \Longrightarrow k * i \le k * j$
⟨*proof*⟩

lemma *imult-iless-mono1*: $[\![\ (i{::}enat) < j;\ 0 < k;\ k \neq \infty\ ]\!] \Longrightarrow i * k \le j * k$
⟨*proof*⟩
lemma *imult-iless-mono2*: $[\![\ (i{::}enat) < j;\ 0 < k;\ k \neq \infty\ ]\!] \Longrightarrow k * i \le k * j$
⟨*proof*⟩

lemma *imod-1*: $(enat\ m)\ mod\ eSuc\ 0 = 0$
⟨*proof*⟩
lemmas *imod-1-enat*[*simp*, *code*] = *imod-1*[*unfolded zero-enat-def*]

lemma *imod-iadd-self2*: $(m + enat\ n)\ mod\ (enat\ n) = m\ mod\ (enat\ n)$
⟨*proof*⟩

lemma *imod-iadd-self1*: $(enat\ n + m)\ mod\ (enat\ n) = m\ mod\ (enat\ n)$
⟨*proof*⟩

lemma *idiv-imod-equality*: $(m{::}enat)\ div\ n * n + m\ mod\ n + k = m + k$
⟨*proof*⟩
lemma *imod-idiv-equality*: $(m{::}enat)\ div\ n * n + m\ mod\ n = m$
⟨*proof*⟩

lemma *idiv-ile-mono*: $m \le (n{::}enat) \Longrightarrow m\ div\ k \le n\ div\ k$
⟨*proof*⟩
lemma *idiv-ile-mono2*: $[\![\ 0 < m;\ m \le (n{::}enat)\ ]\!] \Longrightarrow k\ div\ n \le k\ div\ m$

$\langle proof \rangle$

end

4 Results for natural arithmetics

theory *Util-Nat*
imports *Main*
begin

4.1 Some convenience arithmetic lemmata

lemma *add-1-Suc-conv*: $m + 1 = Suc\ m$ $\langle proof \rangle$
lemma *sub-Suc0-sub-Suc-conv*: $b - a - Suc\ 0 = b - Suc\ a$ $\langle proof \rangle$

lemma *Suc-diff-Suc*: $m < n \Longrightarrow Suc\ (n - Suc\ m) = n - m$
$\langle proof \rangle$

lemma *nat-grSuc0-conv*: $(Suc\ 0 < n) = (n \neq 0 \wedge n \neq Suc\ 0)$
$\langle proof \rangle$

lemma *nat-geSucSuc0-conv*: $(Suc\ (Suc\ 0) \leq n) = (n \neq 0 \wedge n \neq Suc\ 0)$
$\langle proof \rangle$

lemma *nat-lessSucSuc0-conv*: $(n < Suc\ (Suc\ 0)) = (n = 0 \vee n = Suc\ 0)$
$\langle proof \rangle$

lemma *nat-leSuc0-conv*: $(n \leq Suc\ 0) = (n = 0 \vee n = Suc\ 0)$
$\langle proof \rangle$

lemma *mult-pred*: $(m - Suc\ 0) * n = m * n - n$
$\langle proof \rangle$

lemma *mult-pred-right*: $m * (n - Suc\ 0) = m * n - m$
$\langle proof \rangle$

lemma *gr-implies-gr0*: $m < (n::nat) \Longrightarrow 0 < n$ $\langle proof \rangle$

corollary *mult-cancel1-gr0*:
$(0::nat) < k \Longrightarrow (k * m = k * n) = (m = n)$ $\langle proof \rangle$
corollary *mult-cancel2-gr0*:
$(0::nat) < k \Longrightarrow (m * k = n * k) = (m = n)$ $\langle proof \rangle$

corollary *mult-le-cancel1-gr0*:
$(0::nat) < k \Longrightarrow (k * m \leq k * n) = (m \leq n)$ $\langle proof \rangle$
corollary *mult-le-cancel2-gr0*:
$(0::nat) < k \Longrightarrow (m * k \leq n * k) = (m \leq n)$ $\langle proof \rangle$

lemma *gr0-imp-self-le-mult1*: $0 < (k::nat) \Longrightarrow m \leq m * k$

$\langle proof \rangle$

lemma *gr0-imp-self-le-mult2*: $0 < (k::nat) \Longrightarrow m \leq k * m$
$\langle proof \rangle$

lemma *less-imp-Suc-mult-le*: $m < n \Longrightarrow Suc\ m * k \leq n * k$
$\langle proof \rangle$

lemma *less-imp-Suc-mult-pred-less*: $[\![\ m < n;\ 0 < k\ ]\!] \Longrightarrow Suc\ m * k - Suc\ 0 < n * k$
$\langle proof \rangle$

lemma *ord-zero-less-diff*: $(0 < (b::'a::ordered\text{-}ab\text{-}group\text{-}add) - a) = (a < b)$
$\langle proof \rangle$

lemma *ord-zero-le-diff*: $(0 \leq (b::'a::ordered\text{-}ab\text{-}group\text{-}add) - a) = (a \leq b)$
$\langle proof \rangle$

diff-diff-right in rule format

lemmas *diff-diff-right* = *Nat.diff-diff-right*[*rule-format*]

lemma *less-add1*: $(0::nat) < j \Longrightarrow i < i + j$ $\langle proof \rangle$
lemma *less-add2*: $(0::nat) < j \Longrightarrow i < j + i$ $\langle proof \rangle$

lemma *add-lessD2*: $i + j < (k::nat) \Longrightarrow j < k$ $\langle proof \rangle$

lemma *add-le-mono2*: $i \leq (j::nat) \Longrightarrow k + i \leq k + j$ $\langle proof \rangle$

lemma *add-less-mono2*: $i < (j::nat) \Longrightarrow k + i < k + j$ $\langle proof \rangle$

lemma *diff-less-self*: $[\![\ (0::nat) < i;\ \ 0 < j\ ]\!] \Longrightarrow i - j < i$ $\langle proof \rangle$

lemma
ge-less-neq-conv: $((a::'a::linorder) \leq n) = (\forall x.\ x < a \longrightarrow n \neq x)$ **and**
le-greater-neq-conv: $(n \leq (a::'a::linorder)) = (\forall x.\ a < x \longrightarrow n \neq x)$
$\langle proof \rangle$
lemma
greater-le-neq-conv: $((a::'a::linorder) < n) = (\forall x.\ x \leq a \longrightarrow n \neq x)$ **and**
less-ge-neq-conv: $(n < (a::'a::linorder)) = (\forall x.\ a \leq x \longrightarrow n \neq x)$
$\langle proof \rangle$

Lemmas for @termabs function

lemma *leq-pos-imp-abs-leq*: $[\![\ 0 \leq (a::'a::ordered\text{-}ab\text{-}group\text{-}add\text{-}abs);\ a \leq b\ ]\!] \Longrightarrow |a| \leq |b|$
$\langle proof \rangle$
lemma *leq-neg-imp-abs-geq*: $[\![\ (a::'a::ordered\text{-}ab\text{-}group\text{-}add\text{-}abs) \leq 0;\ b \leq a\ ]\!] \Longrightarrow |a| \leq |b|$

$\langle proof \rangle$
lemma *abs-range*: $[\![\; 0 \le (a::'a::\{ordered\text{-}ab\text{-}group\text{-}add\text{-}abs,abs\text{-}if\}); \; -a \le x; \; x \le a \;]\!] \Longrightarrow |x| \le a$
$\langle proof \rangle$

Lemmas for @termsgn function

lemma *sgn-abs*:$(x::'a::linordered\text{-}idom) \ne 0 \Longrightarrow |sgn\ x| = 1$
$\langle proof \rangle$
lemma *sgn-mult-abs*:$|x| * |sgn\ (a::'a::linordered\text{-}idom)| = |x * sgn\ a|$
$\langle proof \rangle$
lemma *abs-imp-sgn-abs*: $|a| = |b| \Longrightarrow |sgn\ (a::'a::linordered\text{-}idom)| = |sgn\ b|$
$\langle proof \rangle$
lemma *sgn-mono*: $a \le b \Longrightarrow sgn\ (a::'a::\{linordered\text{-}idom,linordered\text{-}semidom\}) \le sgn\ b$
$\langle proof \rangle$

4.2 Additional facts about inequalities

lemma *add-diff-le*: $k \le n \Longrightarrow m + k - n \le (m::nat)$
$\langle proof \rangle$

lemma *less-add-diff*: $k < (n::nat) \Longrightarrow m < n + m - k$

$\langle proof \rangle$

lemma *add-diff-less*: $[\![\; k < n; \; 0 < m \;]\!] \Longrightarrow m + k - n < (m::nat)$
$\langle proof \rangle$

lemma *add-le-imp-le-diff1*: $i + k \le j \Longrightarrow i \le j - (k::nat)$
$\langle proof \rangle$

lemma *add-le-imp-le-diff2*: $k + i \le j \Longrightarrow i \le j - (k::nat)$ $\langle proof \rangle$

lemma *diff-less-imp-less-add*: $j - (k::nat) < i \Longrightarrow j < i + k$ $\langle proof \rangle$

lemma *diff-less-conv*: $0 < i \Longrightarrow (j - (k::nat) < i) = (j < i + k)$
$\langle proof \rangle$

lemma *le-diff-swap*: $[\![\; i \le (k::nat); \; j \le k \;]\!] \Longrightarrow (k - j \le i) = (k - i \le j)$
$\langle proof \rangle$

lemma *diff-less-imp-swap*: $[\![\; 0 < (i::nat); \; k - i < j \;]\!] \Longrightarrow (k - j < i)$ $\langle proof \rangle$
lemma *diff-less-swap*: $[\![\; 0 < (i::nat); \; 0 < j \;]\!] \Longrightarrow (k - j < i) = (k - i < j)$
$\langle proof \rangle$

lemma *less-diff-imp-less*: $(i::nat) < j - m \Longrightarrow i < j$ $\langle proof \rangle$
lemma *le-diff-imp-le*: $(i::nat) \le j - m \Longrightarrow i \le j$ $\langle proof \rangle$

lemma *less-diff-le-imp-less*: $[\![\ (i::nat) < j - m;\ n \le m\ ]\!] \Longrightarrow i < j - n$ ⟨*proof*⟩
lemma *le-diff-le-imp-le*: $[\![\ (i::nat) \le j - m;\ n \le m\ ]\!] \Longrightarrow i \le j - n$ ⟨*proof*⟩

lemma *le-imp-diff-le*: $(j::nat) \le k \Longrightarrow j - n \le k$ ⟨*proof*⟩

4.3 Inequalities for Suc and pred

corollary *less-eq-le-pred*: $0 < (n::nat) \Longrightarrow (m < n) = (m \le n - Suc\ 0)$ ⟨*proof*⟩

corollary *less-imp-le-pred*: $m < n \Longrightarrow m \le n - Suc\ 0$ ⟨*proof*⟩
corollary *le-pred-imp-less*: $[\![\ 0 < n;\ m \le n - Suc\ 0\ ]\!] \Longrightarrow m < n$ ⟨*proof*⟩

corollary *pred-less-eq-le*: $0 < m \Longrightarrow (m - Suc\ 0 < n) = (m \le n)$ ⟨*proof*⟩
corollary *pred-less-imp-le*: $m - Suc\ 0 < n \Longrightarrow m \le n$ ⟨*proof*⟩
corollary *le-imp-pred-less*: $[\![\ 0 < m;\ m \le n\ ]\!] \Longrightarrow m - Suc\ 0 < n$ ⟨*proof*⟩

lemma *diff-add-inverse-Suc*: $n < m \Longrightarrow n + (m - Suc\ n) = m - Suc\ 0$ ⟨*proof*⟩

lemma *pred-mono*: $[\![\ m < n;\ 0 < m\ ]\!] \Longrightarrow m - Suc\ 0 < n - Suc\ 0$ ⟨*proof*⟩
corollary *pred-Suc-mono*: $[\![\ m < Suc\ n;\ 0 < m\ ]\!] \Longrightarrow m - Suc\ 0 < n$ ⟨*proof*⟩

lemma *Suc-less-pred-conv*: $(Suc\ m < n) = (m < n - Suc\ 0)$ ⟨*proof*⟩
lemma *Suc-le-pred-conv*: $0 < n \Longrightarrow (Suc\ m \le n) = (m \le n - Suc\ 0)$ ⟨*proof*⟩
lemma *Suc-le-imp-le-pred*: $Suc\ m \le n \Longrightarrow m \le n - Suc\ 0$ ⟨*proof*⟩

4.4 Additional facts about cancellation in (in-)equalities

lemma *diff-cancel-imp-eq*: $[\![\ 0 < (n::nat);\ \ n + i - j = n\ ]\!] \Longrightarrow i = j$ ⟨*proof*⟩

lemma *nat-diff-left-cancel-less*: $k - m < k - (n::nat) \Longrightarrow n < m$ ⟨*proof*⟩
lemma *nat-diff-right-cancel-less*: $n - k < (m::nat) - k \Longrightarrow n < m$ ⟨*proof*⟩

lemma *nat-diff-left-cancel-le1*: $[\![\ k - m \le k - (n::nat);\ m < k\ ]\!] \Longrightarrow n \le m$ ⟨*proof*⟩
lemma *nat-diff-left-cancel-le2*: $[\![\ k - m \le k - (n::nat);\ n \le k\ ]\!] \Longrightarrow n \le m$ ⟨*proof*⟩

lemma *nat-diff-right-cancel-le1*: $[\![\ m - k \le n - (k::nat);\ k < m\ ]\!] \Longrightarrow m \le n$ ⟨*proof*⟩
lemma *nat-diff-right-cancel-le2*: $[\![\ m - k \le n - (k::nat);\ k \le n\ ]\!] \Longrightarrow m \le n$ ⟨*proof*⟩

lemma *nat-diff-left-cancel-eq1*: $[\![\ k - m = k - (n::nat);\ m < k\ ]\!] \Longrightarrow m = n$ ⟨*proof*⟩
lemma *nat-diff-left-cancel-eq2*: $[\![\ k - m = k - (n::nat);\ n < k\ ]\!] \Longrightarrow m = n$ ⟨*proof*⟩

lemma *nat-diff-right-cancel-eq1*: $[\![\ m - k = n - (k::nat);\ k < m\ ]\!] \Longrightarrow m = n$ ⟨*proof*⟩

lemma *nat-diff-right-cancel-eq2*: $[\![\ m - k = n - (k{::}nat);\ k < n\ ]\!] \Longrightarrow m = n$ $\langle proof \rangle$

lemma *eq-diff-left-iff*: $[\![\ (m{::}nat) \le k;\ n \le k]\!] \Longrightarrow (k - m = k - n) = (m = n)$ $\langle proof \rangle$

lemma *eq-imp-diff-eq*: $m = (n{::}nat) \Longrightarrow m - k = n - k$ $\langle proof \rangle$

List of definitions and lemmas

thm
Nat.add-Suc-right
add-1-Suc-conv
sub-Suc0-sub-Suc-conv

thm
Nat.mult-cancel1
Nat.mult-cancel2
mult-cancel1-gr0
mult-cancel2-gr0

thm
Nat.add-lessD1
add-lessD2

thm
Nat.zero-less-diff
ord-zero-less-diff
ord-zero-le-diff

thm
le-add-diff
add-diff-le
less-add-diff
add-diff-less

thm
Nat.le-diff-conv le-diff-conv2
Nat.less-diff-conv
diff-less-imp-less-add
diff-less-conv

thm
le-diff-swap
diff-less-imp-swap
diff-less-swap

thm
less-diff-imp-less
le-diff-imp-le

thm
less-diff-le-imp-less
le-diff-le-imp-le

thm
Nat.less-imp-diff-less
le-imp-diff-le

thm
Nat.less-Suc-eq-le
less-eq-le-pred
less-imp-le-pred
le-pred-imp-less

thm
Nat.Suc-le-eq
pred-less-eq-le
pred-less-imp-le
le-imp-pred-less

thm
diff-cancel-imp-eq
thm
diff-add-inverse-Suc
thm
Nat.nat-add-left-cancel-less
Nat.nat-add-left-cancel-le
add-right-cancel
add-left-cancel
Nat.eq-diff-iff
Nat.less-diff-iff
Nat.le-diff-iff
thm
nat-diff-left-cancel-less
nat-diff-right-cancel-less
thm
nat-diff-left-cancel-le1
nat-diff-left-cancel-le2
nat-diff-right-cancel-le1
nat-diff-right-cancel-le2
thm
nat-diff-left-cancel-eq1
nat-diff-left-cancel-eq2
nat-diff-right-cancel-eq1
nat-diff-right-cancel-eq2

thm
Nat.eq-diff-iff

eq-diff-left-iff

thm
add-right-cancel add-left-cancel
Nat.diff-le-mono
eq-imp-diff-eq

end

5 Results for division and modulo operators on integers

theory *Util-Div*
imports *Util-Nat*
begin

5.1 Additional (in-)equalities with *div* and *mod*

corollary *Suc-mod-le-divisor*: $0 < m \Longrightarrow Suc\ (n\ mod\ m) \le m$
⟨*proof*⟩

lemma *mod-less-dividend*: $[\![\ 0 < m;\ m \le n\ ]\!] \Longrightarrow n\ mod\ m < (n{::}nat)$
⟨*proof*⟩

lemmas *mod-le-dividend* = *mod-less-eq-dividend*

lemma *diff-mod-le*: $(t - r)\ mod\ m \le (t{::}nat)$
⟨*proof*⟩

lemmas *div-mult-cancel* = *minus-mod-eq-div-mult* [*symmetric*]

lemma *mod-0-div-mult-cancel*: $(n\ mod\ (m{::}nat) = 0) = (n\ div\ m * m = n)$
⟨*proof*⟩

lemma *div-mult-le*: $(n{::}nat)\ div\ m * m \le n$
⟨*proof*⟩
lemma *less-div-Suc-mult*: $0 < (m{::}nat) \Longrightarrow n < Suc\ (n\ div\ m) * m$
⟨*proof*⟩

lemma *nat-ge2-conv*: $((2{::}nat) \le n) = (n \ne 0 \land n \ne 1)$
⟨*proof*⟩

lemma *Suc0-mod*: $m \ne Suc\ 0 \Longrightarrow Suc\ 0\ mod\ m = Suc\ 0$
⟨*proof*⟩

corollary *Suc0-mod-subst*:
$\llbracket\ m \neq Suc\ 0;\ P\ (Suc\ 0)\ \rrbracket \Longrightarrow P\ (Suc\ 0\ mod\ m)$
$\langle proof \rangle$
corollary *Suc0-mod-cong*:
$m \neq Suc\ 0 \Longrightarrow f\ (Suc\ 0\ mod\ m) = f\ (Suc\ 0)$
$\langle proof \rangle$

5.2 Additional results for addition and subtraction with *mod*

lemma *mod-Suc-conv*:
$((Suc\ a)\ mod\ m = (Suc\ b)\ mod\ m) = (a\ mod\ m = b\ mod\ m)$
$\langle proof \rangle$

lemma *mod-Suc'*:
$0 < n \Longrightarrow Suc\ m\ mod\ n = (if\ m\ mod\ n < n - Suc\ 0\ then\ Suc\ (m\ mod\ n)\ else\ 0)$
$\langle proof \rangle$

lemma *mod-add*:
$((a + k)\ mod\ m = (b + k)\ mod\ m) =$
$((a::nat)\ mod\ m = b\ mod\ m)$
$\langle proof \rangle$

corollary *mod-sub-add*:
$k \leq (a::nat) \Longrightarrow$
$((a - k)\ mod\ m = b\ mod\ m) = (a\ mod\ m = (b + k)\ mod\ m)$
$\langle proof \rangle$

lemma *mod-sub-eq-mod-0-conv*:
$a + b \leq (n::nat) \Longrightarrow$
$((n - a)\ mod\ m = b\ mod\ m) = ((n - (a + b))\ mod\ m = 0)$
$\langle proof \rangle$
lemma *mod-sub-eq-mod-swap*:
$\llbracket\ a \leq (n::nat);\ b \leq n\ \rrbracket \Longrightarrow$
$((n - a)\ mod\ m = b\ mod\ m) = ((n - b)\ mod\ m = a\ mod\ m)$
$\langle proof \rangle$

lemma *le-mod-greater-imp-div-less*:
$\llbracket\ a \leq (b::nat);\ a\ mod\ m > b\ mod\ m\ \rrbracket \Longrightarrow a\ div\ m < b\ div\ m$
$\langle proof \rangle$

lemma *less-mod-ge-imp-div-less*: $\llbracket\ a < (b::nat);\ a\ mod\ m \geq b\ mod\ m\ \rrbracket \Longrightarrow a\ div\ m < b\ div\ m$
$\langle proof \rangle$
corollary *less-mod-0-imp-div-less*: $\llbracket\ a < (b::nat);\ b\ mod\ m = 0\ \rrbracket \Longrightarrow a\ div\ m < b\ div\ m$
$\langle proof \rangle$

lemma *mod-diff-right-eq*:
$(a{::}nat) \le b \Longrightarrow (b - a)\ mod\ m = (b - a\ mod\ m)\ mod\ m$
⟨*proof*⟩
corollary *mod-eq-imp-diff-mod-eq*:
$[\![\ x\ mod\ m = y\ mod\ m;\ x \le (t{::}nat);\ y \le t\ ]\!] \Longrightarrow$
$(t - x)\ mod\ m = (t - y)\ mod\ m$
⟨*proof*⟩
lemma *mod-eq-imp-diff-mod-eq2*:
$[\![\ x\ mod\ m = y\ mod\ m;\ (t{::}nat) \le x;\ t \le y\ ]\!] \Longrightarrow$
$(x - t)\ mod\ m = (y - t)\ mod\ m$
⟨*proof*⟩

lemma *divisor-add-diff-mod-if*:
$(m + b\ mod\ m - a\ mod\ m)\ mod\ (m{::}nat)= ($
$if\ a\ mod\ m \le b\ mod\ m$
$then\ (b\ mod\ m - a\ mod\ m)$
$else\ (m + b\ mod\ m - a\ mod\ m))$
⟨*proof*⟩
corollary *divisor-add-diff-mod-eq1*:
$a\ mod\ m \le b\ mod\ m \Longrightarrow$
$(m + b\ mod\ m - a\ mod\ m)\ mod\ (m{::}nat) = b\ mod\ m - a\ mod\ m$
⟨*proof*⟩
corollary *divisor-add-diff-mod-eq2*:
$b\ mod\ m < a\ mod\ m \Longrightarrow$
$(m + b\ mod\ m - a\ mod\ m)\ mod\ (m{::}nat) = m + b\ mod\ m - a\ mod\ m$
⟨*proof*⟩

lemma *mod-add-mod-if*:
$(a\ mod\ m + b\ mod\ m)\ mod\ (m{::}nat)= ($
$if\ a\ mod\ m + b\ mod\ m < m$
$then\ a\ mod\ m + b\ mod\ m$
$else\ a\ mod\ m + b\ mod\ m - m)$
⟨*proof*⟩
corollary *mod-add-mod-eq1*:
$a\ mod\ m + b\ mod\ m < m \Longrightarrow$
$(a\ mod\ m + b\ mod\ m)\ mod\ (m{::}nat) = a\ mod\ m + b\ mod\ m$
⟨*proof*⟩
corollary *mod-add-mod-eq2*:
$m \le a\ mod\ m + b\ mod\ m \Longrightarrow$
$(a\ mod\ m + b\ mod\ m)\ mod\ (m{::}nat) = a\ mod\ m + b\ mod\ m - m$
⟨*proof*⟩

lemma *mod-add1-eq-if*:
$(a + b)\ mod\ (m{::}nat) = ($
$if\ (a\ mod\ m + b\ mod\ m < m)\ then\ a\ mod\ m + b\ mod\ m$
$else\ a\ mod\ m + b\ mod\ m - m)$
⟨*proof*⟩

lemma *mod-add-eq-mod-conv*: $0 < (m{::}nat) \Longrightarrow$

$((x + a) \; mod \; m = b \; mod \; m\ ) =$
$(x \; mod \; m = (m + b \; mod \; m - a \; mod \; m) \; mod \; m)$
$\langle proof \rangle$

lemma *mod-diff1-eq*:
$(a::nat) \le b \Longrightarrow (b - a) \; mod \; m = (m + b \; mod \; m - a \; mod \; m) \; mod \; m$
$\langle proof \rangle$
corollary *mod-diff1-eq-if*:
$(a::nat) \le b \Longrightarrow (b - a) \; mod \; m = ($
$if \; a \; mod \; m \le b \; mod \; m \; then \; b \; mod \; m - a \; mod \; m$
$else \; m + b \; mod \; m - a \; mod \; m)$
$\langle proof \rangle$
corollary *mod-diff1-eq1*:
$[\![\ (a::nat) \le b;\ a \; mod \; m \le b \; mod \; m \ ]\!]$
$\Longrightarrow (b - a) \; mod \; m = b \; mod \; m - a \; mod \; m$
$\langle proof \rangle$
corollary *mod-diff1-eq2*:
$[\![\ (a::nat) \le b;\ b \; mod \; m < a \; mod \; m]\!]$
$\Longrightarrow (b - a) \; mod \; m = m + b \; mod \; m - a \; mod \; m$
$\langle proof \rangle$

5.2.1 Divisor subtraction with *div* and *mod*

lemma *mod-diff-self1*:
$0 < (n::nat) \Longrightarrow (m - n) \; mod \; m = m - n$
$\langle proof \rangle$
lemma *mod-diff-self2*:
$m \le (n::nat) \Longrightarrow (n - m) \; mod \; m = n \; mod \; m$
$\langle proof \rangle$
lemma *mod-diff-mult-self1*:
$k * m \le (n::nat) \Longrightarrow (n - k * m) \; mod \; m = n \; mod \; m$
$\langle proof \rangle$
lemma *mod-diff-mult-self2*:
$m * k \le (n::nat) \Longrightarrow (n - m * k) \; mod \; m = n \; mod \; m$
$\langle proof \rangle$

lemma *div-diff-self1*: $0 < (n::nat) \Longrightarrow (m - n) \; div \; m = 0$
$\langle proof \rangle$
lemma *div-diff-self2*: $(n - m) \; div \; m = n \; div \; m - Suc \; 0$
$\langle proof \rangle$

lemma *div-diff-mult-self1*:
$(n - k * m) \; div \; m = n \; div \; m - (k::nat)$
$\langle proof \rangle$
lemma *div-diff-mult-self2*:
$(n - m * k) \; div \; m = n \; div \; m - (k::nat)$

$\langle proof \rangle$

5.2.2 Modulo equality and modulo of difference

lemma *mod-eq-imp-diff-mod-0*:
$(a::nat)\ mod\ m = b\ mod\ m \Longrightarrow (b - a)\ mod\ m = 0$
(**is** $?P \Longrightarrow ?Q$)
$\langle proof \rangle$
corollary *mod-eq-imp-diff-dvd*:
$(a::nat)\ mod\ m = b\ mod\ m \Longrightarrow m\ dvd\ b - a$
$\langle proof \rangle$

lemma *mod-neq-imp-diff-mod-neq0*:
$[\![(a::nat)\ mod\ m \neq b\ mod\ m;\ a \leq b]\!] \Longrightarrow 0 < (b - a)\ mod\ m$
$\langle proof \rangle$
corollary *mod-neq-imp-diff-not-dvd*:
$[\![(a::nat)\ mod\ m \neq b\ mod\ m;\ a \leq b]\!] \Longrightarrow \neg\ m\ dvd\ b - a$
$\langle proof \rangle$

lemma *diff-mod-0-imp-mod-eq*:
$[\![(b - a)\ mod\ m = 0;\ a \leq b]\!] \Longrightarrow (a::nat)\ mod\ m = b\ mod\ m$
$\langle proof \rangle$
corollary *diff-dvd-imp-mod-eq*:
$[\![m\ dvd\ b - a;\ a \leq b]\!] \Longrightarrow (a::nat)\ mod\ m = b\ mod\ m$
$\langle proof \rangle$

lemma *mod-eq-diff-mod-0-conv*:
$a \leq (b::nat) \Longrightarrow (a\ mod\ m = b\ mod\ m) = ((b - a)\ mod\ m = 0)$
$\langle proof \rangle$
corollary *mod-eq-diff-dvd-conv*:
$a \leq (b::nat) \Longrightarrow (a\ mod\ m = b\ mod\ m) = (m\ dvd\ b - a)$
$\langle proof \rangle$

5.3 Some additional lemmata about integer *div* and *mod*

lemma *zmod-eq-imp-diff-mod-0*:
$a\ mod\ m = b\ mod\ m \Longrightarrow (b - a)\ mod\ m = 0$ **for** $a\ b\ m :: int$
$\langle proof \rangle$

lemmas *int-mod-distrib* = *zmod-int*

lemma *zdiff-mod-0-imp-mod-eq--pos*:
$[\![(b - a)\ mod\ m = 0;\ 0 < (m::int)]\!] \Longrightarrow a\ mod\ m = b\ mod\ m$
(**is** $[\![?P;\ ?Pm]\!] \Longrightarrow ?Q$)
$\langle proof \rangle$

lemma *zmod-zminus-eq-conv-pos*:

$0 < (m::int) \Longrightarrow (a\ mod\ -\ m = b\ mod\ -\ m) = (a\ mod\ m = b\ mod\ m)$
⟨proof⟩
lemma *zmod-zminus-eq-conv*:
$((a::int)\ mod\ -\ m = b\ mod\ -\ m) = (a\ mod\ m = b\ mod\ m)$
⟨proof⟩

lemma *zdiff-mod-0-imp-mod-eq*:
$(b - a)\ mod\ m = 0 \Longrightarrow (a::int)\ mod\ m = b\ mod\ m$
⟨proof⟩

lemma *zmod-eq-diff-mod-0-conv*:
$((a::int)\ mod\ m = b\ mod\ m) = ((b - a)\ mod\ m = 0)$
⟨proof⟩

lemma $\neg(\exists (a::int)\ b\ m.\ (b - a)\ mod\ m = 0 \wedge a\ mod\ m \neq b\ mod\ m)$
⟨proof⟩
lemma $\exists (a::nat)\ b\ m.\ (b - a)\ mod\ m = 0 \wedge a\ mod\ m \neq b\ mod\ m$
⟨proof⟩

lemma *zmult-div-leq-mono*:
$[\![(0::int) \leq x;\ a \leq b;\ 0 < d]\!] \Longrightarrow x * a\ div\ d \leq x * b\ div\ d$
⟨proof⟩

lemma *zmult-div-leq-mono-neg*:
$[\![x \leq (0::int);\ a \leq b;\ 0 < d]\!] \Longrightarrow x * b\ div\ d \leq x * a\ div\ d$
⟨proof⟩

lemma *zmult-div-pos-le*:
$[\![(0::int) \leq a;\ 0 \leq b;\ b \leq c]\!] \Longrightarrow a * b\ div\ c \leq a$
⟨proof⟩

lemma *zmult-div-neg-le*:
$[\![a \leq (0::int);\ 0 < c;\ c \leq b]\!] \Longrightarrow a * b\ div\ c \leq a$
⟨proof⟩

lemma *zmult-div-ge-0*:$[\![(0::int) \leq x;\ 0 \leq a;\ 0 < c]\!] \Longrightarrow 0 \leq a * x\ div\ c$
⟨proof⟩

corollary *zmult-div-plus-ge-0*:
$[\![(0::int) \leq x;\ 0 \leq a;\ 0 \leq b;\ 0 < c]\!] \Longrightarrow 0 \leq a * x\ div\ c + b$
⟨proof⟩

lemma *zmult-div-abs-ge*:
$[\![(0::int) \leq b;\ b \leq b';\ 0 \leq a;\ 0 < c]\!] \Longrightarrow$
$|a * b\ div\ c| \leq |a * b'\ div\ c|$
⟨proof⟩

lemma *zmult-div-plus-abs-ge*:
$[\![(0::int) \leq b;\ b \leq b';\ 0 \leq a;\ 0 < c]\!] \Longrightarrow$
$|a * b\ div\ c + a| \leq |a * b'\ div\ c + a|$
⟨*proof*⟩

5.4 Some further (in-)equality results for *div* and *mod*

lemma *less-mod-eq-imp-add-divisor-le*:
$[\![(x::nat) < y;\ x\ mod\ m = y\ mod\ m]\!] \Longrightarrow x + m \leq y$
⟨*proof*⟩

lemma *less-div-imp-mult-add-divisor-le*:
$(x::nat) < n\ div\ m \Longrightarrow x * m + m \leq n$
⟨*proof*⟩

lemma *mod-add-eq-imp-mod-0*:
$((n + k)\ mod\ (m::nat) = n\ mod\ m) = (k\ mod\ m = 0)$
⟨*proof*⟩

lemma *between-imp-mod-between*:
$[\![b < (m::nat);\ m * k + a \leq n;\ n \leq m * k + b]\!] \Longrightarrow$
$a \leq n\ mod\ m \wedge n\ mod\ m \leq b$
⟨*proof*⟩

corollary *between-imp-mod-le*:
$[\![b < (m::nat);\ m * k \leq n;\ n \leq m * k + b]\!] \Longrightarrow n\ mod\ m \leq b$
⟨*proof*⟩
corollary *between-imp-mod-gr0*:
$[\![(m::nat) * k < n;\ n < m * k + m]\!] \Longrightarrow 0 < n\ mod\ m$
⟨*proof*⟩

corollary *le-less-div-conv*:
$0 < m \Longrightarrow (k * m \leq n \wedge n < Suc\ k * m) = (n\ div\ m = k)$
⟨*proof*⟩

lemma *le-less-imp-div*:
$[\![k * m \leq n;\ n < Suc\ k * m]\!] \Longrightarrow n\ div\ m = k$
⟨*proof*⟩

lemma *div-imp-le-less*:
$[\![n\ div\ m = k;\ 0 < m]\!] \Longrightarrow k * m \leq n \wedge n < Suc\ k * m$
⟨*proof*⟩

lemma *div-le-mod-le-imp-le*:
$[\![(a::nat)\ div\ m \leq b\ div\ m;\ a\ mod\ m \leq b\ mod\ m]\!] \Longrightarrow a \leq b$
⟨*proof*⟩

lemma *le-mod-add-eq-imp-add-mod-le*:
$[\![\ a \leq b;\ (a + k)\ mod\ m = (b{::}nat)\ mod\ m\ ]\!] \Longrightarrow a + k\ mod\ m \leq b$
$\langle proof \rangle$

corollary *mult-divisor-le-mod-ge-imp-ge*:
$[\![\ (m{::}nat) * k \leq n;\ r \leq n\ mod\ m\ ]\!] \Longrightarrow m * k + r \leq n$
$\langle proof \rangle$

5.5 Additional multiplication results for *mod* and *div*

lemma *mod-0-imp-mod-mult-right-0*:
$n\ mod\ m = (0{::}nat) \Longrightarrow n * k\ mod\ m = 0$
$\langle proof \rangle$
lemma *mod-0-imp-mod-mult-left-0*:
$n\ mod\ m = (0{::}nat) \Longrightarrow k * n\ mod\ m = 0$
$\langle proof \rangle$

lemma *mod-0-imp-div-mult-left-eq*:
$n\ mod\ m = (0{::}nat) \Longrightarrow k * n\ div\ m = k * (n\ div\ m)$
$\langle proof \rangle$
lemma *mod-0-imp-div-mult-right-eq*:
$n\ mod\ m = (0{::}nat) \Longrightarrow n * k\ div\ m = k * (n\ div\ m)$
$\langle proof \rangle$

lemma *mod-0-imp-mod-factor-0-left*:
$n\ mod\ (m * m') = (0{::}nat) \Longrightarrow n\ mod\ m = 0$
$\langle proof \rangle$
lemma *mod-0-imp-mod-factor-0-right*:
$n\ mod\ (m * m') = (0{::}nat) \Longrightarrow n\ mod\ m' = 0$
$\langle proof \rangle$

5.6 Some factor distribution facts for *mod*

lemma *mod-eq-mult-distrib*:
$(a{::}nat)\ mod\ m = b\ mod\ m \Longrightarrow$
$a * k\ mod\ (m * k) = b * k\ mod\ (m * k)$
$\langle proof \rangle$

lemma *mod-mult-eq-imp-mod-eq*:
$(a{::}nat)\ mod\ (m * k) = b\ mod\ (m * k) \Longrightarrow a\ mod\ m = b\ mod\ m$
$\langle proof \rangle$
corollary *mod-eq-mod-0-imp-mod-eq*:
$[\![\ (a{::}nat)\ mod\ m' = b\ mod\ m';\ m'\ mod\ m = 0\ ]\!]$
$\Longrightarrow a\ mod\ m = b\ mod\ m$
$\langle proof \rangle$

lemma *mod-factor-imp-mod-0*:
$[\![(x{::}nat)\ mod\ (m * k) = y * k\ mod\ (m * k)]\!] \Longrightarrow x\ mod\ k = 0$
(**is** $[\![\ ?P1\ ]\!] \Longrightarrow ?Q$)

$\langle proof \rangle$
corollary *mod-factor-div*:
$[\![(x::nat)\ mod\ (m * k) = y * k\ mod\ (m * k)]\!] \Longrightarrow x\ div\ k * k = x$
$\langle proof \rangle$

lemma *mod-factor-div-mod*:
$[\![\ (x::nat)\ mod\ (m * k) = y * k\ mod\ (m * k);\ 0 < k\ ]\!]$
$\Longrightarrow x\ div\ k\ mod\ m = y\ mod\ m$
(**is** $[\![\ ?P1;\ ?P2\ ]\!] \Longrightarrow ?L = ?R$)
$\langle proof \rangle$

5.7 More results about quotient *div* with addition and subtraction

lemma *div-add1-eq-if*: $0 < m \Longrightarrow$
$(a + b)\ div\ (m::nat) = a\ div\ m + b\ div\ m + ($
$if\ a\ mod\ m + b\ mod\ m < m\ then\ 0\ else\ Suc\ 0)$
$\langle proof \rangle$
corollary *div-add1-eq1*:
$a\ mod\ m + b\ mod\ m < (m::nat) \Longrightarrow$
$(a + b)\ div\ (m::nat) = a\ div\ m + b\ div\ m$
$\langle proof \rangle$
corollary *div-add1-eq1-mod-0-left*:
$a\ mod\ m = 0 \Longrightarrow (a + b)\ div\ (m::nat) = a\ div\ m + b\ div\ m$
$\langle proof \rangle$
corollary *div-add1-eq1-mod-0-right*:
$b\ mod\ m = 0 \Longrightarrow (a + b)\ div\ (m::nat) = a\ div\ m + b\ div\ m$
$\langle proof \rangle$
corollary *div-add1-eq2*:
$[\![\ 0 < m;\ (m::nat) \le a\ mod\ m + b\ mod\ m\ ]\!] \Longrightarrow$
$(a + b)\ div\ (m::nat) = Suc\ (a\ div\ m + b\ div\ m)$
$\langle proof \rangle$

lemma *div-Suc*:
$0 < n \Longrightarrow Suc\ m\ div\ n = (if\ Suc\ (m\ mod\ n) = n\ then\ Suc\ (m\ div\ n)\ else\ m\ div\ n)$
$\langle proof \rangle$
lemma *div-Suc'*:
$0 < n \Longrightarrow Suc\ m\ div\ n = (if\ m\ mod\ n < n - Suc\ 0\ then\ m\ div\ n\ else\ Suc\ (m\ div\ n))$
$\langle proof \rangle$

lemma *div-diff1-eq-if*:
$(b - a)\ div\ (m::nat) =$
$b\ div\ m - a\ div\ m - (if\ a\ mod\ m \le b\ mod\ m\ then\ 0\ else\ Suc\ 0)$
$\langle proof \rangle$

corollary *div-diff1-eq*:
$(b - a)\ div\ (m::nat) =$

$b\ div\ m - a\ div\ m - (m + a\ mod\ m - Suc\ (b\ mod\ m))\ div\ m$
⟨*proof*⟩

corollary *div-diff1-eq1*:
$a\ mod\ m \le b\ mod\ m \Longrightarrow$
$(b - a)\ div\ (m::nat) = b\ div\ m - a\ div\ m$
⟨*proof*⟩
corollary *div-diff1-eq1-mod-0*:
$a\ mod\ m = 0 \Longrightarrow$
$(b - a)\ div\ (m::nat) = b\ div\ m - a\ div\ m$
⟨*proof*⟩
corollary *div-diff1-eq2*:
$b\ mod\ m < a\ mod\ m \Longrightarrow$
$(b - a)\ div\ (m::nat) = b\ div\ m - Suc\ (a\ div\ m)$
⟨*proof*⟩

5.8 Further results about *div* and *mod*

5.8.1 Some auxiliary facts about *mod*

lemma *diff-less-divisor-imp-sub-mod-eq*:
$[\![(x::nat) \le y;\ y - x < m]\!] \Longrightarrow x = y - (y - x)\ mod\ m$
⟨*proof*⟩
lemma *diff-ge-divisor-imp-sub-mod-less*:
$[\![(x::nat) \le y;\ m \le y - x;\ 0 < m]\!] \Longrightarrow x < y - (y - x)\ mod\ m$
⟨*proof*⟩

lemma *le-imp-sub-mod-le*:
$(x::nat) \le y \Longrightarrow x \le y - (y - x)\ mod\ m$
⟨*proof*⟩

lemma *mod-less-diff-mod*:
$[\![n\ mod\ m < r;\ r \le m;\ r \le (n::nat)]\!] \Longrightarrow$
$(n - r)\ mod\ m = m + n\ mod\ m - r$
⟨*proof*⟩

lemma *mod-0-imp-mod-pred*:
$[\![0 < (n::nat);\ n\ mod\ m = 0]\!] \Longrightarrow$
$(n - Suc\ 0)\ mod\ m = m - Suc\ 0$
⟨*proof*⟩

lemma *mod-pred*:
$0 < n \Longrightarrow$
$(n - Suc\ 0)\ mod\ m = ($
$\quad if\ n\ mod\ m = 0\ then\ m - Suc\ 0\ else\ n\ mod\ m - Suc\ 0)$
⟨*proof*⟩
corollary *mod-pred-Suc-mod*:
$0 < n \Longrightarrow Suc\ ((n - Suc\ 0)\ mod\ m)\ mod\ m = n\ mod\ m$
⟨*proof*⟩
corollary *diff-mod-pred*:

$a < b \Longrightarrow$
$(b - Suc\ a)\ mod\ m = ($
$if\ a\ mod\ m = b\ mod\ m\ then\ m - Suc\ 0\ else\ (b - a)\ mod\ m - Suc\ 0)$
⟨*proof*⟩
corollary *diff-mod-pred-Suc-mod*:
$a < b \Longrightarrow Suc\ ((b - Suc\ a)\ mod\ m)\ mod\ m = (b - a)\ mod\ m$
⟨*proof*⟩

lemma *mod-eq-imp-diff-mod-eq-divisor*:
$[\![\ a < b;\ 0 < m;\ a\ mod\ m = b\ mod\ m\ ]\!] \Longrightarrow$
$Suc\ ((b - Suc\ a)\ mod\ m) = m$
⟨*proof*⟩

lemma *sub-diff-mod-eq*:
$r \le t \Longrightarrow (t - (t - r)\ mod\ m)\ mod\ (m{::}nat) = r\ mod\ m$
⟨*proof*⟩

lemma *sub-diff-mod-eq′*:
$r \le t \Longrightarrow (k * m + t - (t - r)\ mod\ m)\ mod\ (m{::}nat) = r\ mod\ m$
⟨*proof*⟩

lemma *mod-eq-Suc-0-conv*: $Suc\ 0 < k \Longrightarrow ((x + k - Suc\ 0)\ mod\ k = 0) = (x\ mod\ k = Suc\ 0)$
⟨*proof*⟩

lemma *mod-eq-divisor-minus-Suc-0-conv*: $Suc\ 0 < k \Longrightarrow (x\ mod\ k = k - Suc\ 0) = (Suc\ x\ mod\ k = 0)$
⟨*proof*⟩

5.8.2 Some auxiliary facts about *div*

lemma *sub-mod-div-eq-div*: $((n{::}nat) - n\ mod\ m)\ div\ m = n\ div\ m$
⟨*proof*⟩

lemma *mod-less-imp-diff-div-conv*:
$[\![\ n\ mod\ m < r;\ r \le m + n\ mod\ m]\!] \Longrightarrow (n - r)\ div\ m = n\ div\ m - Suc\ 0$
⟨*proof*⟩

corollary *mod-0-le-imp-diff-div-conv*:
$[\![\ n\ mod\ m = 0;\ 0 < r;\ r \le m\ ]\!] \Longrightarrow (n - r)\ div\ m = n\ div\ m - Suc\ 0$
⟨*proof*⟩
corollary *mod-0-less-imp-diff-Suc-div-conv*:
$[\![\ n\ mod\ m = 0;\ r < m\ ]\!] \Longrightarrow (n - Suc\ r)\ div\ m = n\ div\ m - Suc\ 0$
⟨*proof*⟩
corollary *mod-0-imp-diff-Suc-div-conv*:
$(n - r)\ mod\ m = 0 \Longrightarrow (n - Suc\ r)\ div\ m = (n - r)\ div\ m - Suc\ 0$
⟨*proof*⟩
corollary *mod-0-imp-sub-1-div-conv*:

$n\ mod\ m = 0 \Longrightarrow (n - Suc\ 0)\ div\ m = n\ div\ m - Suc\ 0$
⟨*proof*⟩
corollary *sub-Suc-mod-div-conv*:
$(n - Suc\ (n\ mod\ m))\ div\ m = n\ div\ m - Suc\ 0$
⟨*proof*⟩

lemma *div-le-conv*: $0 < m \Longrightarrow n\ div\ m \leq k = (n \leq Suc\ k * m - Suc\ 0)$
⟨*proof*⟩

lemma *le-div-conv*: $0 < (m::nat) \Longrightarrow (n \leq k\ div\ m) = (n * m \leq k)$
⟨*proof*⟩

lemma *less-mult-imp-div-less*: $n < k * m \Longrightarrow n\ div\ m < (k::nat)$
⟨*proof*⟩

lemma *div-less-imp-less-mult*: $[\![\ 0 < (m::nat);\ n\ div\ m < k\ ]\!] \Longrightarrow n < k * m$
⟨*proof*⟩

lemma *div-less-conv*: $0 < (m::nat) \Longrightarrow (n\ div\ m < k) = (n < k * m)$
⟨*proof*⟩

lemma *div-eq-0-conv*: $(n\ div\ (m::nat) = 0) = (m = 0 \lor n < m)$
⟨*proof*⟩
lemma *div-eq-0-conv'*: $0 < m \Longrightarrow (n\ div\ (m::nat) = 0) = (n < m)$
⟨*proof*⟩
corollary *div-gr-imp-gr-divisor*: $x < n\ div\ (m::nat) \Longrightarrow m \leq n$
⟨*proof*⟩

lemma *mod-0-less-div-conv*:
$n\ mod\ (m::nat) = 0 \Longrightarrow (k * m < n) = (k < n\ div\ m)$
⟨*proof*⟩

lemma *add-le-divisor-imp-le-Suc-div*:
$[\![\ x\ div\ m \leq n;\ y \leq m\ ]\!] \Longrightarrow (x + y)\ div\ m \leq Suc\ n$
⟨*proof*⟩

List of definitions and lemmas

thm
minus-mod-eq-mult-div [*symmetric*]
mod-0-div-mult-cancel
div-mult-le
less-div-Suc-mult
thm
Suc0-mod
Suc0-mod-subst
Suc0-mod-cong

thm

mod-Suc-conv

thm
mod-add
mod-sub-add

thm
mod-sub-eq-mod-0-conv
mod-sub-eq-mod-swap

thm
le-mod-greater-imp-div-less
thm
mod-diff-right-eq
mod-eq-imp-diff-mod-eq

thm
divisor-add-diff-mod-if
divisor-add-diff-mod-eq1
divisor-add-diff-mod-eq2

thm
mod-add-eq
mod-add1-eq-if
thm
mod-diff1-eq-if
mod-diff1-eq
mod-diff1-eq1
mod-diff1-eq2

thm
nat-mod-distrib
int-mod-distrib

thm
zmod-zminus-eq-conv

thm
mod-eq-imp-diff-mod-0
zmod-eq-imp-diff-mod-0

thm
mod-neq-imp-diff-mod-neq0
diff-mod-0-imp-mod-eq
zdiff-mod-0-imp-mod-eq

thm
zmod-eq-diff-mod-0-conv
mod-eq-diff-mod-0-conv

thm
less-mod-eq-imp-add-divisor-le
thm
mod-add-eq-imp-mod-0
thm
mod-eq-mult-distrib
mod-factor-imp-mod-0
mod-factor-div
mod-factor-div-mod

thm
mod-diff-self1
mod-diff-self2
mod-diff-mult-self1
mod-diff-mult-self2

thm
div-diff-self1
div-diff-self2
div-diff-mult-self1
div-diff-mult-self2

thm
le-less-imp-div
div-imp-le-less
thm
le-less-div-conv

thm
diff-less-divisor-imp-sub-mod-eq
diff-ge-divisor-imp-sub-mod-less
le-imp-sub-mod-le

thm
sub-mod-div-eq-div

thm
mod-less-imp-diff-div-conv
mod-0-le-imp-diff-div-conv
mod-0-less-imp-diff-Suc-div-conv
mod-0-imp-sub-1-div-conv

thm
sub-Suc-mod-div-conv

thm

mod-less-diff-mod
mod-0-imp-mod-pred

thm
mod-pred
mod-pred-Suc-mod

thm
mod-eq-imp-diff-mod-eq-divisor

thm
diff-mod-le
sub-diff-mod-eq
sub-diff-mod-eq'

thm
div-diff1-eq-if
div-diff1-eq
div-diff1-eq1
div-diff1-eq2

thm
div-le-conv

end

6 Sets of natural numbers

theory *SetInterval2*
imports
HOL−Library.Infinite-Set
Util-Set
../CommonArith/Util-MinMax
../CommonArith/Util-NatInf
../CommonArith/Util-Div
begin

6.1 Auxiliary results for monotonic, injective and surjective functions over sets

6.1.1 Monotonicity

definition *mono-on* :: $('a::order \Rightarrow 'b::order) \Rightarrow 'a\ set \Rightarrow bool$
where *mono-on* $f\ A \equiv \forall\, a{\in}A.\ \forall\, b{\in}A.\ a \le b \longrightarrow f\ a \le f\ b$

definition *strict-mono-on* :: $('a::order \Rightarrow 'b::order) \Rightarrow 'a\ set \Rightarrow bool$
where *strict-mono-on* $f\ A \equiv \forall\, a{\in}A.\ \forall\, b{\in}A.\ a < b \longrightarrow f\ a < f\ b$

lemma *mono-on-subset*: $[\![$ *mono-on f A* ; $B \subseteq A$ $]\!] \Longrightarrow$ *mono-on f B*
⟨*proof*⟩

lemma *strict-mono-on-subset*: $[\![$ *strict-mono-on f A* ; $B \subseteq A$ $]\!] \Longrightarrow$ *strict-mono-on f B*
⟨*proof*⟩

lemma *mono-on-UNIV-mono-conv*: *mono-on f UNIV* = *mono f*
⟨*proof*⟩
lemma *strict-mono-on-UNIV-strict-mono-conv*:
strict-mono-on f UNIV = *strict-mono f*
⟨*proof*⟩

lemma *mono-imp-mono-on*: *mono f* $\Longrightarrow$ *mono-on f A*
⟨*proof*⟩
lemma *strict-mono-imp-strict-mono-on*: *strict-mono f* $\Longrightarrow$ *strict-mono-on f A*
⟨*proof*⟩

lemma *strict-mono-on-imp-mono-on*: *strict-mono-on f A* $\Longrightarrow$ *mono-on f A*
⟨*proof*⟩

6.1.2 Injectivity

lemma *inj-imp-inj-on*: *inj f* $\Longrightarrow$ *inj-on f A*
⟨*proof*⟩

lemma *strict-mono-on-imp-inj-on*:
strict-mono-on f (*A*::'*a*::*linorder set*) $\Longrightarrow$ *inj-on f A*
⟨*proof*⟩

lemma *strict-mono-imp-inj*: *strict-mono* (*f*::('*a*::*linorder* ⇒ '*b*::*order*)) $\Longrightarrow$ *inj f*
⟨*proof*⟩

lemma *strict-mono-on-mono-on-conv*:
strict-mono-on f (*A*::'*a*::*linorder set*) = (*mono-on f A* ∧ *inj-on f A*)
⟨*proof*⟩

corollary *strict-mono-mono-conv*:
strict-mono (*f*::('*a*::*linorder* ⇒ '*b*::*order*)) = (*mono f* ∧ *inj f*)
⟨*proof*⟩

lemma *inj-on-image-mem-iff*:
$[\![$ *inj-on f A*; $B \subseteq A$; $a \in A$ $]\!] \Longrightarrow$ ($f\ a \in f$ ‘ B) = ($a \in B$)
⟨*proof*⟩

corollary *inj-on-union-image-Int*:
  $inj\text{-}on\ f\ (A \cup B) \Longrightarrow f\ `\ (A \cap B) = f\ `\ A \cap f\ `\ B$
$\langle proof \rangle$

6.1.3 Surjectivity

definition *surj-on* :: $('a \Rightarrow 'b) \Rightarrow 'a\ set \Rightarrow 'b\ set \Rightarrow bool$
  where $surj\text{-}on\ f\ A\ B \equiv \forall\, b{\in}B.\ \exists\, a{\in}A.\ b = f\ a$

lemma *surj-on-conv*: $(surj\text{-}on\ f\ A\ B) = (\forall\, b{\in}B.\ \exists\, a{\in}A.\ b = f\ a)$
$\langle proof \rangle$

lemma *surj-on-image-conv*: $(surj\text{-}on\ f\ A\ B) = (B \subseteq f\ `\ A)$
$\langle proof \rangle$

lemma *surj-on-id*: $surj\text{-}on\ id\ A\ A$
$\langle proof \rangle$

lemma
  surj-onI: $[\![\ \forall\, b \in B.\ \exists\, a{\in}A.\ b = f\ a\ ]\!] \Longrightarrow surj\text{-}on\ f\ A\ B$ **and**
  surj-onD2: $surj\text{-}on\ f\ A\ B \Longrightarrow \forall\, b \in B.\ \exists\, a{\in}A.\ b = f\ a$ **and**
  surj-onD: $[\![\ surj\text{-}on\ f\ A\ B;\ b \in B\ ]\!] \Longrightarrow \exists\, a{\in}A.\ b = f\ a$
$\langle proof \rangle$

lemma *comp-surj-on*:
  $[\![\ surj\text{-}on\ f\ A\ B;\ surj\text{-}on\ g\ B\ C\ ]\!] \Longrightarrow surj\text{-}on\ (g \circ f)\ A\ C$
$\langle proof \rangle$

lemma *surj-on-Un-right*: $surj\text{-}on\ f\ A\ (B1 \cup B2) = (surj\text{-}on\ f\ A\ B1 \wedge surj\text{-}on\ f\ A\ B2)$
$\langle proof \rangle$

lemma *surj-on-Un-left*:
  $surj\text{-}on\ f\ (A1 \cup A2)\ B =$
  $(\exists\, B1.\ \exists\, B2.\ B \subseteq B1 \cup B2 \wedge surj\text{-}on\ f\ A1\ B1 \wedge surj\text{-}on\ f\ A2\ B2)$
$\langle proof \rangle$

lemma *surj-on-diff-right*: $surj\text{-}on\ f\ A\ B \Longrightarrow surj\text{-}on\ f\ A\ (B - B')$
$\langle proof \rangle$

lemma *surj-on-empty-right*: $surj\text{-}on\ f\ A\ \{\}$
$\langle proof \rangle$

lemma *surj-on-empty-left*: $surj\text{-}on\ f\ \{\}\ B = (B = \{\})$
$\langle proof \rangle$

lemma *surj-on-imageI*: $surj\text{-}on\ (g \circ f)\ A\ B \Longrightarrow surj\text{-}on\ g\ (f\ `\ A)\ B$
$\langle proof \rangle$

lemma *surj-on-insert-right*: $surj\text{-}on\ f\ A\ (insert\ b\ B) = (surj\text{-}on\ f\ A\ B \wedge surj\text{-}on\ f\ A\ \{b\})$
$\langle proof \rangle$

lemma *surj-on-insert-left*: $surj\text{-}on\ f\ (insert\ a\ A)\ B = (surj\text{-}on\ f\ A\ (B - \{f\ a\}))$
$\langle proof \rangle$

lemma *surj-on-subset-right*: $[\![\ surj\text{-}on\ f\ A\ B;\ B' \subseteq B\ ]\!] \Longrightarrow surj\text{-}on\ f\ A\ B'$
$\langle proof \rangle$

lemma *surj-on-subset-left*: $[\![\ surj\text{-}on\ f\ A\ B;\ A \subseteq A'\ ]\!] \Longrightarrow surj\text{-}on\ f\ A'\ B$
$\langle proof \rangle$

lemma *bij-betw-imp-surj-on*: $bij\text{-}betw\ f\ A\ B \Longrightarrow surj\text{-}on\ f\ A\ B$
$\langle proof \rangle$

lemma *bij-betw-inj-on-surj-on-conv*:
$bij\text{-}betw\ f\ A\ B = (inj\text{-}on\ f\ A \wedge surj\text{-}on\ f\ A\ B \wedge f\ `\ A \subseteq B)$
$\langle proof \rangle$

6.1.4 Induction over natural sets

lemma *image-nat-induct*:
$[\![\ P\ (f\ 0);\ \bigwedge n.\ P\ (f\ n) \Longrightarrow P\ (f\ (Suc\ n));\ surj\text{-}on\ f\ UNIV\ I;\ a \in I\ ]\!] \Longrightarrow P\ a$
$\langle proof \rangle$

lemma *nat-induct'*[*rule-format*]:
$[\![\ P\ n0;\ \bigwedge n.\ [\![\ n0 \le n;\ \ P\ n\ ]\!] \Longrightarrow P\ (Suc\ n);\ n0 \le n\ ]\!] \Longrightarrow P\ n$
$\langle proof \rangle$

lemma *enat-induct*:
$[\![\ P\ 0;\ P\ \infty;\ \bigwedge n.\ P\ n \Longrightarrow P\ (eSuc\ n)]\!] \Longrightarrow P\ n$
$\langle proof \rangle$

lemma *eSuc-imp-Suc-aux-0*:
$[\![\ \bigwedge n.\ P\ n \Longrightarrow P\ (eSuc\ n);\ n0' \le n';\ P\ (enat\ n')]\!] \Longrightarrow P\ (enat\ (Suc\ n'))$
$\langle proof \rangle$

lemma *eSuc-imp-Suc-aux-n0*:
$[\![\ \bigwedge n.\ [\![enat\ n0' \le n;\ P\ n]\!] \Longrightarrow P\ (eSuc\ n);\ n0' \le n';\ P\ (enat\ n')]\!] \Longrightarrow P\ (enat\ (Suc\ n'))$
$\langle proof \rangle$

lemma *enat-induct'*:
$[\![\ P\ (n0{::}enat);\ P\ \infty;\ \bigwedge n.\ [\![\ n0 \le n;\ \ P\ n\ ]\!] \Longrightarrow P\ (eSuc\ n);\ n0 \le n\ ]\!] \Longrightarrow P\ n$
$\langle proof \rangle$

lemma *wf-less-interval*:
$wf\ \{\ (x,y).\ x \in (I::nat\ set) \wedge y \in I \wedge x < y\ \}$
$\langle proof \rangle$

lemma *interval-induct*:
$[\![\ \bigwedge x.\ \forall y.\ (x{\in}(I::nat\ set) \wedge y \in I \wedge y < x \longrightarrow P\ y) \Longrightarrow P\ x\ ]\!]$
$\Longrightarrow P\ a$
(**is** $[\![\ \bigwedge x.\ \forall y.\ ?IA\ x\ y \Longrightarrow P\ x\ ]\!] \Longrightarrow P\ a$)
$\langle proof \rangle$

corollary *interval-induct-rule*:
$[\![\ \bigwedge x.\ (\bigwedge y.\ (x{\in}(I::nat\ set) \wedge y \in I \wedge y < x \Longrightarrow P\ y)) \Longrightarrow P\ x\ ]\!]$
$\Longrightarrow P\ a$
$\langle proof \rangle$

6.1.5 Monotonicity and injectivity of artithmetic operators

lemma *add-left-inj*: $inj\ (\lambda x.\ n + (x::'a::cancel\text{-}ab\text{-}semigroup\text{-}add))$
$\langle proof \rangle$

lemma *add-right-inj*: $inj\ (\lambda x.\ x + (n::'a::cancel\text{-}ab\text{-}semigroup\text{-}add))$
$\langle proof \rangle$

lemma *mult-left-inj*: $0 < n \Longrightarrow inj\ (\lambda x.\ (n::nat) * x)$
$\langle proof \rangle$

lemma *mult-right-inj*: $0 < n \Longrightarrow inj\ (\lambda x.\ x * (n::nat))$
$\langle proof \rangle$

lemma *sub-left-inj-on*: $inj\text{-}on\ (\lambda x.\ (x::nat) - k)\ \{k..\}$
$\langle proof \rangle$

lemma *sub-right-inj-on*: $inj\text{-}on\ (\lambda x.\ k - (x::nat))\ \{..k\}$
$\langle proof \rangle$

lemma *add-left-strict-mono*: $strict\text{-}mono\ (\lambda x.\ n + (x::'a::ordered\text{-}cancel\text{-}ab\text{-}semigroup\text{-}add))$
$\langle proof \rangle$

lemma *add-right-strict-mono*: $strict\text{-}mono\ (\lambda x.\ x + (n::'a::ordered\text{-}cancel\text{-}ab\text{-}semigroup\text{-}add))$
$\langle proof \rangle$

lemma *mult-left-strict-mono*: $0 < n \Longrightarrow strict\text{-}mono\ (\lambda x.\ n * (x::nat))$
$\langle proof \rangle$

lemma *mult-right-strict-mono*: $0 < n \Longrightarrow strict\text{-}mono\ (\lambda x.\ x * (n::nat))$
$\langle proof \rangle$

lemma *sub-left-strict-mono-on*: $strict\text{-}mono\text{-}on\ (\lambda x.\ (x::nat) - k)\ \{k..\}$
$\langle proof \rangle$

lemma *div-right-strict-mono-on*:
$[\![\ 0 < (k{::}nat);\ \forall x{\in}I.\ \forall y{\in}I.\ x < y \longrightarrow x + k \le y \ ]\!] \Longrightarrow$
$strict\text{-}mono\text{-}on\ (\lambda x.\ x\ div\ k)\ I$
⟨*proof*⟩

lemma *mod-eq-div-right-strict-mono-on*:
$[\![\ 0 < (k{::}nat);\ \forall x{\in}I.\ \forall y{\in}I.\ x\ mod\ k = y\ mod\ k \ ]\!] \Longrightarrow$
$strict\text{-}mono\text{-}on\ (\lambda x.\ x\ div\ k)\ I$
⟨*proof*⟩

corollary *div-right-inj-on*:
$[\![\ 0 < (k{::}nat);\ \forall x{\in}I.\ \forall y{\in}I.\ x < y \longrightarrow x + k \le y \ ]\!] \Longrightarrow$
$inj\text{-}on\ (\lambda x.\ x\ div\ k)\ I$
⟨*proof*⟩

corollary *mod-eq-imp-div-right-inj-on*:
$[\![\ 0 < (k{::}nat);\ \forall x{\in}I.\ \forall y{\in}I.\ x\ mod\ k = y\ mod\ k \ ]\!] \Longrightarrow$
$inj\text{-}on\ (\lambda x.\ x\ div\ k)\ I$
⟨*proof*⟩

6.2 *Min* and *Max* elements of a set

A special minimum operator is required for dealing with infinite wellordered sets because the standard operator *Min* is usable only with finite sets.

definition $iMin :: 'a{::}wellorder\ set \Rightarrow 'a$
where $iMin\ I \equiv LEAST\ x.\ x \in I$

6.2.1 Basic results, as for *Least*

lemma *iMinI*: $k \in I \Longrightarrow iMin\ I \in I$
⟨*proof*⟩

lemma *iMinI-ex*: $\exists x.\ x \in I \Longrightarrow iMin\ I \in I$
⟨*proof*⟩

corollary *iMinI-ex2*: $I \ne \{\} \Longrightarrow iMin\ I \in I$
⟨*proof*⟩

lemma *iMinI2*: $[\![\ k \in I;\ \bigwedge x.\ x \in I \Longrightarrow P\ x \ ]\!] \Longrightarrow P\ (iMin\ I)$
⟨*proof*⟩

lemma *iMinI2-ex*: $[\![\ \exists x.\ x \in I;\ \bigwedge x.\ x \in I \Longrightarrow P\ x \ ]\!] \Longrightarrow P\ (iMin\ I)$
⟨*proof*⟩

lemma *iMinI2-ex2*: $[\![\ I \ne \{\}\ ;\ \bigwedge x.\ x \in I \Longrightarrow P\ x \ ]\!] \Longrightarrow P\ (iMin\ I)$
⟨*proof*⟩

lemma *iMin-le*[*dest*]: $k \in I \Longrightarrow iMin\ I \le k$

$\langle proof \rangle$

lemma *iMin-neq-imp-greater*[*dest*]: $[\![k \in I;\ k \neq iMin\ I]\!] \Longrightarrow iMin\ I < k$
$\langle proof \rangle$

lemma *iMin-mono*:
$[\![mono\ f;\ \exists\, x.\ x \in I]\!] \Longrightarrow iMin\ (f\ `\ I) = f\ (iMin\ I)$
$\langle proof \rangle$

corollary *iMin-mono2*:
$[\![mono\ f;\ I \neq \{\}]\!] \Longrightarrow iMin\ (f\ `\ I) = f\ (iMin\ I)$
$\langle proof \rangle$

lemma *not-less-iMin*: $k < iMin\ I \Longrightarrow k \notin I$
$\langle proof \rangle$

lemma *Collect-not-less-iMin*: $k < iMin\ \{x.\ P\ x\} \Longrightarrow \neg\ P\ k$
$\langle proof \rangle$

lemma *Collect-iMin-le*: $P\ k \Longrightarrow iMin\ \{x.\ P\ x\} \leq k$
$\langle proof \rangle$

lemma *Collect-minI*: $[\![k \in I;\ P\ (k::('a::wellorder))]\!] \Longrightarrow \exists\, x{\in}I.\ P\ x \wedge (\forall\, y{\in}I.\ y < x \longrightarrow \neg\ P\ y)$
$\langle proof \rangle$

corollary *Collect-minI-ex*: $\exists\, k{\in}I.\ P\ (k::('a::wellorder)) \Longrightarrow \exists\, x{\in}I.\ P\ x \wedge (\forall\, y{\in}I.\ y < x \longrightarrow \neg\ P\ y)$
$\langle proof \rangle$

corollary *Collect-minI-ex2*: $\{k{\in}I.\ P\ (k::('a::wellorder))\} \neq \{\} \Longrightarrow \exists\, x{\in}I.\ P\ x \wedge (\forall\, y{\in}I.\ y < x \longrightarrow \neg\ P\ y)$
$\langle proof \rangle$

lemma *iMin-the*: $iMin\ I = (THE\ x.\ x \in I \wedge (\forall\, y.\ y \in I \longrightarrow x \leq y))$
$\langle proof \rangle$

lemma *iMin-the2*: $iMin\ I = (THE\ x.\ x \in I \wedge (\forall\, y{\in}I.\ x \leq y))$
$\langle proof \rangle$

lemma *iMin-equality*:
$[\![k \in I;\ \bigwedge x.\ x \in I \Longrightarrow k \leq x]\!] \Longrightarrow iMin\ I = k$
$\langle proof \rangle$

lemma *iMin-mono-on*:
$[\![mono\text{-}on\ f\ I;\ \exists\, x.\ x \in I]\!] \Longrightarrow iMin\ (f\ `\ I) = f\ (iMin\ I)$
$\langle proof \rangle$

lemma *iMin-mono-on2*:

$\llbracket$ *mono-on f I*; $I \neq \{\}$ $\rrbracket \Longrightarrow$ *iMin* $(f$ ‘ $I)$ = f (*iMin* I)
⟨*proof*⟩

lemma *iMinI2-order*:
$\llbracket$ $k \in I$; $\bigwedge y.\ y \in I \Longrightarrow k \leq y$;
$\bigwedge x.\ \llbracket\ x \in I;\ \forall\, y{\in}I.\ x \leq y\ \rrbracket \Longrightarrow P\ x$ $\rrbracket \Longrightarrow$
P (*iMin* I)
⟨*proof*⟩

lemma *wellorder-iMin-lemma*:
$k \in I \Longrightarrow$ *iMin* $I \in I \wedge$ *iMin* $I \leq k$
⟨*proof*⟩

lemma *iMin-Min-conv*:
$\llbracket$ *finite* I; $I \neq \{\}$ $\rrbracket \Longrightarrow$ *iMin* I = *Min* I
⟨*proof*⟩

lemma *Min-neq-imp-greater*[*dest*]: $\llbracket$ *finite* I; $k \in I$; $k \neq$ *Min* I $\rrbracket \Longrightarrow$ *Min* $I < k$
⟨*proof*⟩

lemma *Max-neq-imp-less*[*dest*]: $\llbracket$ *finite* I; $k \in I$; $k \neq$ *Max* I $\rrbracket \Longrightarrow k <$ *Max* I
⟨*proof*⟩

lemma *nat-Least-mono*:
$\llbracket$ $A \neq \{\}$; *mono* $(f{::}(nat{\Rightarrow}nat))$ $\rrbracket \Longrightarrow$
$(LEAST\ x.\ x \in f$ ‘ $A) = f\ (LEAST\ x.\ x \in A)$
⟨*proof*⟩

lemma *Least-disj*:
$\llbracket\ \exists\, x.\ P\ x;\ \exists\, x.\ Q\ x\ \rrbracket \Longrightarrow$
$(LEAST\ (x{::}'a{::}wellorder).\ (P\ x \vee Q\ x)) = min\ (LEAST\ x.\ P\ x)\ (LEAST\ x.\ Q$
$x)$
⟨*proof*⟩

lemma *Least-imp-le*:
$\llbracket\ \exists\, x.\ P\ x;\ \bigwedge x.\ P\ x \Longrightarrow Q\ x\ \rrbracket \Longrightarrow$
$(LEAST\ (x{::}'a{::}wellorder).\ Q\ x) \leq (LEAST\ x.\ P\ x)$
⟨*proof*⟩

lemma *Least-imp-disj-eq*:
$\llbracket\ \exists\, x.\ P\ x;\ \bigwedge x.\ P\ x \Longrightarrow Q\ x\ \rrbracket \Longrightarrow$
$(LEAST\ (x{::}'a{::}wellorder).\ P\ x \vee Q\ x) = (LEAST\ x.\ Q\ x)$
⟨*proof*⟩

lemma *Least-le-imp-le*:
$\llbracket\ \exists\, x.\ P\ x;\ \exists\, x.\ Q\ x;\ \bigwedge x\ y.\ \llbracket\ P\ x;\ Q\ y\ \rrbracket \Longrightarrow x \leq y\ \rrbracket \Longrightarrow$
$(LEAST\ (x{::}'a{::}wellorder).\ P\ x) \leq (LEAST\ (x{::}'a{::}wellorder).\ Q\ x)$
⟨*proof*⟩

lemma *Least-le-imp-le-disj*:
$[\![\ \exists\, x.\ P\ x;\ \bigwedge x\ y.\ [\![\ P\ x;\ Q\ y\ ]\!] \Longrightarrow x \le y\ ]\!] \Longrightarrow$
$(LEAST\ (x::'a::wellorder).\ P\ x \vee Q\ x) = (LEAST\ (x::'a::wellorder).\ P\ x)$
⟨*proof*⟩

lemma *Max-equality*: $[\![\ (k::'a::linorder) \in A;\ finite\ A;\ \bigwedge x.\ x \in A \Longrightarrow x \le k\ ]\!] \Longrightarrow$
$Max\ A = k$
⟨*proof*⟩

lemma *not-greater-Max*: $[\![\ finite\ A;\ Max\ A < k\ ]\!] \Longrightarrow k \notin A$
⟨*proof*⟩

lemma *Collect-not-greater-Max*: $[\![\ finite\ \{x.\ P\ x\};\ Max\ \{x.\ P\ x\} < k\ ]\!] \Longrightarrow \neg\ P\ k$
⟨*proof*⟩

lemma *Collect-Max-ge*: $[\![\ finite\ \{x.\ P\ x\};\ P\ k\ ]\!] \Longrightarrow k \le Max\ \{x.\ P\ x\}$
⟨*proof*⟩

lemma *MaxI*: $[\![\ k \in A;\ finite\ A\ ]\!] \Longrightarrow Max\ A \in A$
⟨*proof*⟩

lemma *MaxI-ex*: $[\![\ \exists\, x.\ x \in A;\ finite\ A\ ]\!] \Longrightarrow Max\ A \in A$
⟨*proof*⟩

lemma *MaxI-ex2*: $[\![\ A \ne \{\};\ finite\ A\ ]\!] \Longrightarrow Max\ A \in A$
⟨*proof*⟩

lemma *MaxI2*: $[\![\ k \in A;\ \bigwedge x.\ x \in A \Longrightarrow P\ x;\ finite\ A\ ]\!] \Longrightarrow P\ (Max\ A)$
⟨*proof*⟩

lemma *MaxI2-ex*:$[\![\ \exists\, x.\ x \in A;\ \bigwedge x.\ x \in A \Longrightarrow P\ x;\ finite\ A\ ]\!] \Longrightarrow P\ (Max\ A)$
⟨*proof*⟩

lemma *MaxI2-ex2*:$[\![\ A \ne \{\};\ \bigwedge x.\ x \in A \Longrightarrow P\ x;\ finite\ A\ ]\!] \Longrightarrow P\ (Max\ A)$
⟨*proof*⟩

lemma *Max-mono*: $[\![\ mono\ f;\ \exists\, x.\ x \in A;\ finite\ A\ ]\!] \Longrightarrow Max\ (f\ `\ A) = f\ (Max\ A)$
⟨*proof*⟩

lemma *Max-mono2*:$[\![\ mono\ f;\ A \ne \{\};\ finite\ A\ ]\!] \Longrightarrow Max\ (f\ `\ A) = f\ (Max\ A)$
⟨*proof*⟩

lemma *Max-mono-on*: $[\![\ mono\text{-}on\ f\ A;\ \exists\, x.\ x \in A;\ finite\ A\ ]\!] \Longrightarrow Max\ (f\ `\ A) = f$
$(Max\ A)$
⟨*proof*⟩

lemma *Max-mono-on2*:
$[\![\ mono\text{-}on\ f\ A;\ A \ne \{\};\ finite\ A\ ]\!] \Longrightarrow Max\ (f\ `\ A) = f\ (Max\ A)$

$\langle proof \rangle$

lemma *Max-the*:
$[\![A \neq \{\}; \textit{finite } A]\!] \Longrightarrow$
$Max\ A = (THE\ x.\ x \in A \wedge (\forall y.\ y \in A \longrightarrow y \leq x))$
$\langle proof \rangle$

lemma *Max-the2*: $[\![A \neq \{\}; \textit{finite } A]\!] \Longrightarrow$
$Max\ A = (THE\ x.\ x \in A \wedge (\forall y{\in}A.\ y \leq x))$
$\langle proof \rangle$

lemma *wellorder-Max-lemma*: $[\![k \in A; \textit{finite } A]\!] \Longrightarrow Max\ A \in A \wedge k \leq Max\ A$
$\langle proof \rangle$

lemma *MaxI2-order*: $[\![k \in A; \textit{finite } A; \bigwedge y.\ y \in A \Longrightarrow y \leq k;$
$\bigwedge x.\ [\![x \in A; \forall y{\in}A.\ y \leq x]\!] \Longrightarrow P\ x]\!]$
$\Longrightarrow P\ (Max\ A)$
$\langle proof \rangle$

lemma *Min-le-Max*: $[\![\textit{finite } A; A \neq \{\}]\!] \Longrightarrow Min\ A \leq Max\ A$
$\langle proof \rangle$

lemma *iMin-le-Max*: $[\![\textit{finite } A; A \neq \{\}]\!] \Longrightarrow iMin\ A \leq Max\ A$
$\langle proof \rangle$

6.2.2 *Max* for sets over *enat*

definition *iMax* :: *nat set* $\Rightarrow$ *enat*
where $iMax\ i \equiv if\ (\textit{finite } i)\ then\ (enat\ (Max\ i))\ else\ \infty$

lemma *iMax-finite-conv*: $\textit{finite } I = (iMax\ I \neq \infty)$
$\langle proof \rangle$

lemma *iMax-infinite-conv*: $\textit{infinite } I = (iMax\ I = \infty)$
$\langle proof \rangle$

lemma *class.distrib-lattice* $(min::('a::linorder \Rightarrow 'a \Rightarrow 'a))\ (\leq)\ (<)\ max$
$\langle proof \rangle$

print-locale *Lattices.distrib-lattice*

lemma *max-Min-eq-Min-max*[*rule-format*]:
$\textit{finite } A \Longrightarrow$
$A \neq \{\} \longrightarrow$
$max\ x\ (Min\ A) = Min\ \{max\ x\ a\ |a.\ a \in A\}$
$\langle proof \rangle$

lemma *max-iMin-eq-iMin-max*:
$[\![\textit{finite } A; A \neq \{\}]\!] \Longrightarrow$

$max\ x\ (iMin\ A) = iMin\ \{max\ x\ a\ |a.\ a \in A\}$
⟨*proof*⟩

lemma $[\![\ finite\ A;\ A \neq\{\}\ ]\!] \Longrightarrow \forall\, x{\in}A.\ x \le Max\ A$
⟨*proof*⟩

6.2.3 *Min* and *Max* for set operations

lemma *iMin-subset*: $[\![\ A \neq \{\};\ A \subseteq B\ ]\!] \Longrightarrow iMin\ B \le iMin\ A$
⟨*proof*⟩

lemma *Max-subset*: $[\![\ A \neq \{\};\ A \subseteq B;\ finite\ B\ ]\!] \Longrightarrow Max\ A \le Max\ B$
⟨*proof*⟩

lemma *Min-subset*: $[\![\ A \neq \{\};\ A \subseteq B;\ finite\ B\ ]\!] \Longrightarrow Min\ B \le Min\ A$
⟨*proof*⟩

lemma *iMin-Int-ge1*: $(A \cap B) \neq \{\} \Longrightarrow iMin\ A \le iMin\ (A \cap B)$
⟨*proof*⟩

lemma *iMin-Int-ge2*: $(A \cap B) \neq \{\} \Longrightarrow iMin\ B \le iMin\ (A \cap B)$
⟨*proof*⟩

lemma *iMin-Int-ge*: $(A \cap B) \neq \{\} \Longrightarrow max\ (iMin\ A)\ (iMin\ B) \le iMin\ (A \cap B)$
⟨*proof*⟩

lemma *Max-Int-le1*: $[\![\ (A \cap B) \neq \{\};\ finite\ A\ ]\!] \Longrightarrow Max\ (A \cap B) \le Max\ A$
⟨*proof*⟩

lemma *Max-Int-le2*: $[\![\ (A \cap B) \neq \{\};\ finite\ B\ ]\!] \Longrightarrow Max\ (A \cap B) \le Max\ B$
⟨*proof*⟩

lemma *Max-Int-le*: $[\![\ (A \cap B) \neq \{\};\ finite\ A;\ finite\ B\ ]\!] \Longrightarrow$
$Max\ (A \cap B) \le min\ (Max\ A)\ (Max\ B)$
⟨*proof*⟩

lemma *iMin-Un*[*rule-format*]:
$[\![\ A \neq \{\};\ B \neq \{\}\ ]\!] \Longrightarrow$
$iMin\ (A \cup B) = min\ (iMin\ A)\ (iMin\ B)$
⟨*proof*⟩

lemma *iMin-singleton*[*simp*]: $iMin\ \{a\} = a$
⟨*proof*⟩

lemma *iMax-singleton*[*simp*]: $iMax\ \{a\} = enat\ a$

$\langle proof \rangle$

lemma *Max-le-Min-imp-singleton*:
$[\![$ *finite* A; $A \neq \{\}$; *Max* $A \le$ *Min* A $]\!] \Longrightarrow A = \{$*Min* $A\}$
$\langle proof \rangle$
lemma *Max-le-Min-conv-singleton*:
$[\![$ *finite* A; $A \neq \{\}$ $]\!] \Longrightarrow$ (*Max* $A \le$ *Min* A) $= (\exists\, x.\ A = \{x\})$
$\langle proof \rangle$

lemma *Max-le-iMin-imp-le*:
$[\![$ *finite* A; *Max* $A \le$ *iMin* B; $a \in A$; $b \in B$ $]\!] \Longrightarrow a \le b$
$\langle proof \rangle$

lemma *le-imp-Max-le-iMin*:
$[\![$ *finite* A; $A \neq \{\}$; $B \neq \{\}$; $\forall\, a{\in}A.\ \forall\, b{\in}B.\ a \le b$ $]\!] \Longrightarrow$ *Max* $A \le$ *iMin* B
$\langle proof \rangle$

lemma *Max-le-iMin-conv-le*:
$[\![$ *finite* A; $A \neq \{\}$; $B \neq \{\}$ $]\!] \Longrightarrow$ (*Max* $A \le$ *iMin* B) $= (\forall\, a{\in}A.\ \forall\, b{\in}B.\ a \le b)$
$\langle proof \rangle$

lemma *Max-less-iMin-imp-less*:
$[\![$ *finite* A; *Max* $A <$ *iMin* B; $a \in A$; $b \in B$ $]\!] \Longrightarrow a < b$
$\langle proof \rangle$

lemma *less-imp-Max-less-iMin*:
$[\![$ *finite* A; $A \neq \{\}$; $B \neq \{\}$; $\forall\, a{\in}A.\ \forall\, b{\in}B.\ a < b$ $]\!] \Longrightarrow$ *Max* $A <$ *iMin* B
$\langle proof \rangle$

lemma *Max-less-iMin-conv-less*:
$[\![$ *finite* A; $A \neq \{\}$; $B \neq \{\}$ $]\!] \Longrightarrow$ (*Max* $A <$ *iMin* B) $= (\forall\, a{\in}A.\ \forall\, b{\in}B.\ a < b)$
$\langle proof \rangle$

lemma *Max-less-iMin-imp-disjoint*:
$[\![$ *finite* A; *Max* $A <$ *iMin* B $]\!] \Longrightarrow A \cap B = \{\}$
$\langle proof \rangle$

lemma *iMin-in-idem*: $n \in I \Longrightarrow$ *min* n (*iMin* I) $=$ *iMin* I
$\langle proof \rangle$

lemma *iMin-insert*: $I \neq \{\} \Longrightarrow$ *iMin* (*insert* n I) $=$ *min* n (*iMin* I)
$\langle proof \rangle$

lemma *iMin-insert-remove*:
iMin (*insert* n I) $=$
(*if* $I - \{n\} = \{\}$ *then* n *else* *min* n (*iMin* $(I - \{n\})$))

⟨*proof*⟩

lemma *iMin-remove*: $n \in I \Longrightarrow iMin\ I = (if\ I - \{n\} = \{\}\ then\ n\ else\ min\ n\ (iMin\ (I - \{n\})))$
⟨*proof*⟩

lemma *iMin-subset-idem*: $[\![B \neq \{\}; B \subseteq A]\!] \Longrightarrow min\ (iMin\ B)\ (iMin\ A) = iMin\ A$
⟨*proof*⟩

lemma *iMin-union-inter*: $A \cap B \neq \{\} \Longrightarrow min\ (iMin\ (A \cup B))\ (iMin\ (A \cap B)) = min\ (iMin\ A)\ (iMin\ B)$
⟨*proof*⟩

lemma *iMin-ge-iff*: $I \neq \{\} \Longrightarrow (n \leq iMin\ I) = (\forall a{\in}I.\ n \leq a)$
⟨*proof*⟩

lemma *iMin-gr-iff*: $I \neq \{\} \Longrightarrow (n < iMin\ I) = (\forall a{\in}I.\ n < a)$
⟨*proof*⟩

lemma *iMin-le-iff*: $I \neq \{\} \Longrightarrow (iMin\ I \leq n) = (\exists a{\in}I.\ a \leq n)$
⟨*proof*⟩

lemma *iMin-less-iff*: $I \neq \{\} \Longrightarrow (iMin\ I < n) = (\exists a{\in}I.\ a < n)$
⟨*proof*⟩

lemma *hom-iMin-commute*: $[\![\bigwedge x\ y.\ h\ (min\ x\ y) = min\ (h\ x)\ (h\ y); I \neq \{\}]\!] \Longrightarrow iMin\ (h\ `\ I) = h\ (iMin\ I)$
⟨*proof*⟩

Synonyms for similarity with theorem names for *Min*"

lemmas *iMin-eqI* = *iMin-equality*

lemmas *iMin-in* = *iMinI-ex2*

6.3 Some auxiliary results for set operations

6.3.1 Some additional abbreviations for relations

Abbreviations for *refl*, *sym*, *equiv*, *refl*, *trans*

abbreviation (*input*) *reflP* :: $('a \Rightarrow 'a \Rightarrow bool) \Rightarrow bool$ **where**
  $reflP\ r \equiv refl\ \{(x, y).\ r\ x\ y\}$

abbreviation (*input*) *symP* :: $('a => 'a => bool) => bool$ **where**
  $symP\ r == sym\ \{(x, y).\ r\ x\ y\}$

abbreviation (*input*) *transP* :: $('a => 'a => bool) => bool$ **where**
  $transP\ r == trans\ \{(x, y).\ r\ x\ y\}$

abbreviation (*input*) *equivP* :: ($'a \Rightarrow 'a \Rightarrow bool$) $\Rightarrow$ *bool* **where**
equivP $r \equiv$ *reflP* $r \wedge$ *symP* $r \wedge$ *transp* r

abbreviation (*input*) *irreflP* :: ($'a \Rightarrow 'a \Rightarrow bool$) $\Rightarrow$ *bool* **where**
irreflP $r \equiv$ *irrefl* $\{(x, y).\ r\ x\ y\}$

Example for *reflP*

lemma *reflP* (($\leq$)::($'a$::*preorder* $\Rightarrow 'a \Rightarrow bool$))
⟨*proof*⟩

Example for *symP*

lemma *symP* (=)
⟨*proof*⟩

Example for *equivP*

lemma *equivP* (=)
⟨*proof*⟩

Example for *irreflP*

lemma *irreflP* (($<$)::($'a$::*preorder* $\Rightarrow 'a \Rightarrow bool$))
⟨*proof*⟩

6.3.2 Auxiliary results for *singletons*

lemma *singleton-not-empty*: $\{a\} \neq \{\}$ ⟨*proof*⟩
lemma *singleton-finite*: *finite* $\{a\}$ ⟨*proof*⟩

lemma *singleton-ball*: $(\forall x\in\{a\}.\ P\ x) = P\ a$ ⟨*proof*⟩
lemma *singleton-bex*: $(\exists x\in\{a\}.\ P\ x) = P\ a$ ⟨*proof*⟩

lemma *subset-singleton-conv*: $(A \subseteq \{a\}) = (A = \{\} \vee A = \{a\})$ ⟨*proof*⟩
lemma *singleton-subset-conv*: $(\{a\} \subseteq A) = (a \in A)$ ⟨*proof*⟩

lemma *singleton-eq-conv*: $(\{a\} = \{b\}) = (a = b)$ ⟨*proof*⟩
lemma *singleton-subset-singleton-conv*: $(\{a\} \subseteq \{b\}) = (a = b)$ ⟨*proof*⟩

lemma *singleton-Int1-if*: $\{a\} \cap A = (\mathit{if}\ a \in A\ \mathit{then}\ \{a\}\ \mathit{else}\ \{\})$
⟨*proof*⟩

lemma *singleton-Int2-if*: $A \cap \{a\} = (\mathit{if}\ a \in A\ \mathit{then}\ \{a\}\ \mathit{else}\ \{\})$
⟨*proof*⟩

lemma *singleton-image*: f ‘ $\{a\} = \{f\ a\}$ ⟨*proof*⟩
lemma *inj-on-singleton*: *inj-on* f $\{a\}$ ⟨*proof*⟩
lemma *strict-mono-on-singleton*: *strict-mono-on* f $\{a\}$
⟨*proof*⟩

6.3.3 Auxiliary results for *finite* and *infinite* sets

lemma *infinite-imp-not-singleton*: $\textit{infinite } A \Longrightarrow \neg\ (\exists\, a.\ A = \{a\})$ ⟨*proof*⟩

lemma *infinite-insert*: $\textit{infinite } (\textit{insert } a\ A) = \textit{infinite } A$
⟨*proof*⟩

lemma *infinite-Diff-insert*: $\textit{infinite } (A - \textit{insert } a\ B) = \textit{infinite } (A - B)$
⟨*proof*⟩

lemma *subset-finite-imp-finite*: $[\![\ \textit{finite } A;\ B \subseteq A\ ]\!] \Longrightarrow \textit{finite } B$
⟨*proof*⟩

lemma *infinite-not-subset-finite*: $[\![\ \textit{infinite } A;\ \textit{finite } B\ ]\!] \Longrightarrow \neg\ A \subseteq B$
⟨*proof*⟩

lemma *Un-infinite-right*: $\textit{infinite } T \Longrightarrow \textit{infinite } (S \cup T)$ ⟨*proof*⟩
lemma *Un-infinite-iff*: $\textit{infinite } (S \cup T) = (\textit{infinite } S \vee \textit{infinite } T)$ ⟨*proof*⟩

Give own name to the lemma about finiteness of the integer image of a nat set

corollary *finite-A-int-A-conv*: $\textit{finite } A = \textit{finite } (\textit{int}\ `\ A)$
⟨*proof*⟩

Corresponding fact fo infinite sets

corollary *infinite-A-int-A-conv*: $\textit{infinite } A = \textit{infinite } (\textit{int}\ `\ A)$
⟨*proof*⟩

lemma *cartesian-product-infiniteL-imp-infinite*: $[\![\ \textit{infinite } A;\ B \neq \{\}\ ]\!] \Longrightarrow \textit{infinite } (A \times B)$
⟨*proof*⟩

lemma *cartesian-product-infiniteR-imp-infinite*: $[\![\ \textit{infinite } B;\ A \neq \{\}\ ]\!] \Longrightarrow \textit{infinite } (A \times B)$
⟨*proof*⟩

lemma *finite-nat-iff-bounded2*:
$\textit{finite } S = (\exists\,(k::\textit{nat}).\forall\, n{\in}S.\ n < k)$
⟨*proof*⟩

lemma *finite-nat-iff-bounded-le2*:
$\textit{finite } S = (\exists\,(k::\textit{nat}).\forall\, n{\in}S.\ n \le k)$
⟨*proof*⟩

lemma *nat-asc-chain-imp-unbounded*:
$[\![\ S \neq \{\};\ (\forall\, m{\in}S.\ \exists\, n{\in}S.\ m < (n::\textit{nat}))\ ]\!] \Longrightarrow \forall\, m.\ \exists\, n{\in}S.\ m \le n$
⟨*proof*⟩

lemma *infinite-nat-iff-asc-chain*:
$S \neq \{\} \Longrightarrow \mathit{infinite}\ S = (\forall\, m{\in}S.\ \exists\, n{\in}S.\ m < (n{::}nat))$
$\langle proof \rangle$

lemma *infinite-imp-asc-chain*: $\mathit{infinite}\ S \Longrightarrow \forall\, m{\in}S.\ \exists\, n{\in}S.\ m < (n{::}nat)$
$\langle proof \rangle$

lemma *infinite-image-imp-infinite*: $\mathit{infinite}\ (f\ `\ A) \Longrightarrow \mathit{infinite}\ A$
$\langle proof \rangle$

lemma *inj-on-imp-infinite-image*: $[\![\ \mathit{infinite}\ A;\ \mathit{inj\text{-}on}\ f\ A\ ]\!] \Longrightarrow \mathit{infinite}\ (f\ `\ A)$
$\langle proof \rangle$

lemma *inj-on-infinite-image-iff*: $\mathit{inj\text{-}on}\ f\ A \Longrightarrow \mathit{infinite}\ (f\ `\ A) = \mathit{infinite}\ A$
$\langle proof \rangle$

lemma *inj-on-finite-image-iff*: $\mathit{inj\text{-}on}\ f\ A \Longrightarrow \mathit{finite}\ (f\ `\ A) = \mathit{finite}\ A$
$\langle proof \rangle$

lemma *nat-ex-greater-finite-Max-conv*:
$A \neq \{\} \Longrightarrow (\exists\, x{\in}A.\ (n{::}nat) < x) = (\mathit{finite}\ A \longrightarrow n < \mathit{Max}\ A)$
$\langle proof \rangle$

corollary *nat-ex-greater-infinite-finite-Max-conv'*:
$(\exists\, x{\in}A.\ (n{::}nat) < x) = (\mathit{finite}\ A \wedge A \neq \{\} \wedge n < \mathit{Max}\ A \vee \mathit{infinite}\ A)$
$\langle proof \rangle$

6.3.4 Some auxiliary results for disjoint sets

lemma *disjoint-iff-in-not-in1*: $(A \cap B = \{\}) = (\forall\, x{\in}A.\ x \notin B)$ $\langle proof \rangle$
lemma *disjoint-iff-in-not-in2*: $(A \cap B = \{\}) = (\forall\, x{\in}B.\ x \notin A)$ $\langle proof \rangle$

lemma *disjoint-in-Un*:
$[\![\ A \cap B = \{\};\ x \in A \cup B\ ]\!] \Longrightarrow x \notin A \vee x \notin B$
$\langle proof \rangle$

lemma *partition-Union*: $A \subseteq \bigcup C \Longrightarrow (\bigcup c{\in}C.\ A \cap c) = A$ $\langle proof \rangle$

lemma *disjoint-partition-Int*:
$\forall\, c1{\in}C.\ \forall\, c2{\in}C.\ c1 \neq c2 \longrightarrow c1 \cap c2 = \{\} \Longrightarrow$
$\forall\, a1{\in}\{A \cap c|\ c.\ c \in C\}.\ \forall\, a2{\in}\{A \cap c|\ c.\ c \in C\}.$
$a1 \neq a2 \longrightarrow a1 \cap a2 = \{\}$
$\langle proof \rangle$

lemma $\{f\ x\ |x.\ x \in A\} = (\bigcup x{\in}A.\ \{f\ x\})$
$\langle proof \rangle$

This lemma version drops the superfluous precondition *finite* $(\bigcup\ C)$ (and turns the resulting equation to the sensible order $card\ ..\ = k * card\ ..$).

lemma *card-partition*:
$[\![$ *finite* C; $\bigwedge c.\ c \in C \Longrightarrow card\ c = k$; $\bigwedge c1\ c2.\ [\![c1 \in C;\ c2 \in C;\ c1 \neq c2]\!] \Longrightarrow c1 \cap c2 = \{\}$ $]\!] \Longrightarrow$
$card\ (\bigcup C) = k * card\ C$
⟨*proof*⟩

6.3.5 Some auxiliary results for subset relation

lemma *subset-image-Int*: $A \subseteq B \Longrightarrow f\ `\ (A \cap B) = f\ `\ A \cap f\ `\ B$
⟨*proof*⟩

lemma *image-diff-disjoint-image-Int*:
$[\![\ f\ `\ (A - B) \cap f\ `\ B = \{\}\ ]\!] \Longrightarrow$
$f\ `\ (A \cap B) = f\ `\ A \cap f\ `\ B$
⟨*proof*⟩

lemma *subset-imp-Int-subset1*: $A \subseteq C \Longrightarrow A \cap B \subseteq C$
⟨*proof*⟩

lemma *subset-imp-Int-subset2*: $B \subseteq C \Longrightarrow A \cap B \subseteq C$
⟨*proof*⟩

6.3.6 Auxiliary results for intervals from *SetInterval*

lemmas *set-interval-defs* =
lessThan-def atMost-def
greaterThan-def atLeast-def
greaterThanLessThan-def atLeastLessThan-def
greaterThanAtMost-def atLeastAtMost-def

lemma *image-add-atLeast*:
$(\lambda n{::}nat.\ n{+}k)\ `\ \{i..\} = \{i{+}k..\}$ (**is** $?A = ?B$)
⟨*proof*⟩

lemma *image-add-atMost*:
$(\lambda n{::}nat.\ n{+}k)\ `\ \{..i\} = \{k..i{+}k\}$ (**is** $?A = ?B$)

⟨*proof*⟩

corollary *image-Suc-atLeast*: $Suc\ `\ \{i..\} = \{Suc\ i..\}$
⟨*proof*⟩

corollary *image-Suc-atMost*: $Suc\ `\ \{..i\} = \{Suc\ 0..Suc\ i\}$
⟨*proof*⟩

lemmas *image-add-lemmas* =
image-add-atLeastAtMost
image-add-atLeast

image-add-atMost
lemmas *image-Suc-lemmas* =
image-Suc-atLeastAtMost
image-Suc-atLeast
image-Suc-atMost

lemma *atMost-atLeastAtMost-0-conv*: $\{..i::nat\} = \{0..i\}$
⟨*proof*⟩

lemma *atLeastAtMost-subset-atMost*: $(k::'a::order) \le k' \Longrightarrow \{l..k\} \subseteq \{..k'\}$
⟨*proof*⟩

lemma *lessThan-insert*: *insert* $(n::'a::order)$ $\{..<n\} = \{..n\}$
⟨*proof*⟩

lemma *greaterThan-insert*: *insert* $(n::'a::order)$ $\{n<..\} = \{n..\}$
⟨*proof*⟩

lemma *atMost-remove*: $\{..n\} - \{(n::'a::order)\} = \{..<n\}$
⟨*proof*⟩

lemma *atLeast-remove*: $\{n..\} - \{(n::'a::order)\} = \{n<..\}$
⟨*proof*⟩

lemma *atMost-lessThan-conv*: $\{..n\} = \{..<Suc\ n\}$
⟨*proof*⟩

lemma *atLeastAtMost-atLeastLessThan-conv*: $\{l..u\} = \{l..<Suc\ u\}$
⟨*proof*⟩

lemma *atLeast-greaterThan-conv*: $\{Suc\ n..\} = \{n<..\}$
⟨*proof*⟩

lemma *atLeastAtMost-greaterThanAtMost-conv*: $\{Suc\ l..u\} = \{l<..u\}$
⟨*proof*⟩

lemma *finite-subset-atLeastAtMost*: *finite* $A \Longrightarrow A \subseteq \{Min\ A..Max\ A\}$
⟨*proof*⟩

lemma *Max-le-imp-subset-atMost*:
$[\![$ *finite* A; $Max\ A \le n$ $]\!] \Longrightarrow A \subseteq \{..n\}$
⟨*proof*⟩

lemma *subset-atMost-imp-Max-le*:
$[\![$ *finite* A; $A \ne \{\}$; $A \subseteq \{..n\}$ $]\!] \Longrightarrow Max\ A \le n$
⟨*proof*⟩

lemma *subset-atMost-Max-le-conv*:
$[\![\ finite\ A;\ A \neq \{\}\ ]\!] \Longrightarrow (A \subseteq \{..n\}) = (Max\ A \leq n)$
⟨*proof*⟩

lemma *iMin-atLeast*: $iMin\ \{n..\} = n$
⟨*proof*⟩

lemma *iMin-greaterThan*: $iMin\ \{n{<}..\} = Suc\ n$
⟨*proof*⟩

lemma *iMin-atMost*: $iMin\ \{..(n::nat)\} = 0$
⟨*proof*⟩

lemma *iMin-lessThan*: $0 < n \Longrightarrow iMin\ \{..{<}(n::nat)\} = 0$
⟨*proof*⟩

lemma *Max-atMost*: $Max\ \{..(n::nat)\} = n$
⟨*proof*⟩

lemma *Max-lessThan*: $0 < n \Longrightarrow Max\ \{..{<}n\} = n - Suc\ 0$
⟨*proof*⟩

lemma *iMin-atLeastLessThan*: $m < n \Longrightarrow iMin\ \{m..{<}n\} = m$
⟨*proof*⟩

lemma *Max-atLeastLessThan*: $m < n \Longrightarrow Max\ \{m..{<}n\} = n - Suc\ 0$
⟨*proof*⟩

lemma *iMin-greaterThanLessThan*: $Suc\ m < n \Longrightarrow iMin\ \{m{<}..{<}n\} = Suc\ m$
⟨*proof*⟩

lemma *Max-greaterThanLessThan*: $Suc\ m < n \Longrightarrow Max\ \{m{<}..{<}n\} = n - Suc\ 0$
⟨*proof*⟩

lemma *iMin-greaterThanAtMost*: $m < n \Longrightarrow iMin\ \{m{<}..n\} = Suc\ m$
⟨*proof*⟩

lemma *Max-greaterThanAtMost*: $m < n \Longrightarrow Max\ \{m{<}..(n::nat)\} = n$
⟨*proof*⟩

lemma *iMin-atLeastAtMost*: $m \leq n \Longrightarrow iMin\ \{m..n\} = m$
⟨*proof*⟩

lemma *Max-atLeastAtMost*: $m \leq n \Longrightarrow Max\ \{m..(n::nat)\} = n$
⟨*proof*⟩

lemma *infinite-atLeast*: $infinite\ \{(n::nat)..\}$

⟨*proof*⟩

lemma *infinite-greaterThan*: *infinite* $\{(n::nat)<..\}$
⟨*proof*⟩

lemma *infinite-atLeast-int*: *infinite* $\{(n::int)..\}$
⟨*proof*⟩

lemma *infinite-greaterThan-int*: *infinite* $\{(n::int)<..\}$
⟨*proof*⟩

lemma *infinite-atMost-int*: *infinite* $\{..(n::int)\}$
⟨*proof*⟩

lemma *infinite-lessThan-int*: *infinite* $\{..<(n::int)\}$
⟨*proof*⟩

6.3.7 Auxiliary results for *card*

lemma *sum-singleton*: $(\sum x \in \{a\}.\ f\ x) = f\ a$
⟨*proof*⟩

lemma *card-singleton*: $card\ \{a\} = Suc\ 0$
⟨*proof*⟩

lemma *card-cartesian-product-singleton-right*: $card\ (A \times \{x\}) = card\ A$
⟨*proof*⟩

lemma *card-1-imp-singleton*: $card\ A = Suc\ 0 \Longrightarrow (\exists\, a.\ A = \{a\})$
⟨*proof*⟩

lemma *card-1-singleton-conv*: $(card\ A = Suc\ 0) = (\exists\, a.\ A = \{a\})$
⟨*proof*⟩

lemma *card-gr0-imp-finite*: $0 < card\ A \Longrightarrow finite\ A$
⟨*proof*⟩
lemma *card-gr0-imp-not-empty*: $(0 < card\ A) \Longrightarrow A \neq \{\}$
⟨*proof*⟩
lemma *not-empty-card-gr0-conv*: $finite\ A \Longrightarrow (A \neq \{\}) = (0 < card\ A)$
⟨*proof*⟩

lemma *nat-card-le-Max*: $card\ (A::nat\ set) \leq Suc\ (Max\ A)$
⟨*proof*⟩

lemma *Int-card1*: $finite\ A \Longrightarrow card\ (A \cap B) \leq card\ A$
⟨*proof*⟩
lemma *Int-card2*: $finite\ B \Longrightarrow card\ (A \cap B) \leq card\ B$
⟨*proof*⟩
lemma *Un-card1*: $[\![\ finite\ A;\ finite\ B\ ]\!] \Longrightarrow card\ A \leq card\ (A \cup B)$

$\langle proof \rangle$
lemma *Un-card2*: $[\![$ *finite* A; *finite* B $]\!] \Longrightarrow$ *card* $B \le$ *card* $(A \cup B)$
$\langle proof \rangle$

lemma *card-Un-conv*:
$[\![$ *finite* A; *finite* B $]\!] \Longrightarrow$
card $(A \cup B) =$ *card* $A +$ *card* $B -$ *card* $(A \cap B)$
$\langle proof \rangle$
lemma *card-Int-conv*:
$[\![$ *finite* A; *finite* B $]\!] \Longrightarrow$
card $(A \cap B) =$ *card* $A +$ *card* $B -$ *card* $(A \cup B)$
$\langle proof \rangle$

Pigeonhole principle, dirichlet's box principle

lemma *pigeonhole-principle*[*rule-format*]:
card $(f$ ` $A) <$ *card* $A \longrightarrow (\exists x{\in}A.\ \exists y{\in}A.\ x \neq y \wedge f\ x = f\ y)$
$\langle proof \rangle$

corollary *pigeonhole-principle-linorder*[*rule-format*]:
card $(f$ ` $A) <$ *card* $(A{::}'a{::}linorder\ set) \Longrightarrow (\exists x{\in}A.\ \exists y{\in}A.\ x < y \wedge f\ x = f\ y)$
$\langle proof \rangle$

corollary *pigeonhole-mod*:
$[\![\ 0 < m;\ m <$ *card* $A\ ]\!] \Longrightarrow \exists x{\in}A.\ \exists y{\in}A.\ x < y \wedge x\ mod\ m = y\ mod\ m$
$\langle proof \rangle$

corollary *pigeonhole-mod2*:
$[\![\ (0{::}nat) < m;\ m \le c;\ inj\text{-}on\ f\ \{..c\}\ ]\!] \Longrightarrow \exists x{\le}c.\ \exists y{\le}c.\ x < y \wedge f\ x\ mod\ m = f\ y\ mod\ m$
$\langle proof \rangle$

end

7 Cutting linearly ordered and natural sets

theory *SetIntervalCut*
imports *SetInterval2*
begin

7.1 Set restriction

A set to set function f is a *set restriction*, if there exists a predicate P, so that for every set s the function result $f\ s$ contains all its elements fulfilling P

definition *set-restriction* :: $('a\ set \Rightarrow 'a\ set) \Rightarrow bool$
where *set-restriction* $f \equiv \exists P.\ \forall A.\ f\ A = \{x \in A.\ P\ x\}$

lemma *set-restrictionD*: *set-restriction* $f \Longrightarrow \exists P.\ \forall A.\ f\ A = \{x \in A.\ P\ x\}$

⟨*proof*⟩
lemma *set-restrictionD-spec*: $\textit{set-restriction}\ f \Longrightarrow \exists P.\ f\ A = \{x \in A.\ P\ x\}$
⟨*proof*⟩
lemma *set-restrictionI*: $f = (\lambda A.\ \{x \in A.\ P\ x\}) \Longrightarrow \textit{set-restriction}\ f$
⟨*proof*⟩

lemma *set-restriction-comp*:
$[\![\ \textit{set-restriction}\ f;\ \textit{set-restriction}\ g\ ]\!] \Longrightarrow \textit{set-restriction}\ (f \circ g)$
⟨*proof*⟩
lemma *set-restriction-commute*:
$[\![\ \textit{set-restriction}\ f;\ \textit{set-restriction}\ g\ ]\!] \Longrightarrow f\ (g\ I) = g\ (f\ I)$
⟨*proof*⟩

Constructs a set restriction function with the given restriction predicate

definition
$\textit{set-restriction-fun} :: ('a \Rightarrow \textit{bool}) \Rightarrow ('a\ \textit{set} \Rightarrow 'a\ \textit{set})$
where
$\textit{set-restriction-fun}\ P \equiv \lambda A.\ \{x \in A.\ P\ x\}$

lemma *set-restriction-fun-is-set-restriction*:
$\textit{set-restriction}\ (\textit{set-restriction-fun}\ P)$
⟨*proof*⟩

lemma *set-restriction-Int-conv*:
$\textit{set-restriction}\ f = (\exists B.\ \forall A.\ f\ A = A \cap B)$
⟨*proof*⟩

lemma *set-restriction-Un*:
$\textit{set-restriction}\ f \Longrightarrow f\ (A \cup B) = f\ A \cup f\ B$
⟨*proof*⟩
lemma *set-restriction-Int*:
$\textit{set-restriction}\ f \Longrightarrow f\ (A \cap B) = f\ A \cap f\ B$
⟨*proof*⟩
lemma *set-restriction-Diff*:
$\textit{set-restriction}\ f \Longrightarrow f\ (A - B) = f\ A - f\ B$
⟨*proof*⟩
lemma *set-restriction-mono*:
$[\![\ \textit{set-restriction}\ f;\ A \subseteq B\ ]\!] \Longrightarrow f\ A \subseteq f\ B$
⟨*proof*⟩
lemma *set-restriction-absorb*:
$\textit{set-restriction}\ f \Longrightarrow f\ (f\ A) = f\ A$
⟨*proof*⟩
lemma *set-restriction-empty*:
$\textit{set-restriction}\ f \Longrightarrow f\ \{\} = \{\}$
⟨*proof*⟩
lemma *set-restriction-non-empty-imp*:
$[\![\ \textit{set-restriction}\ f;\ f\ A \neq \{\}\ ]\!] \Longrightarrow A \neq \{\}$
⟨*proof*⟩
lemma *set-restriction-subset*:

$set\text{-}restriction\ f \Longrightarrow f\ A \subseteq A$
⟨*proof*⟩
lemma *set-restriction-finite*:
$[\![\ set\text{-}restriction\ f;\ finite\ A\ ]\!] \Longrightarrow finite\ (f\ A)$
⟨*proof*⟩
lemma *set-restriction-card*:
$[\![\ set\text{-}restriction\ f;\ finite\ A\ ]\!] \Longrightarrow$
$card\ (f\ A) = card\ A - card\ \{a \in A.\ f\ \{a\} = \{\}\}$
⟨*proof*⟩

lemma *set-restriction-card-le*:
$[\![\ set\text{-}restriction\ f;\ finite\ A\ ]\!] \Longrightarrow card\ (f\ A) \le card\ A$
⟨*proof*⟩

lemma *set-restriction-not-in-imp*:
$[\![\ set\text{-}restriction\ f;\ x \notin A\ ]\!] \Longrightarrow x \notin f\ A$
⟨*proof*⟩

lemma *set-restriction-in-imp*:
$[\![\ set\text{-}restriction\ f;\ x \in f\ A\ ]\!] \Longrightarrow x \in A$
⟨*proof*⟩

lemma *set-restriction-fun-singleton*:
$set\text{-}restriction\text{-}fun\ P\ \{a\} = (if\ P\ a\ then\ \{a\}\ else\ \{\})$
⟨*proof*⟩
lemma *set-restriction-fun-all-conv*:
$((set\text{-}restriction\text{-}fun\ P)\ A = A) = (\forall\, x{\in}A.\ P\ x)$
⟨*proof*⟩
lemma *set-restriction-fun-empty-conv*:
$((set\text{-}restriction\text{-}fun\ P)\ A = \{\}) = (\forall\, x{\in}A.\ \neg\ P\ x)$
⟨*proof*⟩

7.2 Cut operators for sets/intervals

7.2.1 Definitions and basic lemmata for cut operators

definition *cut-le* :: $'a{::}linorder\ set \Rightarrow\ 'a \Rightarrow\ 'a\ set$ (**infixl** $\downarrow\le$ *100*)
where $I \downarrow\le t \equiv \{x{\in}I.\ x \le t\}$

definition *cut-less* :: $'a{::}linorder\ set \Rightarrow\ 'a \Rightarrow\ 'a\ set$ (**infixl** $\downarrow<$ *100*)
where $I \downarrow< t \equiv \{x{\in}I.\ x < t\}$

definition *cut-ge* :: $'a{::}linorder\ set \Rightarrow\ 'a \Rightarrow\ 'a\ set$ (**infixl** $\downarrow\ge$ *100*)
where $I \downarrow\ge t \equiv \{x{\in}I.\ t \le x\}$

definition *cut-greater* :: $'a{::}linorder\ set \Rightarrow\ 'a \Rightarrow\ 'a\ set$ (**infixl** $\downarrow>$ *100*)
where $I \downarrow> t \equiv \{x{\in}I.\ t < x\}$

lemmas *i-cut-defs* =
cut-le-def cut-less-def

cut-ge-def cut-greater-def

lemma *cut-le-mem-iff*: $x \in I \downarrow\leq t = (x \in I \wedge x \leq t)$
⟨*proof*⟩

lemma *cut-less-mem-iff*: $x \in I \downarrow< t = (x \in I \wedge x < t)$
⟨*proof*⟩

lemma *cut-ge-mem-iff*: $x \in I \downarrow\geq t = (x \in I \wedge t \leq x)$
⟨*proof*⟩

lemma *cut-greater-mem-iff*: $x \in I \downarrow> t = (x \in I \wedge t < x)$
⟨*proof*⟩

lemmas *i-cut-mem-iff* =
cut-le-mem-iff cut-less-mem-iff
cut-ge-mem-iff cut-greater-mem-iff

lemma
cut-leI [*intro*!]: $x \in I \Longrightarrow x \leq t \Longrightarrow x \in I \downarrow\leq t$ **and**
cut-lessI [*intro*!]: $x \in I \Longrightarrow x < t \Longrightarrow x \in I \downarrow< t$ **and**
cut-geI [*intro*!]: $x \in I \Longrightarrow x \geq t \Longrightarrow x \in I \downarrow\geq t$ **and**
cut-greaterI [*intro*!]: $x \in I \Longrightarrow x > t \Longrightarrow x \in I \downarrow> t$
⟨*proof*⟩

lemma
cut-leE [*elim*!]: $x \in I \downarrow\leq t \Longrightarrow (x \in I \Longrightarrow x \leq t \Longrightarrow P) \Longrightarrow P$ **and**
cut-lessE [*elim*!]: $x \in I \downarrow< t \Longrightarrow (x \in I \Longrightarrow x < t \Longrightarrow P) \Longrightarrow P$ **and**
cut-geE [*elim*!]: $x \in I \downarrow\geq t \Longrightarrow (x \in I \Longrightarrow x \geq t \Longrightarrow P) \Longrightarrow P$ **and**
cut-greaterE [*elim*!]: $x \in I \downarrow> t \Longrightarrow (x \in I \Longrightarrow x > t \Longrightarrow P) \Longrightarrow P$
⟨*proof*⟩

lemma
cut-less-bound: $n \in I \downarrow< t \Longrightarrow n < t$ **and**
cut-le-bound: $n \in I \downarrow\leq t \Longrightarrow n \leq t$ **and**
cut-greater-bound: $n \in i \downarrow> t \Longrightarrow t < n$ **and**
cut-ge-bound: $n \in i \downarrow\geq t \Longrightarrow t \leq n$
⟨*proof*⟩

lemmas *i-cut-bound* =
cut-less-bound cut-le-bound
cut-greater-bound cut-ge-bound

lemma
cut-le-Int-conv: $I \downarrow\leq t = I \cap \{..t\}$ **and**
cut-less-Int-conv: $I \downarrow< t = I \cap \{..<t\}$ **and**
cut-ge-Int-conv: $I \downarrow\geq t = I \cap \{t..\}$ **and**
cut-greater-Int-conv: $I \downarrow> t = I \cap \{t<..\}$

$\langle proof \rangle$

lemmas *i-cut-Int-conv* =
cut-le-Int-conv cut-less-Int-conv
cut-ge-Int-conv cut-greater-Int-conv

lemma
cut-le-Diff-conv: $I \downarrow\leq t = I - \{t<..\}$ **and**
cut-less-Diff-conv: $I \downarrow< t = I - \{t..\}$ **and**
cut-ge-Diff-conv: $I \downarrow\geq t = I - \{..<t\}$ **and**
cut-greater-Diff-conv: $I \downarrow> t = I - \{..t\}$
$\langle proof \rangle$
lemmas *i-cut-Diff-conv* =
cut-le-Diff-conv cut-less-Diff-conv
cut-ge-Diff-conv cut-greater-Diff-conv

7.2.2 Basic results for cut operators

lemma
cut-less-eq-set-restriction-fun': $(\lambda I.\ I \downarrow< t)$ = *set-restriction-fun* $(\lambda x.\ x < t)$
and
cut-le-eq-set-restriction-fun': $(\lambda I.\ I \downarrow\leq t)$ = *set-restriction-fun* $(\lambda x.\ x \leq t)$
and
cut-greater-eq-set-restriction-fun': $(\lambda I.\ I \downarrow> t)$ = *set-restriction-fun* $(\lambda x.\ x > t)$
and
cut-ge-eq-set-restriction-fun': $(\lambda I.\ I \downarrow\geq t)$ = *set-restriction-fun* $(\lambda x.\ x \geq t)$
$\langle proof \rangle$
lemmas *i-cut-eq-set-restriction-fun'* =
cut-less-eq-set-restriction-fun' cut-le-eq-set-restriction-fun'
cut-greater-eq-set-restriction-fun' cut-ge-eq-set-restriction-fun'

lemma
cut-less-eq-set-restriction-fun: $I \downarrow< t$ = *set-restriction-fun* $(\lambda x.\ x < t)\ I$ **and**
cut-le-eq-set-restriction-fun: $I \downarrow\leq t$ = *set-restriction-fun* $(\lambda x.\ x \leq t)\ I$ **and**
cut-greater-eq-set-restriction-fun: $I \downarrow> t$ = *set-restriction-fun* $(\lambda x.\ x > t)\ I$ **and**
cut-ge-eq-set-restriction-fun: $I \downarrow\geq t$ = *set-restriction-fun* $(\lambda x.\ x \geq t)\ I$
$\langle proof \rangle$
lemmas *i-cut-eq-set-restriction-fun* =
cut-less-eq-set-restriction-fun cut-le-eq-set-restriction-fun
cut-greater-eq-set-restriction-fun cut-ge-eq-set-restriction-fun

lemma *i-cut-set-restriction-disj*:
$[\![$ *cut-op* = $(\downarrow<)$ $\vee$ *cut-op* = $(\downarrow\leq)$ $\vee$
cut-op = $(\downarrow>)$ $\vee$ *cut-op* = $(\downarrow\geq)$;
$f = (\lambda I.$ *cut-op* $I\ t)$ $]\!] \Longrightarrow$ *set-restriction f*
$\langle proof \rangle$

corollary
i-cut-less-set-restriction: *set-restriction* $(\lambda I.\ I \downarrow< t)$ **and**

i-cut-le-set-restriction: *set-restriction* $(\lambda I.\ I \downarrow\leq t)$ **and**
i-cut-greater-set-restriction: *set-restriction* $(\lambda I.\ I \downarrow> t)$ **and**
i-cut-ge-set-restriction: *set-restriction* $(\lambda I.\ I \downarrow\geq t)$
⟨*proof*⟩

lemmas *i-cut-set-restriction* =
i-cut-le-set-restriction i-cut-less-set-restriction
i-cut-ge-set-restriction i-cut-greater-set-restriction

lemma *i-cut-commute-disj*: $[\![$
cut-op1 = $(\downarrow<)$ $\vee$ *cut-op1* = $(\downarrow\leq)$ $\vee$
cut-op1 = $(\downarrow>)$ $\vee$ *cut-op1* = $(\downarrow\geq)$;
cut-op2 = $(\downarrow<)$ $\vee$ *cut-op2* = $(\downarrow\leq)$ $\vee$
cut-op2 = $(\downarrow>)$ $\vee$ *cut-op2* = $(\downarrow\geq)$ $]\!] \Longrightarrow$
cut-op2 (*cut-op1 I t1*) *t2* = *cut-op1* (*cut-op2 I t2*) *t1*
⟨*proof*⟩

lemma
cut-less-empty: $\{\} \downarrow< t = \{\}$ **and**
cut-le-empty: $\{\} \downarrow\leq t = \{\}$ **and**
cut-greater-empty: $\{\} \downarrow> t = \{\}$ **and**
cut-ge-empty: $\{\} \downarrow\geq t = \{\}$
⟨*proof*⟩

lemmas *i-cut-empty* =
cut-less-empty cut-le-empty
cut-greater-empty cut-ge-empty

lemma
cut-less-not-empty-imp: $I \downarrow< t \neq \{\} \Longrightarrow I \neq \{\}$ **and**
cut-le-not-empty-imp: $I \downarrow\leq t \neq \{\} \Longrightarrow I \neq \{\}$ **and**
cut-greater-not-empty-imp: $I \downarrow> t \neq \{\} \Longrightarrow I \neq \{\}$ **and**
cut-ge-not-empty-imp: $I \downarrow\geq t \neq \{\} \Longrightarrow I \neq \{\}$
⟨*proof*⟩

lemma
cut-less-singleton: $\{a\} \downarrow< t = (\textit{if } a < t \textit{ then } \{a\} \textit{ else } \{\})$ **and**
cut-le-singleton: $\{a\} \downarrow\leq t = (\textit{if } a \leq t \textit{ then } \{a\} \textit{ else } \{\})$ **and**
cut-greater-singleton: $\{a\} \downarrow> t = (\textit{if } a > t \textit{ then } \{a\} \textit{ else } \{\})$ **and**
cut-ge-singleton: $\{a\} \downarrow\geq t = (\textit{if } a \geq t \textit{ then } \{a\} \textit{ else } \{\})$
⟨*proof*⟩
lemmas *i-cut-singleton* =
cut-le-singleton cut-less-singleton
cut-ge-singleton cut-greater-singleton

lemma
cut-less-subset: $I \downarrow< t \subseteq I$ **and**
cut-le-subset: $I \downarrow\le t \subseteq I$ **and**
cut-greater-subset: $I \downarrow> t \subseteq I$ **and**
cut-ge-subset: $I \downarrow\ge t \subseteq I$
⟨*proof*⟩

lemmas *i-cut-subset* =
cut-less-subset cut-le-subset
cut-greater-subset cut-ge-subset

lemma *i-cut-Un-disj*:
$\llbracket$ *cut-op* = $(\downarrow<)$ ∨ *cut-op* = $(\downarrow\le)$ ∨
cut-op = $(\downarrow>)$ ∨ *cut-op* = $(\downarrow\ge)$ $\rrbracket$
$\Longrightarrow$ *cut-op* $(A \cup B)$ t = *cut-op* A t ∪ *cut-op* B t
⟨*proof*⟩

corollary
cut-less-Un: $(A \cup B) \downarrow< t = A \downarrow< t \cup B \downarrow< t$ **and**
cut-le-Un: $(A \cup B) \downarrow\le t = A \downarrow\le t \cup B \downarrow\le t$ **and**
cut-greater-Un: $(A \cup B) \downarrow> t = A \downarrow> t \cup B \downarrow> t$ **and**
cut-ge-Un: $(A \cup B) \downarrow\ge t = A \downarrow\ge t \cup B \downarrow\ge t$
⟨*proof*⟩
lemmas *i-cut-Un* =
cut-less-Un cut-le-Un
cut-greater-Un cut-ge-Un

lemma *i-cut-Int-disj*:
$\llbracket$ *cut-op* = $(\downarrow<)$ ∨ *cut-op* = $(\downarrow\le)$ ∨
cut-op = $(\downarrow>)$ ∨ *cut-op* = $(\downarrow\ge)$ $\rrbracket$
$\Longrightarrow$ *cut-op* $(A \cap B)$ t = *cut-op* A t ∩ *cut-op* B t
⟨*proof*⟩

lemma
cut-less-Int: $(A \cap B) \downarrow< t = A \downarrow< t \cap B \downarrow< t$ **and**
cut-le-Int: $(A \cap B) \downarrow\le t = A \downarrow\le t \cap B \downarrow\le t$ **and**
cut-greater-Int: $(A \cap B) \downarrow> t = A \downarrow> t \cap B \downarrow> t$ **and**
cut-ge-Int: $(A \cap B) \downarrow\ge t = A \downarrow\ge t \cap B \downarrow\ge t$
⟨*proof*⟩
lemmas *i-cut-Int* =
cut-less-Int cut-le-Int
cut-greater-Int cut-ge-Int

lemma
cut-less-Int-left: $(A \cap B) \downarrow< t = A \downarrow< t \cap B$ **and**

cut-le-Int-left: $(A \cap B) \downarrow\leq t = A \downarrow\leq t \cap B$ **and**
cut-greater-Int-left: $(A \cap B) \downarrow> t = A \downarrow> t \cap B$ **and**
cut-ge-Int-left: $(A \cap B) \downarrow\geq t = A \downarrow\geq t \cap B$
⟨*proof*⟩
lemmas *i-cut-Int-left* =
cut-less-Int-left cut-le-Int-left
cut-greater-Int-left cut-ge-Int-left

lemma
cut-less-Int-right: $(A \cap B) \downarrow< t = A \cap B \downarrow< t$ **and**
cut-le-Int-right: $(A \cap B) \downarrow\leq t = A \cap B \downarrow\leq t$ **and**
cut-greater-Int-right: $(A \cap B) \downarrow> t = A \cap B \downarrow> t$ **and**
cut-ge-Int-right: $(A \cap B) \downarrow\geq t = A \cap B \downarrow\geq t$
⟨*proof*⟩
lemmas *i-cut-Int-right* =
cut-less-Int-right cut-le-Int-right
cut-greater-Int-right cut-ge-Int-right

lemma *i-cut-Diff-disj*:
$[\![$ *cut-op* $= (\downarrow<) \vee$ *cut-op* $= (\downarrow\leq) \vee$
cut-op $= (\downarrow>) \vee$ *cut-op* $= (\downarrow\geq)$ $]\!]$
$\Longrightarrow$ *cut-op* $(A - B)$ t = *cut-op* A t − *cut-op* B t
⟨*proof*⟩
corollary
cut-less-Diff: $(A - B) \downarrow< t = A \downarrow< t - B \downarrow< t$ **and**
cut-le-Diff: $(A - B) \downarrow\leq t = A \downarrow\leq t - B \downarrow\leq t$ **and**
cut-greater-Diff: $(A - B) \downarrow> t = A \downarrow> t - B \downarrow> t$ **and**
cut-ge-Diff: $(A - B) \downarrow\geq t = A \downarrow\geq t - B \downarrow\geq t$
⟨*proof*⟩
lemmas *i-cut-Diff* =
cut-less-Diff cut-le-Diff
cut-greater-Diff cut-ge-Diff

lemma *i-cut-subset-mono-disj*:
$[\![$ *cut-op* $= (\downarrow<) \vee$ *cut-op* $= (\downarrow\leq) \vee$
cut-op $= (\downarrow>) \vee$ *cut-op* $= (\downarrow\geq)$; $A \subseteq B$ $]\!]$
$\Longrightarrow$ *cut-op* A $t \subseteq$ *cut-op* B t
⟨*proof*⟩

corollary
cut-less-subset-mono: $A \subseteq B \Longrightarrow A \downarrow< t \subseteq B \downarrow< t$ **and**
cut-le-subset-mono: $A \subseteq B \Longrightarrow A \downarrow\leq t \subseteq B \downarrow\leq t$ **and**
cut-greater-subset-mono: $A \subseteq B \Longrightarrow A \downarrow> t \subseteq B \downarrow> t$ **and**
cut-ge-subset-mono: $A \subseteq B \Longrightarrow A \downarrow\geq t \subseteq B \downarrow\geq t$
⟨*proof*⟩

lemmas *i-cut-subset-mono* =
cut-less-subset-mono cut-le-subset-mono
cut-greater-subset-mono cut-ge-subset-mono

lemma
cut-less-mono: $t \le t' \Longrightarrow I \downarrow< t \subseteq I \downarrow< t'$ **and**
cut-greater-mono: $t' \le t \Longrightarrow I \downarrow> t \subseteq I \downarrow> t'$ **and**
cut-le-mono: $t \le t' \Longrightarrow I \downarrow\le t \subseteq I \downarrow\le t'$ **and**
cut-ge-mono: $t' \le t \Longrightarrow I \downarrow\ge t \subseteq I \downarrow\ge t'$
$\langle proof \rangle$
lemmas *i-cut-mono* =
cut-le-mono cut-less-mono
cut-ge-mono cut-greater-mono

lemma
cut-cut-le: $i \downarrow\le a \downarrow\le b = i \downarrow\le min\ a\ b$ **and**
cut-cut-less: $i \downarrow< a \downarrow< b = i \downarrow< min\ a\ b$ **and**
cut-cut-ge: $i \downarrow\ge a \downarrow\ge b = i \downarrow\ge max\ a\ b$ **and**
cut-cut-greater: $i \downarrow> a \downarrow> b = i \downarrow> max\ a\ b$
$\langle proof \rangle$
lemmas *i-cut-cut* =
cut-cut-le cut-cut-less
cut-cut-ge cut-cut-greater

lemma *i-cut-absorb-disj*:
$[\![$ *cut-op* = $(\downarrow<) \vee$ *cut-op* = $(\downarrow\le) \vee$
cut-op = $(\downarrow>) \vee$ *cut-op* = $(\downarrow\ge)$ $]\!]$
$\Longrightarrow$ *cut-op* (*cut-op* $I\ t$) t = *cut-op* $I\ t$
$\langle proof \rangle$

corollary
cut-le-absorb: $I \downarrow\le t \downarrow\le t = I \downarrow\le t$ **and**
cut-less-absorb: $I \downarrow< t \downarrow< t = I \downarrow< t$ **and**
cut-ge-absorb: $I \downarrow\ge t \downarrow\ge t = I \downarrow\ge t$ **and**
cut-greater-absorb: $I \downarrow> t \downarrow> t = I \downarrow> t$
$\langle proof \rangle$

lemmas *i-cut-absorb* =
cut-le-absorb cut-less-absorb
cut-ge-absorb cut-greater-absorb

lemma
cut-less-0-empty: $I \downarrow< (0::nat) = \{\}$ **and**
cut-ge-0-all: $I \downarrow\ge (0::nat) = I$
$\langle proof \rangle$

lemma
cut-le-all-iff: $(I \downarrow\leq t = I) = (\forall x\in I.\ x \leq t)$ **and**
cut-less-all-iff: $(I \downarrow< t = I) = (\forall x\in I.\ x < t)$ **and**
cut-ge-all-iff: $(I \downarrow\geq t = I) = (\forall x\in I.\ x \geq t)$ **and**
cut-greater-all-iff: $(I \downarrow> t = I) = (\forall x\in I.\ x > t)$
$\langle proof \rangle$

lemmas *i-cut-all-iff* =
cut-le-all-iff cut-less-all-iff
cut-ge-all-iff cut-greater-all-iff

lemma
cut-le-empty-iff: $(I \downarrow\leq t = \{\}) = (\forall x\in I.\ t < x)$ **and**
cut-less-empty-iff: $(I \downarrow< t = \{\}) = (\forall x\in I.\ t \leq x)$ **and**
cut-ge-empty-iff: $(I \downarrow\geq t = \{\}) = (\forall x\in I.\ x < t)$ **and**
cut-greater-empty-iff: $(I \downarrow> t = \{\}) = (\forall x\in I.\ x \leq t)$
$\langle proof \rangle$

lemmas *i-cut-empty-iff* =
cut-le-empty-iff cut-less-empty-iff
cut-ge-empty-iff cut-greater-empty-iff

lemma
cut-le-not-empty-iff: $(I \downarrow\leq t \neq \{\}) = (\exists x\in I.\ x \leq t)$ **and**
cut-less-not-empty-iff: $(I \downarrow< t \neq \{\}) = (\exists x\in I.\ x < t)$ **and**
cut-ge-not-empty-iff: $(I \downarrow\geq t \neq \{\}) = (\exists x\in I.\ t \leq x)$ **and**
cut-greater-not-empty-iff: $(I \downarrow> t \neq \{\}) = (\exists x\in I.\ t < x)$
$\langle proof \rangle$

lemmas *i-cut-not-empty-iff* =
cut-le-not-empty-iff cut-less-not-empty-iff
cut-ge-not-empty-iff cut-greater-not-empty-iff

lemma *nat-cut-ge-infinite-not-empty*: *infinite* $I \Longrightarrow I \downarrow\geq (t{::}nat) \neq \{\}$
$\langle proof \rangle$

lemma *nat-cut-greater-infinite-not-empty*: *infinite* $I \Longrightarrow I \downarrow> (t{::}nat) \neq \{\}$
$\langle proof \rangle$

corollary
cut-le-not-in-imp: $x \notin I \Longrightarrow x \notin I \downarrow\leq t$ **and**
cut-less-not-in-imp: $x \notin I \Longrightarrow x \notin I \downarrow< t$ **and**
cut-ge-not-in-imp: $x \notin I \Longrightarrow x \notin I \downarrow\geq t$ **and**
cut-greater-not-in-imp: $x \notin I \Longrightarrow x \notin I \downarrow> t$
$\langle proof \rangle$

lemmas *i-cut-not-in-imp* =
cut-le-not-in-imp cut-less-not-in-imp

cut-ge-not-in-imp cut-greater-not-in-imp

corollary
cut-le-in-imp: $x \in I \downarrow\leq t \Longrightarrow x \in I$ **and**
cut-less-in-imp: $x \in I \downarrow< t \Longrightarrow x \in I$ **and**
cut-ge-in-imp: $x \in I \downarrow\geq t \Longrightarrow x \in I$ **and**
cut-greater-in-imp: $x \in I \downarrow> t \Longrightarrow x \in I$
⟨*proof*⟩

lemmas *i-cut-in-imp* =
cut-le-in-imp cut-less-in-imp
cut-ge-in-imp cut-greater-in-imp

lemma *Collect-minI-cut*: $[\![k \in I; P\ (k::('a::wellorder))]\!] \Longrightarrow \exists x{\in}I.\ P\ x \wedge (\forall y{\in}(I \downarrow< x).\ \neg\ P\ y)$
⟨*proof*⟩

corollary *Collect-minI-ex-cut*: $\exists k{\in}I.\ P\ (k::('a::wellorder)) \Longrightarrow \exists x{\in}I.\ P\ x \wedge (\forall y{\in}(I \downarrow< x).\ \neg\ P\ y)$
⟨*proof*⟩

corollary *Collect-minI-ex2-cut*: $\{k{\in}I.\ P\ (k::('a::wellorder))\} \neq \{\} \Longrightarrow \exists x{\in}I.\ P\ x \wedge (\forall y{\in}(I \downarrow< x).\ \neg\ P\ y)$
⟨*proof*⟩

lemma *cut-le-cut-greater-ident*: $t2 \leq t1 \Longrightarrow I \downarrow\leq t1 \cup I \downarrow> t2 = I$
⟨*proof*⟩
lemma *cut-less-cut-ge-ident*: $t2 \leq t1 \Longrightarrow I \downarrow< t1 \cup I \downarrow\geq t2 = I$
⟨*proof*⟩
lemma *cut-le-cut-ge-ident*: $t2 \leq t1 \Longrightarrow I \downarrow\leq t1 \cup I \downarrow\geq t2 = I$
⟨*proof*⟩

lemma *cut-less-cut-greater-ident*:
$[\![t2 \leq t1; I \cap \{t1..t2\} = \{\}]\!] \Longrightarrow I \downarrow< t1 \cup I \downarrow> t2 = I$
⟨*proof*⟩
corollary *cut-less-cut-greater-ident'*:
$t \notin I \Longrightarrow I \downarrow< t \cup I \downarrow> t = I$
⟨*proof*⟩

lemma *insert-eq-cut-less-cut-greater*: $insert\ n\ I = I \downarrow< n \cup \{n\} \cup I \downarrow> n$
⟨*proof*⟩

7.2.3 Relations between cut operators

lemma *insert-Int-conv-if*: $A \cap (\mathit{insert}\ x\ B) = ($
$\mathit{if}\ x \in A\ \mathit{then}\ \mathit{insert}\ x\ (A \cap B)\ \mathit{else}\ A \cap B)$
$\langle\mathit{proof}\rangle$

lemma *cut-le-less-conv-if*: $I \downarrow\leq t = ($
$\mathit{if}\ t \in I\ \mathit{then}\ \mathit{insert}\ t\ (I \downarrow< t)\ \mathit{else}\ (I \downarrow< t))$
$\langle\mathit{proof}\rangle$

lemma *cut-le-less-conv*: $I \downarrow\leq t = (\{t\} \cap I) \cup (I \downarrow< t)$
$\langle\mathit{proof}\rangle$

lemma *cut-less-le-conv*: $I \downarrow< t = (I \downarrow\leq t) - \{t\}$
$\langle\mathit{proof}\rangle$
lemma *cut-less-le-conv-if*: $I \downarrow< t = ($
$\mathit{if}\ t \in I\ \mathit{then}\ (I \downarrow\leq t) - \{t\}\ \mathit{else}\ (I \downarrow\leq t))$
$\langle\mathit{proof}\rangle$

lemma *cut-ge-greater-conv-if*: $I \downarrow\geq t = ($
$\mathit{if}\ t \in I\ \mathit{then}\ \mathit{insert}\ t\ (I \downarrow> t)\ \mathit{else}\ (I \downarrow> t))$
$\langle\mathit{proof}\rangle$
lemma *cut-ge-greater-conv*: $I \downarrow\geq t = (\{t\} \cap I) \cup (I \downarrow> t)$
$\langle\mathit{proof}\rangle$
lemma *cut-greater-ge-conv*: $I \downarrow> t = (I \downarrow\geq t) - \{t\}$
$\langle\mathit{proof}\rangle$
lemma *cut-greater-ge-conv-if*: $I \downarrow> t = ($
$\mathit{if}\ t \in I\ \mathit{then}\ (I \downarrow\geq t) - \{t\}\ \mathit{else}\ (I \downarrow\geq t))$
$\langle\mathit{proof}\rangle$

lemma *nat-cut-le-less-conv*: $I \downarrow\leq t = I \downarrow< \mathit{Suc}\ t$
$\langle\mathit{proof}\rangle$
lemma *nat-cut-less-le-conv*: $0 < t \Longrightarrow I \downarrow< t = I \downarrow\leq (t - \mathit{Suc}\ 0)$
$\langle\mathit{proof}\rangle$
lemma *nat-cut-ge-greater-conv*: $I \downarrow\geq \mathit{Suc}\ t = I \downarrow> t$
$\langle\mathit{proof}\rangle$
lemma *nat-cut-greater-ge-conv*: $0 < t \Longrightarrow I \downarrow> (t - \mathit{Suc}\ 0) = I \downarrow\geq t$
$\langle\mathit{proof}\rangle$

7.2.4 Function images with cut operators

lemma *cut-less-image*:

$\llbracket$ *strict-mono-on f A*; $I \subseteq A$; $n \in A$ $\rrbracket \Longrightarrow$
$(f\ `\ I) \downarrow< f\ n = f\ `\ (I \downarrow< n)$
⟨*proof*⟩

lemma *cut-le-image*:
$\llbracket$ *strict-mono-on f A*; $I \subseteq A$; $n \in A$ $\rrbracket \Longrightarrow$
$(f\ `\ I) \downarrow\le f\ n = f\ `\ (I \downarrow\le n)$
⟨*proof*⟩

lemma *cut-greater-image*:
$\llbracket$ *strict-mono-on f A*; $I \subseteq A$; $n \in A$ $\rrbracket \Longrightarrow$
$(f\ `\ I) \downarrow> f\ n = f\ `\ (I \downarrow> n)$
⟨*proof*⟩

lemma *cut-ge-image*:
$\llbracket$ *strict-mono-on f A*; $I \subseteq A$; $n \in A$ $\rrbracket \Longrightarrow$
$(f\ `\ I) \downarrow\ge f\ n = f\ `\ (I \downarrow\ge n)$
⟨*proof*⟩

lemmas *i-cut-image* =
cut-le-image cut-less-image
cut-ge-image cut-greater-image

7.2.5 Finiteness and cardinality with cut operators

lemma
cut-le-finite: *finite I* $\Longrightarrow$ *finite* $(I \downarrow\le t)$ **and**
cut-less-finite: *finite I* $\Longrightarrow$ *finite* $(I \downarrow< t)$ **and**
cut-ge-finite: *finite I* $\Longrightarrow$ *finite* $(I \downarrow\ge t)$ **and**
cut-greater-finite: *finite I* $\Longrightarrow$ *finite* $(I \downarrow> t)$
⟨*proof*⟩

lemma *nat-cut-le-finite*: *finite* $(I \downarrow\le (t{::}nat))$
⟨*proof*⟩

lemma *nat-cut-less-finite*: *finite* $(I \downarrow< (t{::}nat))$
⟨*proof*⟩

lemma *nat-cut-ge-finite-iff*: *finite* $(I \downarrow\ge (t{::}nat))$ = *finite I*
⟨*proof*⟩

lemma *nat-cut-greater-finite-iff*: *finite* $(I \downarrow> (t{::}nat))$ = *finite I*
⟨*proof*⟩

lemma
cut-le-card: *finite I* $\Longrightarrow$ *card* $(I \downarrow\le t) \le$ *card I* **and**
cut-less-card: *finite I* $\Longrightarrow$ *card* $(I \downarrow< t) \le$ *card I* **and**
cut-ge-card: *finite I* $\Longrightarrow$ *card* $(I \downarrow\ge t) \le$ *card I* **and**
cut-greater-card: *finite I* $\Longrightarrow$ *card* $(I \downarrow> t) \le$ *card I*

$\langle proof \rangle$

lemma *nat-cut-greater-card*: $card\ (I \downarrow> (t::nat)) \le card\ I$
$\langle proof \rangle$

lemma *nat-cut-ge-card*: $card\ (I \downarrow\ge (t::nat)) \le card\ I$
$\langle proof \rangle$

7.2.6 Cutting a set at *Min* or *Max* element

lemma *cut-greater-Min-eq-Diff*: $I \downarrow> (iMin\ I) = I - \{iMin\ I\}$
$\langle proof \rangle$
lemma *cut-less-Max-eq-Diff*: $finite\ I \Longrightarrow I \downarrow< (Max\ I) = I - \{Max\ I\}$
$\langle proof \rangle$

lemma *cut-le-Min-empty*: $t < iMin\ I \Longrightarrow I \downarrow\le t = \{\}$
$\langle proof \rangle$
lemma *cut-less-Min-empty*: $t \le iMin\ I \Longrightarrow I \downarrow< t = \{\}$
$\langle proof \rangle$

lemma *cut-le-Min-not-empty*: $[\![\ I \ne \{\};\ iMin\ I \le t\ ]\!] \Longrightarrow I \downarrow\le t \ne \{\}$
$\langle proof \rangle$

lemma *cut-less-Min-not-empty*: $[\![\ I \ne \{\};\ iMin\ I < t\ ]\!] \Longrightarrow I \downarrow< t \ne \{\}$
$\langle proof \rangle$

lemma *cut-ge-Min-all*: $t \le iMin\ I \Longrightarrow I \downarrow\ge t = I$
$\langle proof \rangle$

lemma *cut-greater-Min-all*: $t < iMin\ I \Longrightarrow I \downarrow> t = I$
$\langle proof \rangle$

lemmas *i-cut-min-empty* =
  cut-le-Min-empty
  cut-less-Min-empty
  cut-le-Min-not-empty
  cut-less-Min-not-empty
lemmas *i-cut-min-all* =
  cut-ge-Min-all
  cut-greater-Min-all

lemma *cut-ge-Max-empty*: $finite\ I \Longrightarrow Max\ I < t \Longrightarrow I \downarrow\ge t = \{\}$
$\langle proof \rangle$

lemma *cut-greater-Max-empty*: $finite\ I \Longrightarrow Max\ I \le t \Longrightarrow I \downarrow> t = \{\}$
$\langle proof \rangle$

lemma *cut-ge-Max-not-empty*: $[\![\ I \ne \{\};\ finite\ I;\ t \le Max\ I\ ]\!] \Longrightarrow I \downarrow\ge t \ne \{\}$

$\langle proof \rangle$

lemma *cut-greater-Max-not-empty*: $[\![\ I \neq \{\};\ finite\ I;\ t < Max\ I\ ]\!] \Longrightarrow I \downarrow> t \neq \{\}$
$\langle proof \rangle$

lemma *cut-le-Max-all*: $finite\ I \Longrightarrow Max\ I \leq t \Longrightarrow I \downarrow\leq t = I$
$\langle proof \rangle$

lemma *cut-less-Max-all*: $finite\ I \Longrightarrow Max\ I < t \Longrightarrow I \downarrow< t = I$
$\langle proof \rangle$

lemmas *i-cut-max-empty* =
cut-ge-Max-empty
cut-greater-Max-empty
cut-ge-Max-not-empty
cut-greater-Max-not-empty
lemmas *i-cut-max-all* =
cut-le-Max-all
cut-less-Max-all

lemma *cut-less-Max-less*:
$[\![\ finite\ (I \downarrow< t);\ I \downarrow< t \neq \{\}\ ]\!] \Longrightarrow Max\ (I \downarrow< t) < t$
$\langle proof \rangle$
lemma *cut-le-Max-le*:
$[\![\ finite\ (I \downarrow\leq t);\ I \downarrow\leq t \neq \{\}\ ]\!] \Longrightarrow Max\ (I \downarrow\leq t) \leq t$
$\langle proof \rangle$
lemma *nat-cut-less-Max-less*:
$I \downarrow< t \neq \{\} \Longrightarrow Max\ (I \downarrow< t) < (t{::}nat)$
$\langle proof \rangle$
lemma *nat-cut-le-Max-le*:
$I \downarrow\leq t \neq \{\} \Longrightarrow Max\ (I \downarrow\leq t) \leq (t{::}nat)$
$\langle proof \rangle$
lemma *cut-greater-Min-greater*:
$I \downarrow> t \neq \{\} \Longrightarrow iMin\ (I \downarrow> t) > t$
$\langle proof \rangle$
lemma *cut-ge-Min-greater*:
$I \downarrow\geq t \neq \{\} \Longrightarrow iMin\ (I \downarrow\geq t) \geq t$
$\langle proof \rangle$

lemma *cut-less-Min-eq*: $I \downarrow< t \neq \{\} \Longrightarrow iMin\ (I \downarrow< t) = iMin\ I$
$\langle proof \rangle$

lemma *cut-le-Min-eq*: $I \downarrow\leq t \neq \{\} \Longrightarrow iMin\ (I \downarrow\leq t) = iMin\ I$
$\langle proof \rangle$

lemma *cut-ge-Max-eq*: $[\![$ *finite* I; $I \downarrow\geq t \neq \{\}$ $]\!] \Longrightarrow$ *Max* $(I \downarrow\geq t) =$ *Max* I
$\langle proof \rangle$

lemma *cut-greater-Max-eq*: $[\![$ *finite* I; $I \downarrow> t \neq \{\}$ $]\!] \Longrightarrow$ *Max* $(I \downarrow> t) =$ *Max* I
$\langle proof \rangle$

7.2.7 Cut operators with intervals from SetInterval

lemma
UNIV-cut-le: $UNIV \downarrow\leq t = \{..t\}$ **and**
UNIV-cut-less: $UNIV \downarrow< t = \{..<t\}$ **and**
UNIV-cut-ge: $UNIV \downarrow\geq t = \{t..\}$ **and**
UNIV-cut-greater: $UNIV \downarrow> t = \{t<..\}$
$\langle proof \rangle$

lemma
lessThan-cut-le: $\{..<n\} \downarrow\leq t = (\textit{if } n \leq t \textit{ then } \{..<n\} \textit{ else } \{..t\})$ **and**
lessThan-cut-less: $\{..<n\} \downarrow< t = (\textit{if } n \leq t \textit{ then } \{..<n\} \textit{ else } \{..<t\})$ **and**
lessThan-cut-ge: $\{..<n\} \downarrow\geq t = \{t..<n\}$ **and**
lessThan-cut-greater: $\{..<n\} \downarrow> t = \{t<..<n\}$ **and**
atMost-cut-le: $\{..n\} \downarrow\leq t = (\textit{if } n \leq t \textit{ then } \{..n\} \textit{ else } \{..t\})$ **and**
atMost-cut-less: $\{..n\} \downarrow< t = (\textit{if } n < t \textit{ then } \{..n\} \textit{ else } \{..<t\})$ **and**
atMost-cut-ge: $\{..n\} \downarrow\geq t = \{t..n\}$ **and**
atMost-cut-greager: $\{..n\} \downarrow> t = \{t<..n\}$ **and**
greaterThan-cut-le: $\{n<..\} \downarrow\leq t = \{n<..t\}$ **and**
greaterThan-cut-less: $\{n<..\} \downarrow< t = \{n<..<t\}$ **and**
greaterThan-cut-ge: $\{n<..\} \downarrow\geq t = (\textit{if } t \leq n \textit{ then } \{n<..\} \textit{ else } \{t..\})$ **and**
greaterThan-cut-greater: $\{n<..\} \downarrow> t = (\textit{if } t \leq n \textit{ then } \{n<..\} \textit{ else } \{t<..\})$ **and**
atLeast-cut-le: $\{n..\} \downarrow\leq t = \{n..t\}$ **and**
atLeast-cut-less: $\{n..\} \downarrow< t = \{n..<t\}$ **and**
atLeast-cut-ge: $\{n..\} \downarrow\geq t = (\textit{if } t \leq n \textit{ then } \{n..\} \textit{ else } \{t..\})$ **and**
atLeast-cut-greater: $\{n..\} \downarrow\geq t = (\textit{if } t \leq n \textit{ then } \{n..\} \textit{ else } \{t..\})$
$\langle proof \rangle$

lemma
greaterThanLessThan-cut-le: $\{m<..<n\} \downarrow\leq t = (\textit{if } n \leq t \textit{ then } \{m<..<n\} \textit{ else } \{m<..t\})$ **and**
greaterThanLessThan-cut-less: $\{m<..<n\} \downarrow< t = (\textit{if } n \leq t \textit{ then } \{m<..<n\} \textit{ else } \{m<..<t\})$ **and**
greaterThanLessThan-cut-ge: $\{m<..<n\} \downarrow\geq t = (\textit{if } t \leq m \textit{ then } \{m<..<n\} \textit{ else } \{t..<n\})$ **and**
greaterThanLessThan-cut-greater: $\{m<..<n\} \downarrow> t = (\textit{if } t \leq m \textit{ then } \{m<..<n\} \textit{ else } \{t<..<n\})$ **and**
atLeastLessThan-cut-le: $\{m..<n\} \downarrow\leq t = (\textit{if } n \leq t \textit{ then } \{m..<n\} \textit{ else } \{m..t\})$ **and**
atLeastLessThan-cut-less: $\{m..<n\} \downarrow< t = (\textit{if } n \leq t \textit{ then } \{m..<n\} \textit{ else } \{m..<t\})$ **and**
atLeastLessThan-cut-ge: $\{m..<n\} \downarrow\geq t = (\textit{if } t \leq m \textit{ then } \{m..<n\} \textit{ else } \{t..<n\})$

and
atLeastLessThan-cut-greater: $\{m..{<}n\} \downarrow> t = (\mathit{if}\ t < m\ \mathit{then}\ \{m..{<}n\}\ \mathit{else}\ \{t{<}..{<}n\})$ **and**
greaterThanAtMost-cut-le: $\{m{<}..n\} \downarrow\leq t = (\mathit{if}\ n \leq t\ \mathit{then}\ \{m{<}..n\}\ \mathit{else}\ \{m{<}..t\})$ **and**
greaterThanAtMost-cut-less: $\{m{<}..n\} \downarrow< t = (\mathit{if}\ n < t\ \mathit{then}\ \{m{<}..n\}\ \mathit{else}\ \{m{<}..{<}t\})$ **and**
greaterThanAtMost-cut-ge: $\{m{<}..n\} \downarrow\geq t = (\mathit{if}\ t \leq m\ \mathit{then}\ \{m{<}..n\}\ \mathit{else}\ \{t..n\})$ **and**
greaterThanAtMost-cut-greater: $\{m{<}..n\} \downarrow> t = (\mathit{if}\ t \leq m\ \mathit{then}\ \{m{<}..n\}\ \mathit{else}\ \{t{<}..n\})$ **and**
atLeastAtMost-cut-le: $\{m..n\} \downarrow\leq t = (\mathit{if}\ n \leq t\ \mathit{then}\ \{m..n\}\ \mathit{else}\ \{m..t\})$ **and**
atLeastAtMost-cut-less: $\{m..n\} \downarrow< t = (\mathit{if}\ n < t\ \mathit{then}\ \{m..n\}\ \mathit{else}\ \{m..{<}t\})$ **and**
atLeastAtMost-cut-ge: $\{m..n\} \downarrow\geq t = (\mathit{if}\ t \leq m\ \mathit{then}\ \{m..n\}\ \mathit{else}\ \{t..n\})$ **and**
atLeastAtMost-cut-greater: $\{m..n\} \downarrow> t = (\mathit{if}\ t < m\ \mathit{then}\ \{m..n\}\ \mathit{else}\ \{t{<}..n\})$
$\langle \mathit{proof} \rangle$

7.2.8 Mirroring finite natural sets between their *Min* and *Max* element

Mirroring a number at the middle of the interval min l r..max l r

Mirroring a single element n between the interval boundaries l and r

definition *nat-mirror* :: $\mathit{nat} \Rightarrow \mathit{nat} \Rightarrow \mathit{nat} \Rightarrow \mathit{nat}$
where $\mathit{nat\text{-}mirror}\ n\ l\ r \equiv l + r - n$

lemma *nat-mirror-commute*: $\mathit{nat\text{-}mirror}\ n\ l\ r = \mathit{nat\text{-}mirror}\ n\ r\ l$
$\langle \mathit{proof} \rangle$

lemma *nat-mirror-inj-on*: $\mathit{inj\text{-}on}\ (\lambda x.\ \mathit{nat\text{-}mirror}\ x\ l\ r)\ \{..l + r\}$
$\langle \mathit{proof} \rangle$

lemma *nat-mirror-nat-mirror-ident*:
$n \leq l + r \Longrightarrow \mathit{nat\text{-}mirror}\ (\mathit{nat\text{-}mirror}\ n\ l\ r)\ l\ r = n$
$\langle \mathit{proof} \rangle$

lemma *nat-mirror-add*:
$\mathit{nat\text{-}mirror}\ (n + k)\ l\ r = (\mathit{nat\text{-}mirror}\ n\ l\ r) - k$
$\langle \mathit{proof} \rangle$
lemma *nat-mirror-diff*:
$[\![\ k \leq n;\ n \leq l + r\ ]\!] \Longrightarrow$
$\mathit{nat\text{-}mirror}\ (n - k)\ l\ r = (\mathit{nat\text{-}mirror}\ n\ l\ r) + k$
$\langle \mathit{proof} \rangle$

lemma *nat-mirror-le*: $a \leq b \Longrightarrow \mathit{nat\text{-}mirror}\ b\ l\ r \leq \mathit{nat\text{-}mirror}\ a\ l\ r$
$\langle \mathit{proof} \rangle$
lemma *nat-mirror-le-conv*:

$a \le l + r \Longrightarrow (\textit{nat-mirror } b\ l\ r \le \textit{nat-mirror } a\ l\ r) = (a \le b)$
⟨*proof*⟩

lemma *nat-mirror-less*:
$[\![\ a < b;\ a < l + r\ ]\!] \Longrightarrow$
$\textit{nat-mirror } b\ l\ r < \textit{nat-mirror } a\ l\ r$
⟨*proof*⟩
lemma *nat-mirror-less-imp-less*:
$\textit{nat-mirror } b\ l\ r < \textit{nat-mirror } a\ l\ r \Longrightarrow a < b$
⟨*proof*⟩
lemma *nat-mirror-less-conv*:
$a < l + r \Longrightarrow (\textit{nat-mirror } b\ l\ r < \textit{nat-mirror } a\ l\ r) = (a < b)$
⟨*proof*⟩

lemma *nat-mirror-eq-conv*:
$[\![\ a \le l + r;\ b \le l + r\ ]\!] \Longrightarrow$
$(\textit{nat-mirror } a\ l\ r = \textit{nat-mirror } b\ l\ r) = (a = b)$
⟨*proof*⟩

Mirroring a single element n between the interval boundaries of I

definition
$\textit{mirror-elem} :: \textit{nat} \Rightarrow \textit{nat set} \Rightarrow \textit{nat}$
where
$\textit{mirror-elem } n\ I \equiv \textit{nat-mirror } n\ (\textit{iMin } I)\ (\textit{Max } I)$

lemma *mirror-elem-inj-on*: $\textit{finite } I \Longrightarrow \textit{inj-on } (\lambda x.\ \textit{mirror-elem } x\ I)\ I$
⟨*proof*⟩

lemma *mirror-elem-add*:
$\textit{finite } I \Longrightarrow \textit{mirror-elem } (n + k)\ I = \textit{mirror-elem } n\ I - k$
⟨*proof*⟩
lemma *mirror-elem-diff*:
$[\![\ \textit{finite } I;\ k \le n;\ n \in I\ ]\!] \Longrightarrow \textit{mirror-elem } (n - k)\ I = \textit{mirror-elem } n\ I + k$
⟨*proof*⟩
lemma *mirror-elem-Min*:
$[\![\ \textit{finite } I;\ I \ne \{\}\ ]\!] \Longrightarrow \textit{mirror-elem } (\textit{iMin } I)\ I = \textit{Max } I$
⟨*proof*⟩
lemma *mirror-elem-Max*:
$[\![\ \textit{finite } I;\ I \ne \{\}\ ]\!] \Longrightarrow \textit{mirror-elem } (\textit{Max } I)\ I = \textit{iMin } I$
⟨*proof*⟩

lemma *mirror-elem-mirror-elem-ident*:
$[\![\ \textit{finite } I;\ n \le \textit{iMin } I + \textit{Max } I\ ]\!] \Longrightarrow \textit{mirror-elem } (\textit{mirror-elem } n\ I)\ I = n$
⟨*proof*⟩

lemma *mirror-elem-le-conv*:
$[\![\ \textit{finite } I;\ a \in I;\ b \in I\ ]\!] \Longrightarrow$
$(\textit{mirror-elem } b\ I \le \textit{mirror-elem } a\ I) = (a \le b)$
⟨*proof*⟩

lemma *mirror-elem-less-conv*:
$[\![$ *finite* I; $a \in I$; $b \in I$ $]\!] \Longrightarrow$
$(\mathit{mirror\text{-}elem}\ b\ I < \mathit{mirror\text{-}elem}\ a\ I) = (a < b)$
⟨*proof*⟩

lemma *mirror-elem-eq-conv*:
$[\![$ $a \le \mathit{iMin}\ I + \mathit{Max}\ I$; $b \le \mathit{iMin}\ I + \mathit{Max}\ I$ $]\!] \Longrightarrow$
$(\mathit{mirror\text{-}elem}\ a\ I = \mathit{mirror\text{-}elem}\ b\ I) = (a = b)$
⟨*proof*⟩
lemma *mirror-elem-eq-conv'*:
$[\![$ *finite* I; $a \in I$; $b \in I$ $]\!] \Longrightarrow (\mathit{mirror\text{-}elem}\ a\ I = \mathit{mirror\text{-}elem}\ b\ I) = (a = b)$
⟨*proof*⟩

definition
imirror-bounds :: *nat set* ⇒ *nat* ⇒ *nat* ⇒ *nat set*
where
$\mathit{imirror\text{-}bounds}\ I\ l\ r \equiv (\lambda x.\ \mathit{nat\text{-}mirror}\ x\ l\ r)$ ‘ I

Mirroring all elements between the interval boundaries of I

definition
imirror :: *nat set* ⇒ *nat set*
where
$\mathit{imirror}\ I \equiv (\lambda x.\ \mathit{iMin}\ I + \mathit{Max}\ I - x)$ ‘ I

lemma *imirror-eq-nat-mirror-image*:
$\mathit{imirror}\ I = (\lambda x.\ \mathit{nat\text{-}mirror}\ x\ (\mathit{iMin}\ I)\ (\mathit{Max}\ I))$ ‘ I
⟨*proof*⟩
lemma *imirror-eq-mirror-elem-image*:
$\mathit{imirror}\ I = (\lambda x.\ \mathit{mirror\text{-}elem}\ x\ I)$ ‘ I
⟨*proof*⟩

lemma *imirror-eq-imirror-bounds*:
$\mathit{imirror}\ I = \mathit{imirror\text{-}bounds}\ I\ (\mathit{iMin}\ I)\ (\mathit{Max}\ I)$
⟨*proof*⟩

lemma *imirror-empty*: $\mathit{imirror}\ \{\} = \{\}$
⟨*proof*⟩
lemma *imirror-is-empty*: $(\mathit{imirror}\ I = \{\}) = (I = \{\})$
⟨*proof*⟩
lemma *imirror-not-empty*: $I \neq \{\} \Longrightarrow \mathit{imirror}\ I \neq \{\}$
⟨*proof*⟩

lemma *imirror-singleton*: $\mathit{imirror}\ \{a\} = \{a\}$
⟨*proof*⟩

lemma *imirror-finite*: $finite\ I \Longrightarrow finite\ (imirror\ I)$
⟨*proof*⟩

lemma *imirror-bounds-iMin*:
$[\![\ finite\ I;\ I \neq \{\};\ iMin\ I \leq l + r\ ]\!] \Longrightarrow$
$iMin\ (imirror\text{-}bounds\ I\ l\ r) = l + r - Max\ I$
⟨*proof*⟩

lemma *imirror-bounds-Max*:
$[\![\ finite\ I;\ I \neq \{\};\ Max\ I \leq l + r\ ]\!] \Longrightarrow$
$Max\ (imirror\text{-}bounds\ I\ l\ r) = l + r - iMin\ I$
⟨*proof*⟩

lemma *imirror-iMin*: $finite\ I \Longrightarrow iMin\ (imirror\ I) = iMin\ I$
⟨*proof*⟩

lemma *imirror-Max*: $finite\ I \Longrightarrow Max\ (imirror\ I) = Max\ I$
⟨*proof*⟩
corollary *imirror-iMin-Max*: $[\![\ finite\ I;\ n \in imirror\ I\ ]\!] \Longrightarrow iMin\ I \leq n \wedge n \leq Max\ I$
⟨*proof*⟩

lemma *imirror-bounds-iff*:
$(n \in imirror\text{-}bounds\ I\ l\ r) = (\exists\, x{\in}I.\ n = l + r - x)$
⟨*proof*⟩

lemma *imirror-iff*: $(n \in imirror\ I) = (\exists\, x{\in}I.\ n = iMin\ I + Max\ I - x)$
⟨*proof*⟩

lemma *imirror-bounds-imirror-bounds-ident*:
$[\![\ finite\ I;\ Max\ I \leq l + r\ ]\!] \Longrightarrow$
$imirror\text{-}bounds\ (imirror\text{-}bounds\ I\ l\ r)\ l\ r = I$
⟨*proof*⟩

lemma *imirror-imirror-ident*: $finite\ I \Longrightarrow imirror\ (imirror\ I) = I$
⟨*proof*⟩

lemma *mirror-elem-imirror*:
$finite\ I \Longrightarrow mirror\text{-}elem\ t\ (imirror\ I) = mirror\text{-}elem\ t\ I$
⟨*proof*⟩

lemma *imirror-card*: $finite\ I \Longrightarrow card\ (imirror\ I) = card\ I$
⟨*proof*⟩

lemma *imirror-bounds-elem-conv*:
$[\![\ finite\ I;\ n \leq l + r;\ Max\ I \leq l + r\ ]\!] \Longrightarrow$

$((\mathit{nat\text{-}mirror}\ n\ l\ r) \in \mathit{imirror\text{-}bounds}\ I\ l\ r) = (n \in I)$
$\langle \mathit{proof} \rangle$

lemma *imirror-mem-conv*:
$[\![\ \mathit{finite}\ I;\ n \leq \mathit{iMin}\ I + \mathit{Max}\ I\ ]\!] \Longrightarrow ((\mathit{mirror\text{-}elem}\ n\ I) \in \mathit{imirror}\ I) = (n \in I)$
$\langle \mathit{proof} \rangle$

corollary *in-imp-mirror-elem-in*:
$[\![\ \mathit{finite}\ I;\ n \in I\ ]\!] \Longrightarrow (\mathit{mirror\text{-}elem}\ n\ I) \in \mathit{imirror}\ I$
$\langle \mathit{proof} \rangle$

lemma *imirror-cut-less*:
$\mathit{finite}\ I \Longrightarrow$
$\mathit{imirror}\ I \downarrow< (\mathit{mirror\text{-}elem}\ t\ I) =$
$\mathit{imirror\text{-}bounds}\ (I \downarrow> t)\ (\mathit{iMin}\ I)\ (\mathit{Max}\ I)$
$\langle \mathit{proof} \rangle$

lemma *imirror-cut-le*:
$[\![\ \mathit{finite}\ I;\ t \leq \mathit{iMin}\ I + \mathit{Max}\ I\ ]\!] \Longrightarrow$
$\mathit{imirror}\ I \downarrow\leq (\mathit{mirror\text{-}elem}\ t\ I) =$
$\mathit{imirror\text{-}bounds}\ (I \downarrow\geq t)\ (\mathit{iMin}\ I)\ (\mathit{Max}\ I)$
$\langle \mathit{proof} \rangle$

lemma *imirror-cut-ge*:
$\mathit{finite}\ I \Longrightarrow$
$\mathit{imirror}\ I \downarrow\geq (\mathit{mirror\text{-}elem}\ t\ I) =$
$\mathit{imirror\text{-}bounds}\ (I \downarrow\leq t)\ (\mathit{iMin}\ I)\ (\mathit{Max}\ I)$
(**is** $?P \Longrightarrow ?\mathit{lhs}\ I = ?\mathit{rhs}\ I\ t$)
$\langle \mathit{proof} \rangle$

lemma *imirror-cut-greater*: $[\![\ \mathit{finite}\ I;\ t \leq \mathit{iMin}\ I + \mathit{Max}\ I\ ]\!] \Longrightarrow$
$\mathit{imirror}\ I \downarrow> (\mathit{mirror\text{-}elem}\ t\ I) =$
$\mathit{imirror\text{-}bounds}\ (I \downarrow< t)\ (\mathit{iMin}\ I)\ (\mathit{Max}\ I)$
$\langle \mathit{proof} \rangle$

lemmas *imirror-cut* =
imirror-cut-less imirror-cut-ge
imirror-cut-le imirror-cut-greater

corollary *imirror-cut-le′*:
$[\![\ \mathit{finite}\ I;\ t \in I\ ]\!] \Longrightarrow$
$\mathit{imirror}\ I \downarrow\leq \mathit{mirror\text{-}elem}\ t\ I =$
$\mathit{imirror\text{-}bounds}\ (I \downarrow\geq t)\ (\mathit{iMin}\ I)\ (\mathit{Max}\ I)$
$\langle \mathit{proof} \rangle$

corollary *imirror-cut-greater′*:
$[\![\ \mathit{finite}\ I;\ t \in I\ ]\!] \Longrightarrow$
$\mathit{imirror}\ I \downarrow> \mathit{mirror\text{-}elem}\ t\ I =$
$\mathit{imirror\text{-}bounds}\ (I \downarrow< t)\ (\mathit{iMin}\ I)\ (\mathit{Max}\ I)$

$\langle proof \rangle$

lemmas *imirror-cut′* =
  imirror-cut-le′ imirror-cut-greater′

lemma *imirror-bounds-Un*:
  imirror-bounds $(A \cup B)$ *l r* =
  imirror-bounds A l r $\cup$ *imirror-bounds B l r*
$\langle proof \rangle$
lemma *imirror-bounds-Int*:
  $[\![A \subseteq \{..l + r\}; B \subseteq \{..l + r\}]\!] \Longrightarrow$
  imirror-bounds $(A \cap B)$ *l r* =
  imirror-bounds A l r $\cap$ *imirror-bounds B l r*
$\langle proof \rangle$

end

8 Stepping through sets of natural numbers

theory *SetIntervalStep*
imports *SetIntervalCut*
begin

8.1 Function *inext* and *iprev* for stepping through natural sets

definition *inext* :: *nat* $\Rightarrow$ *nat set* $\Rightarrow$ *nat*
where
  inext n I $\equiv$ (
    if $(n \in I \wedge (I \downarrow> n \neq \{\}))$
    then iMin $(I \downarrow> n)$
    else n)

definition *iprev* :: *nat* $\Rightarrow$ *nat set* $\Rightarrow$ *nat*
where
  iprev n I $\equiv$ (
    if $(n \in I \wedge (I \downarrow< n \neq \{\}))$
    then Max $(I \downarrow< n)$
    else n)

inext and *iprev* can be viewed as generalisations of *Suc* and *prev*

lemma *inext-UNIV*: *inext n UNIV* = *Suc n*
$\langle proof \rangle$
lemma *iprev-UNIV*: *iprev n UNIV* = $n -$ *Suc 0*
$\langle proof \rangle$

lemma *inext-empty*: *inext n* $\{\} = n$
$\langle proof \rangle$
lemma *iprev-empty*: *iprev n* $\{\} = n$

⟨*proof*⟩

lemma *not-in-inext-fix*: $n \notin I \Longrightarrow inext\ n\ I = n$
⟨*proof*⟩
lemma *not-in-iprev-fix*: $n \notin I \Longrightarrow iprev\ n\ I = n$
⟨*proof*⟩

lemma *inext-all-le-fix*: $\forall x{\in}I.\ x \le n \Longrightarrow inext\ n\ I = n$
⟨*proof*⟩
lemma *iprev-all-ge-fix*: $\forall x{\in}I.\ n \le x \Longrightarrow iprev\ n\ I = n$
⟨*proof*⟩

lemma *inext-Max*: $finite\ I \Longrightarrow inext\ (Max\ I)\ I = Max\ I$
⟨*proof*⟩
lemma *iprev-iMin*: $iprev\ (iMin\ I)\ I = iMin\ I$
⟨*proof*⟩

lemma *inext-ge-Max*: $[\![\ finite\ I;\ Max\ I \le n\ ]\!] \Longrightarrow inext\ n\ I = n$
⟨*proof*⟩

lemma *iprev-le-iMin*: $n \le iMin\ I \Longrightarrow iprev\ n\ I = n$
⟨*proof*⟩

lemma *inext-singleton*: $inext\ n\ \{a\} = n$
⟨*proof*⟩

lemma *iprev-singleton*: $iprev\ n\ \{a\} = n$
⟨*proof*⟩

lemma *inext-closed*: $n \in I \Longrightarrow inext\ n\ I \in I$
⟨*proof*⟩

lemma *iprev-closed*: $n \in I \Longrightarrow iprev\ n\ I \in I$
⟨*proof*⟩

lemma *inext-in-imp-in*: $inext\ n\ I \in I \Longrightarrow n \in I$
⟨*proof*⟩

lemma *inext-in-iff*: $(inext\ n\ I \in I) = (n \in I)$
⟨*proof*⟩

lemma *subset-inext-closed*: $[\![\ n \in B;\ A \subseteq B\ ]\!] \Longrightarrow inext\ n\ A \in B$
⟨*proof*⟩
lemma *subset-inext-in-imp-in*: $[\![\ inext\ n\ A \in B;\ A \subseteq B\ ]\!] \Longrightarrow n \in B$
⟨*proof*⟩
lemma *subset-inext-in-iff*: $A \subseteq B \Longrightarrow (inext\ n\ A \in B) = (n \in B)$
⟨*proof*⟩

lemma *iprev-in-imp-in*: $iprev\ n\ I \in I \Longrightarrow n \in I$
⟨*proof*⟩

lemma *iprev-in-iff*: $(iprev\ n\ I \in I) = (n \in I)$
⟨*proof*⟩

lemma *subset-iprev-closed*: $[\![\ n \in B;\ A \subseteq B\ ]\!] \Longrightarrow iprev\ n\ A \in B$
⟨*proof*⟩

lemma *subset-iprev-in-imp-in*: $[\![\ iprev\ n\ A \in B;\ A \subseteq B\ ]\!] \Longrightarrow n \in B$
⟨*proof*⟩

lemma *subset-iprev-in-iff*: $A \subseteq B \Longrightarrow (iprev\ n\ A \in B) = (n \in B)$
⟨*proof*⟩

lemma *inext-mono*: $n \le inext\ n\ I$
⟨*proof*⟩

corollary *inext-neq-imp-less*: $n \neq inext\ n\ I \Longrightarrow n < inext\ n\ I$
⟨*proof*⟩

lemma *inext-mono2*: $[\![\ n \in I;\ \exists\, x{\in}I.\ n < x\ ]\!] \Longrightarrow n < inext\ n\ I$
⟨*proof*⟩

lemma *inext-mono2-infin*: $[\![\ n \in I;\ infinite\ I\ ]\!] \Longrightarrow n < inext\ n\ I$
⟨*proof*⟩

lemma *inext-mono2-fin*: $[\![\ n \in I;\ finite\ I;\ n \neq Max\ I\ ]\!] \Longrightarrow n < inext\ n\ I$
⟨*proof*⟩

lemma *inext-mono2-infin-fin*:
$[\![\ n \in I;\ n \neq Max\ I \vee infinite\ I\ ]\!] \Longrightarrow n < inext\ n\ I$
⟨*proof*⟩

lemma *inext-neq-iMin*: $\exists\, x{\in}I.\ n < x \Longrightarrow inext\ n\ I \neq iMin\ I$
⟨*proof*⟩

lemma *inext-neq-iMin-infin*: $infinite\ I \Longrightarrow inext\ n\ I \neq iMin\ I$
⟨*proof*⟩

lemma *Max-le-iMin-imp-singleton*: $[\![\ finite\ I;\ I \neq \{\};\ Max\ I \le iMin\ I\ ]\!] \Longrightarrow I = \{iMin\ I\}$
⟨*proof*⟩

lemma *inext-neq-iMin-not-singleton*:
$[\![\ I \neq \{\};\ \neg(\exists\, a.\ I = \{a\})\ ]\!] \Longrightarrow inext\ n\ I \neq iMin\ I$
⟨*proof*⟩
corollary *inext-neq-iMin-not-card-1*:
$[\![\ I \neq \{\};\ card\ I \neq Suc\ 0\ ]\!] \Longrightarrow inext\ n\ I \neq iMin\ I$

$\langle proof \rangle$

lemma *inext-neq-imp-Max*: $n \neq inext\ n\ I \Longrightarrow n < Max\ I \vee infinite\ I$
$\langle proof \rangle$

lemma *inext-less-conv*: $(n \in I \wedge (n < Max\ I \vee infinite\ I)) = (n < inext\ n\ I)$
$\langle proof \rangle$

lemma *inext-min-step*: $[\![\ n < k;\ k < inext\ n\ I\ ]\!] \Longrightarrow k \notin I$
$\langle proof \rangle$

corollary *inext-min-step2*: $\neg(\exists\, k{\in}I.\ n < k \wedge k < inext\ n\ I)$
$\langle proof \rangle$

lemma *min-step-inext*[*rule-format*]:
$[\![\ x < y;\ x \in I;\ y \in I;\ \bigwedge k.\ [\![\ x < k;\ k < y\ ]\!] \Longrightarrow k \notin I\ ]\!] \Longrightarrow$
$inext\ x\ I = y$
$\langle proof \rangle$

corollary *min-step-inext2*[*rule-format*]:
$[\![\ x < y;\ x \in I;\ y \in I;\ \neg(\exists\, k \in I.\ x < k \wedge k < y)\ ]\!] \Longrightarrow$
$inext\ x\ I = y$
$\langle proof \rangle$
lemma *between-empty-imp-inext-eq*:
$[\![\ n \in A;\ n < inext\ n\ A;\ n \in B;\ inext\ n\ A \in B;\ B \downarrow> n \downarrow< (inext\ n\ A) = \{\}\ ]\!]$
$\Longrightarrow$
$inext\ n\ B = inext\ n\ A$
$\langle proof \rangle$

lemma *inext-le-mono*: $[\![\ a \le b;\ a \in I;\ b \in I\ ]\!] \Longrightarrow inext\ a\ I \le inext\ b\ I$
$\langle proof \rangle$

lemma *inext-less-mono*:
$[\![\ a < b;\ a \in I;\ b \in I;\ \exists x{\in}I.\ b < x\ ]\!] \Longrightarrow inext\ a\ I < inext\ b\ I$
$\langle proof \rangle$

lemma *inext-less-mono-fin*:
$[\![\ a < b;\ a \in I;\ b \in I;\ finite\ I;\ b \neq Max\ I\ ]\!] \Longrightarrow inext\ a\ I < inext\ b\ I$
$\langle proof \rangle$

lemma *inext-less-mono-infin*:
$[\![\ a < b;\ a \in I;\ b \in I;\ infinite\ I\ ]\!] \Longrightarrow inext\ a\ I < inext\ b\ I$
$\langle proof \rangle$

lemma *inext-less-mono-infin-fin*:
$[\![\ a < b;\ a \in I;\ b \in I;\ b \neq Max\ I \vee infinite\ I\ ]\!] \Longrightarrow inext\ a\ I < inext\ b\ I$

$\langle proof \rangle$

lemma *inext-le-mono-rev*:
$[\![$ *inext a I* $\le$ *inext b I*; $a \in I$; $b \in I$; $\exists x \in I$. *inext a I* $< x$ $]\!] \Longrightarrow a \le b$
$\langle proof \rangle$

lemma *inext-le-mono-fin-rev*:
$[\![$ *inext a I* $\le$ *inext b I*; $a \in I$; $b \in I$; *finite I*; *inext a I* $\neq$ *Max I*$]\!] \Longrightarrow a \le b$
$\langle proof \rangle$

lemma *inext-le-mono-infin-rev*:
$[\![$ *inext a I* $\le$ *inext b I*; $a \in I$; $b \in I$; *infinite I* $]\!] \Longrightarrow a \le b$
$\langle proof \rangle$

lemma *inext-le-mono-infin-fin-rev*:
$[\![$ *inext a I* $\le$ *inext b I*; $a \in I$; $b \in I$; *inext a I* $\neq$ *Max I* $\vee$ *infinite I* $]\!] \Longrightarrow a \le b$
$\langle proof \rangle$

lemma *inext-less-mono-rev*:
$[\![$ *inext a I* $<$ *inext b I*; $a \in I$; $b \in I$ $]\!] \Longrightarrow a < b$
$\langle proof \rangle$

lemma *less-imp-inext-le*: $[\![$ $a < b$; $a \in I$; $b \in I$ $]\!] \Longrightarrow$ *inext a I* $\le b$
$\langle proof \rangle$

lemma *iprev-mono*: *iprev n I* $\le n$
$\langle proof \rangle$
corollary *iprev-neq-imp-greater*: $n \neq$ *iprev n I* $\Longrightarrow$ *iprev n I* $< n$
$\langle proof \rangle$

lemma *iprev-mono2*: $[\![$ $n \in I$; $\exists x \in I$. $x < n]\!] \Longrightarrow$ *iprev n I* $< n$
$\langle proof \rangle$

lemma *iprev-mono2-if-neq-iMin*: $[\![$ $n \in I$; *iMin I* $\neq n]\!] \Longrightarrow$ *iprev n I* $< n$
$\langle proof \rangle$

lemma *iprev-neq-Max*: $[\![$ *finite I*; $\exists x \in I$. $x < n$ $]\!] \Longrightarrow$ *iprev n I* $\neq$ *Max I*
$\langle proof \rangle$

lemma *iprev-neq-Max-not-singleton*:
$[\![$ *finite I*; $I \neq \{\}$; $\neg(\exists a.\ I = \{a\})$ $]\!] \Longrightarrow$ *iprev n I* $\neq$ *Max I*
$\langle proof \rangle$
corollary *iprev-neq-Max-not-card-1*:
$[\![$ *finite I*; $I \neq \{\}$; *card I* $\neq$ *Suc 0* $]\!] \Longrightarrow$ *iprev n I* $\neq$ *Max I*
$\langle proof \rangle$

lemma *iprev-neq-imp-iMin*: *iprev n I* $\neq n \Longrightarrow$ *iMin I* $< n$
$\langle proof \rangle$

lemma *iprev-greater-conv*: $(n \in I \wedge iMin\ I < n) = (iprev\ n\ I < n)$
$\langle proof \rangle$

lemma *inext-fix-iff*: $(n \notin I \vee (finite\ I \wedge Max\ I = n)) = (inext\ n\ I = n)$
$\langle proof \rangle$

lemma *iprev-fix-iff*: $(n \notin I \vee iMin\ I = n) = (iprev\ n\ I = n)$
$\langle proof \rangle$

lemma *iprev-min-step*: $[\![\ iprev\ n\ I < k;\ k < n\ ]\!] \Longrightarrow k \notin I$
$\langle proof \rangle$

corollary *iprev-min-step2*: $\neg(\exists\, x{\in}I.\ iprev\ n\ I < x \wedge x < n)$
$\langle proof \rangle$

lemma *min-step-iprev*:
$[\![\ x < y;\ x \in I;\ y \in I;\ \bigwedge k.\ [\![\ x < k;\ k < y\ ]\!] \Longrightarrow k \notin I\ ]\!] \Longrightarrow$
$iprev\ y\ I = x$
$\langle proof \rangle$

corollary *min-step-iprev2*[*rule-format*]:
$[\![\ x < y;\ x \in I;\ y \in I;\ \neg(\exists\, k \in I.\ x < k \wedge k < y)\ ]\!] \Longrightarrow$
$iprev\ y\ I = x$
$\langle proof \rangle$

lemma *between-empty-imp-iprev-eq*:
$[\![\ n \in A;\ iprev\ n\ A < n;\ n \in B;\ iprev\ n\ A \in B;\ B \downarrow\!> (iprev\ n\ A) \downarrow\!< n = \{\}\ ]\!]$
$\Longrightarrow$
$iprev\ n\ B = iprev\ n\ A$
$\langle proof \rangle$

lemma *iprev-le-mono*: $[\![\ a \le b;\ a \in I;\ b \in I\ ]\!] \Longrightarrow iprev\ a\ I \le iprev\ b\ I$
$\langle proof \rangle$

lemma *iprev-less-mono*:
$[\![\ a < b;\ a \in I;\ b \in I;\ \exists\, x{\in}I.\ x < a\ ]\!] \Longrightarrow iprev\ a\ I < iprev\ b\ I$
$\langle proof \rangle$

lemma *iprev-less-mono-if-neq-iMin*:
$[\![\ a < b;\ a \in I;\ b \in I;\ iMin\ I \neq a\ ]\!] \Longrightarrow iprev\ a\ I < iprev\ b\ I$
$\langle proof \rangle$

lemma *iprev-le-mono-rev*:
$[\![\ iprev\ a\ I \le iprev\ b\ I;\ a \in I;\ b \in I;\ iMin\ I \neq iprev\ b\ I\ ]\!] \Longrightarrow a \le b$
$\langle proof \rangle$

lemma *iprev-less-mono-rev*:
$[\![$ *iprev a I* $<$ *iprev b I*; $a \in I$; $b \in I$ $]\!] \Longrightarrow a < b$
$\langle proof \rangle$

lemma *set-restriction-inext-eq*:
$[\![$ *set-restriction interval-fun*; $n \in$ *interval-fun I*; *inext n I* $\in$ *interval-fun I* $]\!] \Longrightarrow$
inext n (*interval-fun I*) $=$ *inext n I*
$\langle proof \rangle$

lemma *set-restriction-inext-singleton-eq*:
$[\![$ *set-restriction interval-fun*; $n \in$ *interval-fun I*; *inext n I* $\in$ *interval-fun I* $]\!] \Longrightarrow$
$\{$*inext n* (*interval-fun I*)$\}$ $=$ *interval-fun* $\{$*inext n I*$\}$
$\langle proof \rangle$

lemma *inext-iprev*: *iMin I* $\neq n \Longrightarrow$ *inext* (*iprev n I*) $I = n$
$\langle proof \rangle$

lemma *iprev-inext-infin*: *infinite I* $\Longrightarrow$ *iprev* (*inext n I*) $I = n$
$\langle proof \rangle$

lemma *iprev-inext-fin*:
$[\![$ *finite I*; $n \neq$ *Max I* $]\!] \Longrightarrow$ *iprev* (*inext n I*) $I = n$
$\langle proof \rangle$

lemma *iprev-inext*:
$n \neq$ *Max I* $\vee$ *infinite I* $\Longrightarrow$ *iprev* (*inext n I*) $I = n$
$\langle proof \rangle$

lemma *inext-eq-infin*:
$[\![$ *inext a I* $=$ *inext b I*; *infinite I* $]\!] \Longrightarrow a = b$
$\langle proof \rangle$

lemma *inext-eq-fin*:
$[\![$ *inext a I* $=$ *inext b I*; *finite I*; $a \neq$ *Max I*; $b \neq$ *Max I* $]\!] \Longrightarrow a = b$
$\langle proof \rangle$

lemma *inext-eq-infin-fin*:
$[\![$ *inext a I* $=$ *inext b I*; $a \neq$ *Max I* $\wedge$ $b \neq$ *Max I* $\vee$ *infinite I* $]\!] \Longrightarrow a = b$
$\langle proof \rangle$

lemma *inext-eq*:
$[\![$ *inext a I* $=$ *inext b I*; $\exists x \in I.\ a < x$; $\exists x \in I.\ b < x$ $]\!] \Longrightarrow a = b$
$\langle proof \rangle$

lemma *iprev-eq-if-neq-iMin*:

$\llbracket$ *iprev a I* = *iprev b I*; *iMin I* $\neq$ *a*; *iMin I* $\neq$ *b* $\rrbracket \Longrightarrow$ *a* = *b*
⟨*proof*⟩

lemma *iprev-eq*:
$\llbracket$ *iprev a I* = *iprev b I*; $\exists x{\in}I.\ x < a$; $\exists x{\in}I.\ x < b$ $\rrbracket \Longrightarrow$ *a* = *b*
⟨*proof*⟩

lemma *greater-imp-iprev-ge*: $\llbracket$ *b* < *a*; *a* ∈ *I*; *b* ∈ *I* $\rrbracket \Longrightarrow$ *b* ≤ *iprev a I*
⟨*proof*⟩

lemma *inext-cut-less-conv*: *inext n I* < *t* $\Longrightarrow$ *inext n* (*I* ↓< *t*) = *inext n I*
⟨*proof*⟩

lemma *inext-cut-le-conv*: *inext n I* ≤ *t* $\Longrightarrow$ *inext n* (*I* ↓≤ *t*) = *inext n I*
⟨*proof*⟩

lemma *inext-cut-greater-conv*: *t* < *n* $\Longrightarrow$ *inext n* (*I* ↓> *t*) = *inext n I*
⟨*proof*⟩

lemma *inext-cut-ge-conv*: *t* ≤ *n* $\Longrightarrow$ *inext n* (*I* ↓≥ *t*) = *inext n I*
⟨*proof*⟩

lemmas *inext-cut-conv* =
inext-cut-less-conv inext-cut-le-conv
inext-cut-greater-conv inext-cut-ge-conv

lemma *iprev-cut-greater-conv*: *t* < *iprev n I* $\Longrightarrow$ *iprev n* (*I* ↓> *t*) = *iprev n I*
⟨*proof*⟩

lemma *iprev-cut-ge-conv*: *t* ≤ *iprev n I* $\Longrightarrow$ *iprev n* (*I* ↓≥ *t*) = *iprev n I*
⟨*proof*⟩

lemma *iprev-cut-less-conv*: *n* < *t* $\Longrightarrow$ *iprev n* (*I* ↓< *t*) = *iprev n I*
⟨*proof*⟩

lemma *iprev-cut-le-conv*: *n* ≤ *t* $\Longrightarrow$ *iprev n* (*I* ↓≤ *t*) = *iprev n I*
⟨*proof*⟩

lemmas *iprev-cut-conv* =
iprev-cut-less-conv iprev-cut-le-conv
iprev-cut-greater-conv iprev-cut-ge-conv

lemma *inext-cut-less-fix*: *t* ≤ *inext n I* $\Longrightarrow$ *inext n* (*I* ↓< *t*) = *n*
⟨*proof*⟩

lemma *inext-cut-le-fix*: *t* < *inext n I* $\Longrightarrow$ *inext n* (*I* ↓≤ *t*) = *n*

⟨*proof*⟩

lemma *iprev-cut-greater-fix*: $iprev\ n\ I \le t \Longrightarrow iprev\ n\ (I \downarrow> t) = n$
⟨*proof*⟩

lemma *iprev-cut-ge-fix*: $iprev\ n\ I < t \Longrightarrow iprev\ n\ (I \downarrow\ge t) = n$
⟨*proof*⟩

definition
$CommuteWithIntervalCut4 :: (('a::linorder)\ set \Rightarrow 'a\ set) \Rightarrow bool$
where
$CommuteWithIntervalCut4\ fun \equiv$
$\forall t\ fun2\ I.$
$(fun2 = (\lambda I.\ I \downarrow< t) \vee fun2 = (\lambda I.\ I \downarrow\le t) \vee fun2 = (\lambda I.\ I \downarrow> t) \vee fun2 = (\lambda I.\ I \downarrow\ge t)\ ) \longrightarrow$
$fun\ (fun2\ I) = fun2\ (fun\ I)$
definition $CommuteWithIntervalCut2 :: (('a::linorder)\ set \Rightarrow 'a\ set) \Rightarrow bool$
where
$CommuteWithIntervalCut2\ fun \equiv$
$\forall t\ fun2\ I.$
$(fun2 = (\lambda I.\ I \downarrow< t) \vee fun2 = (\lambda I.\ I \downarrow> t)) \longrightarrow$
$fun\ (fun2\ I) = fun2\ (fun\ I)$

lemma *CommuteWithIntervalCut4-imp-2*: $CommuteWithIntervalCut4\ fun \Longrightarrow CommuteWithIntervalCut2\ fun$
⟨*proof*⟩

lemma *nat-CommuteWithIntervalCut2-4-eq*:
$CommuteWithIntervalCut4\ (fun::nat\ set \Rightarrow nat\ set) = CommuteWithIntervalCut2\ fun$
⟨*proof*⟩

lemma
cut-less-CommuteWithIntervalCut4: $CommuteWithIntervalCut4\ (\lambda I.\ I \downarrow< t)$
and
cut-le-CommuteWithIntervalCut4: $CommuteWithIntervalCut4\ (\lambda I.\ I \downarrow\le t)$
and
cut-greater-CommuteWithIntervalCut4: $CommuteWithIntervalCut4\ (\lambda I.\ I \downarrow> t)$
and
cut-ge-CommuteWithIntervalCut4: $CommuteWithIntervalCut4\ (\lambda I.\ I \downarrow\ge t)$
⟨*proof*⟩

lemmas *i-cut-CommuteWithIntervalCut4* =
cut-less-CommuteWithIntervalCut4 cut-le-CommuteWithIntervalCut4
cut-greater-CommuteWithIntervalCut4 cut-ge-CommuteWithIntervalCut4

lemma *inext-image*:
$[\![\ n \in I;\ strict\text{-}mono\text{-}on\ f\ I\ ]\!] \Longrightarrow inext\ (f\ n)\ (f\ `\ I) = f\ (inext\ n\ I)$
⟨*proof*⟩

lemma *iprev-image*:
$\llbracket\ n \in I;\ \textit{strict-mono-on}\ f\ I\ \rrbracket \Longrightarrow \textit{iprev}\ (f\ n)\ (f\ `\ I) = f\ (\textit{iprev}\ n\ I)$
⟨*proof*⟩

lemma *inext-image2*:
$\textit{strict-mono}\ f \Longrightarrow \textit{inext}\ (f\ n)\ (f\ `\ I) = f\ (\textit{inext}\ n\ I)$
⟨*proof*⟩

lemma *iprev-image2*:
$\textit{strict-mono}\ f \Longrightarrow \textit{iprev}\ (f\ n)\ (f\ `\ I) = f\ (\textit{iprev}\ n\ I)$
⟨*proof*⟩

lemma *inext-imirror-iprev-conv*:
$\llbracket\ \textit{finite}\ I;\ n \le \textit{iMin}\ I + \textit{Max}\ I\ \rrbracket \Longrightarrow$
$\textit{inext}\ (\textit{mirror-elem}\ n\ I)\ (\textit{imirror}\ I) = \textit{mirror-elem}\ (\textit{iprev}\ n\ I)\ I$
⟨*proof*⟩

corollary *inext-imirror-iprev-conv'*:
$\llbracket\ \textit{finite}\ I;\ n \in I\ \rrbracket \Longrightarrow$
$\textit{inext}\ (\textit{mirror-elem}\ n\ I)\ (\textit{imirror}\ I) = \textit{mirror-elem}\ (\textit{iprev}\ n\ I)\ I$
⟨*proof*⟩

lemma *iprev-imirror-inext-conv*:
$\llbracket\ \textit{finite}\ I;\ n \le \textit{iMin}\ I + \textit{Max}\ I\ \rrbracket \Longrightarrow$
$\textit{iprev}\ (\textit{mirror-elem}\ n\ I)\ (\textit{imirror}\ I) = \textit{mirror-elem}\ (\textit{inext}\ n\ I)\ I$
⟨*proof*⟩

corollary *iprev-imirror-inext-conv'*:
$\llbracket\ \textit{finite}\ I;\ n \in I\ \rrbracket \Longrightarrow$
$\textit{iprev}\ (\textit{mirror-elem}\ n\ I)\ (\textit{imirror}\ I) = \textit{mirror-elem}\ (\textit{inext}\ n\ I)\ I$
⟨*proof*⟩

lemma *inext-insert-ge-Max*:
$\llbracket\ \textit{finite}\ I;\ I \ne \{\};\ \textit{Max}\ I \le a\ \rrbracket \Longrightarrow \textit{inext}\ (\textit{Max}\ I)\ (\textit{insert}\ a\ I) = a$
⟨*proof*⟩

lemma *iprev-insert-le-iMin*:
$\llbracket\ \textit{finite}\ I;\ I \ne \{\};\ a \le \textit{iMin}\ I\ \rrbracket \Longrightarrow \textit{iprev}\ (\textit{iMin}\ I)\ (\textit{insert}\ a\ I) = a$
⟨*proof*⟩

lemma *cut-less-le-iprev-conv*:
$\llbracket\ t \in I;\ t \ne \textit{iMin}\ I\ \rrbracket \Longrightarrow I \downarrow< t = I \downarrow\le (\textit{iprev}\ t\ I)$
⟨*proof*⟩

lemma *neq-Max-imp-inext-neq-iMin*:
$\llbracket\ t \in I;\ t \ne \textit{Max}\ I \vee \textit{infinite}\ I\ \rrbracket \Longrightarrow \textit{inext}\ t\ I \ne \textit{iMin}\ I$
⟨*proof*⟩

corollary *neq-Max-imp-inext-gr-iMin*:
$[\![t \in I;\ t \neq \mathit{Max}\ I \vee \mathit{infinite}\ I]\!] \Longrightarrow \mathit{iMin}\ I < \mathit{inext}\ t\ I$
$\langle \mathit{proof} \rangle$

lemma *cut-le-less-inext-conv*:
$[\![t \in I;\ t \neq \mathit{Max}\ I \vee \mathit{infinite}\ I]\!] \Longrightarrow I \downarrow\leq t = I \downarrow< (\mathit{inext}\ t\ I)$
$\langle \mathit{proof} \rangle$

lemma *cut-ge-greater-iprev-conv*:
$[\![t \in I;\ t \neq \mathit{iMin}\ I\ ]\!] \Longrightarrow I \downarrow\geq t = I \downarrow> (\mathit{iprev}\ t\ I)$
$\langle \mathit{proof} \rangle$

lemma *cut-greater-ge-inext-conv*:
$[\![t \in I;\ t \neq \mathit{Max}\ I \vee \mathit{infinite}\ I\ ]\!] \Longrightarrow I \downarrow> t = I \downarrow\geq (\mathit{inext}\ t\ I)$
$\langle \mathit{proof} \rangle$

lemma *inext-append*:
$[\![\mathit{finite}\ A;\ A \neq \{\};\ B \neq \{\};\ \mathit{Max}\ A < \mathit{iMin}\ B\ ]\!] \Longrightarrow$
$\mathit{inext}\ n\ (A \cup B) = (\mathit{if}\ n \in B\ \mathit{then}\ \mathit{inext}\ n\ B\ \mathit{else}\ (\mathit{if}\ n = \mathit{Max}\ A\ \mathit{then}\ \mathit{iMin}\ B\ \mathit{else}$
$\mathit{inext}\ n\ A))$
$\langle \mathit{proof} \rangle$
corollary *inext-append-eq1*:
$[\![\mathit{finite}\ A;\ A \neq \{\};\ B \neq \{\};\ \mathit{Max}\ A < \mathit{iMin}\ B;\ n \in A;\ n \neq \mathit{Max}\ A\ ]\!] \Longrightarrow$
$\mathit{inext}\ n\ (A \cup B) = \mathit{inext}\ n\ A$
$\langle \mathit{proof} \rangle$
corollary *inext-append-eq2*:
$[\![\mathit{finite}\ A;\ A \neq \{\};\ B \neq \{\};\ \mathit{Max}\ A < \mathit{iMin}\ B;\ n \in B\ ]\!] \Longrightarrow$
$\mathit{inext}\ n\ (A \cup B) = \mathit{inext}\ n\ B$
$\langle \mathit{proof} \rangle$
corollary *inext-append-eq3*:
$[\![\mathit{finite}\ A;\ A \neq \{\};\ B \neq \{\};\ \mathit{Max}\ A < \mathit{iMin}\ B\ ]\!] \Longrightarrow$
$\mathit{inext}\ (\mathit{Max}\ A)\ (A \cup B) = \mathit{iMin}\ B$
$\langle \mathit{proof} \rangle$

lemma *iprev-append*: $[\![\mathit{finite}\ A;\ A \neq \{\};\ B \neq \{\};\ \mathit{Max}\ A < \mathit{iMin}\ B\ ]\!] \Longrightarrow$
$\mathit{iprev}\ n\ (A \cup B) = (\mathit{if}\ n \in A\ \mathit{then}\ \mathit{iprev}\ n\ A\ \mathit{else}\ (\mathit{if}\ n = \mathit{iMin}\ B\ \mathit{then}\ \mathit{Max}\ A\ \mathit{else}$
$\mathit{iprev}\ n\ B))$
$\langle \mathit{proof} \rangle$

corollary *iprev-append-eq1*:
$[\![\mathit{finite}\ A;\ A \neq \{\};\ B \neq \{\};\ \mathit{Max}\ A < \mathit{iMin}\ B;\ n \in A\ ]\!] \Longrightarrow$
$\mathit{iprev}\ n\ (A \cup B) = \mathit{iprev}\ n\ A$
$\langle \mathit{proof} \rangle$

corollary *iprev-append-eq2*:
$[\![\mathit{finite}\ A;\ A \neq \{\};\ B \neq \{\};\ \mathit{Max}\ A < \mathit{iMin}\ B;\ n \in B;\ n \neq \mathit{iMin}\ B\ ]\!] \Longrightarrow$
$\mathit{iprev}\ n\ (A \cup B) = \mathit{iprev}\ n\ B$
$\langle \mathit{proof} \rangle$

corollary *iprev-append-eq3*:
$[\![$ *finite* A; $A \neq \{\}$; $B \neq \{\}$; *Max* $A <$ *iMin* B $]\!] \Longrightarrow$
iprev (*iMin* B) $(A \cup B) =$ *Max* A
⟨*proof*⟩

lemma *inext-predicate-change-exists-aux*: $\bigwedge a.$
$[\![$ $c =$ *card* $(I \downarrow\geq a \downarrow< b)$; $a < b$; $a \in I$; $b \in I$; $\neg P\ a$; $P\ b$ $]\!] \Longrightarrow$
$\exists\, n \in (I \downarrow\geq a \downarrow< b).\ \neg P\ n \wedge P$ (*inext* n I)
⟨*proof*⟩

lemma *inext-predicate-change-exists*:
$[\![$ $a \leq b$; $a \in I$; $b \in I$; $\neg P\ a$; $P\ b$ $]\!] \Longrightarrow$
$\exists\, n{\in}I.\ a \leq n \wedge n < b \wedge \neg P\ n \wedge P$ (*inext* n I)
⟨*proof*⟩

lemma *iprev-predicate-change-exists*:
$[\![$ $a \leq b$; $a \in I$; $b \in I$; $\neg P\ b$; $P\ a$ $]\!] \Longrightarrow$
$\exists\, n{\in}I.\ a < n \wedge n \leq b \wedge \neg P\ n \wedge P$ (*iprev* n I)
⟨*proof*⟩

corollary *nat-Suc-predicate-change-exists*:
$[\![$ $a \leq b$; $\neg P\ a$; $P\ b$ $]\!] \Longrightarrow \exists\, n{\geq}a.\ n < b \wedge \neg P\ n \wedge P$ (*Suc* n)
⟨*proof*⟩

corollary *nat-pred-predicate-change-exists*:
$[\![$ $a \leq b$; $\neg P\ b$; $P\ a$ $]\!] \Longrightarrow \exists\, n{\leq}b.\ a < n \wedge \neg P\ n \wedge P$ $(n -$ *Suc 0*$)$
⟨*proof*⟩

lemma *inext-predicate-change-exists2-all*:
$[\![$ $(a{::}nat) \leq b$; $a \in I$; $b \in I$; $\neg P\ a$; $\forall\, k \in I \downarrow\geq b.\ P\ k$ $]\!] \Longrightarrow$
$\exists\, n{\in}I.\ a \leq n \wedge n < b \wedge \neg P\ n \wedge (\forall\, k \in I \downarrow> n.\ P\ k)$
⟨*proof*⟩

corollary *inext-predicate-change-exists2*:
$[\![$ $(a{::}nat) \leq b$; $a \in I$; $b \in I$; $\neg P\ a$; $P\ b$ $]\!] \Longrightarrow$
$\exists\, n{\in}I.\ a \leq n \wedge n < b \wedge \neg P\ n \wedge (\forall\, k{\in}I.\ n < k \wedge k \leq b \longrightarrow P\ k)$
⟨*proof*⟩

corollary *nat-Suc-predicate-change-exists2-all*:
$[\![$ $(a{::}nat) \leq b$; $\neg P\ a$; $\forall\, k{\geq}b.\ P\ k$ $]\!] \Longrightarrow$
$\exists\, n{\geq}a.\ n < b \wedge \neg P\ n \wedge (\forall\, k{>}n.\ P\ k)$
⟨*proof*⟩

corollary *nat-Suc-predicate-change-exists2*:
$[\![(a::nat) \le b; \neg P\ a; P\ b]\!] \Longrightarrow$
$\exists n \ge a.\ n < b \wedge \neg P\ n \wedge (\forall k \le b.\ n < k \longrightarrow P\ k)$
⟨*proof*⟩

lemma *iprev-predicate-change-exists2-all*:
$[\![(a::nat) \le b; a \in I; b \in I; \neg P\ b; \forall k \in I{\downarrow}{\le}a.\ P\ k]\!] \Longrightarrow$
$\exists n \in I.\ a < n \wedge n \le b \wedge \neg P\ n \wedge (\forall k \in I{\downarrow}{<}n.\ P\ k)$
⟨*proof*⟩

corollary *iprev-predicate-change-exists2*:
$[\![(a::nat) \le b; a \in I; b \in I; \neg P\ b; P\ a]\!] \Longrightarrow$
$\exists n \in I.\ a < n \wedge n \le b \wedge \neg P\ n \wedge (\forall k \in I.\ a \le k \wedge k < n \longrightarrow P\ k)$
⟨*proof*⟩

corollary *nat-pred-predicate-change-exists2-all*:
$[\![(a::nat) \le b; \neg P\ b; \forall k \le a.\ P\ k]\!] \Longrightarrow$
$\exists n > a.\ n \le b \wedge \neg P\ n \wedge (\forall k < n.\ P\ k)$
⟨*proof*⟩

corollary *nat-pred-predicate-change-exists2*:
$[\![(a::nat) \le b; \neg P\ b; P\ a]\!] \Longrightarrow$
$\exists n > a.\ n \le b \wedge \neg P\ n \wedge (\forall k \ge a.\ k < n \longrightarrow P\ k)$
⟨*proof*⟩

8.2 *inext-nth* and *iprev-nth* – nth element of a natural set

primrec *inext-nth* :: *nat set* ⇒ *nat* ⇒ *nat* ((- → -) [*100*, *100*] *60*)
where
$I \to 0 = iMin\ I$
$|\ I \to Suc\ n = inext\ (inext\text{-}nth\ I\ n)\ I$

lemma *inext-nth-closed*: $I \ne \{\} \Longrightarrow I \to n \in I$
⟨*proof*⟩

lemma *inext-nth-image*:
$[\![I \ne \{\}; strict\text{-}mono\text{-}on\ f\ I]\!] \Longrightarrow (f\ `\ I) \to n = f\ (I \to n)$
⟨*proof*⟩

lemma *inext-nth-Suc-mono*: $I \to n \le I \to Suc\ n$
⟨*proof*⟩

lemma *inext-nth-mono*: $a \le b \Longrightarrow I \to a \le I \to b$
⟨*proof*⟩

lemma *inext-nth-Suc-mono2*: $\exists x \in I.\ I \to n < x \Longrightarrow I \to n < I \to Suc\ n$
⟨*proof*⟩

lemma *inext-nth-mono2*: $\exists\, x{\in}I.\ I \rightarrow a < x \Longrightarrow (I \rightarrow a < I \rightarrow b) = (a < b)$
$\langle proof \rangle$

lemma *inext-nth-mono2-infin*:
$\ \ infinite\ I \Longrightarrow (I \rightarrow a < I \rightarrow b) = (a < b)$
$\langle proof \rangle$

lemma *inext-nth-Max-fix*:
$\ \ [\![\, finite\ I;\ I \neq \{\};\ I \rightarrow a = Max\ I;\ a \leq b\ ]\!] \Longrightarrow I \rightarrow b = Max\ I$
$\langle proof \rangle$

lemma *inext-nth-cut-less-conv*:
$\ \ \bigwedge I.\ I \rightarrow n < t \Longrightarrow (I \downarrow< t) \rightarrow n = I \rightarrow n$
$\langle proof \rangle$

lemma *remove-Min-inext-nth-Suc-conv*: $\bigwedge I.$
$\ \ Suc\ 0 < card\ I \vee infinite\ I \Longrightarrow$
$\ \ (I - \{iMin\ I\}) \rightarrow n = I \rightarrow Suc\ n$

$\langle proof \rangle$

corollary *remove-Min-inext-nth-Suc-conv-finite*: $Suc\ 0 < card\ I \Longrightarrow (I - \{iMin\ I\}) \rightarrow n = I \rightarrow Suc\ n$
$\langle proof \rangle$
corollary *remove-Min-inext-nth-Suc-conv-infinite*: $infinite\ I \Longrightarrow (I - \{iMin\ I\}) \rightarrow n = I \rightarrow Suc\ n$
$\langle proof \rangle$

lemma *remove-Max-eq*: $[\![\, finite\ I;\ I \neq \{\};\ n \neq Max\ I\ ]\!] \Longrightarrow Max\ (I - \{n\}) = Max\ I$
$\langle proof \rangle$
lemma *remove-iMin-eq*: $[\![\ I \neq \{\};\ n \neq iMin\ I\ ]\!] \Longrightarrow iMin\ (I - \{n\}) = iMin\ I$
$\langle proof \rangle$
lemma *remove-Min-eq*: $[\![\, finite\ I;\ I \neq \{\};\ n \neq Min\ I\ ]\!] \Longrightarrow Min\ (I - \{n\}) = Min\ I$
$\langle proof \rangle$
lemma *Max-le-iMin-conv-singleton*: $[\![\, finite\ I;\ I \neq \{\}\ ]\!] \Longrightarrow (Max\ I \leq iMin\ I) = (\exists\, x.\ I = \{x\})$
$\langle proof \rangle$

lemma *inext-nth-card-less-Max*:
$\ \ \bigwedge I.\ Suc\ n < card\ I \Longrightarrow I \rightarrow n < Max\ I$
$\langle proof \rangle$

lemma *inext-nth-card-less-Max'*:
$\ \ n < card\ I - Suc\ 0 \Longrightarrow I \rightarrow n < Max\ I$

$\langle proof \rangle$

lemma *inext-nth-card-Max-aux*:
$\bigwedge I.\ card\ I = Suc\ n \Longrightarrow I \to n = Max\ I$
$\langle proof \rangle$

lemma *inext-nth-card-Max-aux'*:
$\bigwedge I.\ [\![\ finite\ I;\ I \neq \{\}\ ]\!] \Longrightarrow I \to (card\ I - Suc\ 0) = Max\ I$
$\langle proof \rangle$

lemma *inext-nth-card-Max*:
$[\![\ finite\ I;\ I \neq \{\};\ card\ I \leq Suc\ n\ ]\!] \Longrightarrow I \to n = Max\ I$
$\langle proof \rangle$

lemma *inext-nth-card-Max'*:
$[\![\ finite\ I;\ I \neq \{\};\ card\ I - Suc\ 0 \leq n\ ]\!] \Longrightarrow I \to n = Max\ I$
$\langle proof \rangle$

lemma *inext-nth-singleton*: $\{a\} \to n = a$
$\langle proof \rangle$

lemma *inext-nth-eq-Min-conv*:
$I \neq \{\} \Longrightarrow (I \to n = iMin\ I) = (n = 0 \vee (\exists\, a.\ I = \{a\}))$
$\langle proof \rangle$

lemma *inext-nth-gr-Min-conv*:
$I \neq \{\} \Longrightarrow (iMin\ I < I \to n) = (0 < n \wedge \neg(\exists\, a.\ I = \{a\}))$
$\langle proof \rangle$

lemma *inext-nth-gr-Min-conv-infinite*:
$infinite\ I \Longrightarrow (iMin\ I < I \to n) = (0 < n)$
$\langle proof \rangle$

lemma *inext-nth-cut-ge-inext-nth*: $\bigwedge I\ b.$
$I \neq \{\} \Longrightarrow I \downarrow\geq (I \to a) \to b = I \to (a + b)$
$\langle proof \rangle$

lemma *inext-nth-append-eq1*:
$[\![\ finite\ A;\ A \neq \{\};\ Max\ A < iMin\ B;\ A \to n \neq Max\ A\ ]\!] \Longrightarrow$
$(A \cup B) \to n = A \to n$
$\langle proof \rangle$

lemma *inext-nth-card-append-eq1*:
$\bigwedge A\ B.[\![\ Max\ A < iMin\ B;\ n < card\ A\ ]\!] \Longrightarrow$
$(A \cup B) \to n = A \to n$
$\langle proof \rangle$

lemma *inext-nth-card-append-eq3*:
$\llbracket$ *finite* A; $B \neq \{\}$; *Max* $A <$ *iMin* B $\rrbracket \Longrightarrow$
$(A \cup B) \rightarrow (card\ A) = iMin\ B$
⟨*proof*⟩

lemma *inext-nth-card-append-eq2*:
$\llbracket$ *finite* A; $A \neq \{\}$; $B \neq \{\}$; *Max* $A <$ *iMin* B; *card* $A \leq n$ $\rrbracket \Longrightarrow$
$(A \cup B) \rightarrow n = B \rightarrow (n - card\ A)$
⟨*proof*⟩

lemma *inext-nth-card-append*:
$\llbracket$ *finite* A; $A \neq \{\}$; $B \neq \{\}$; *Max* $A <$ *iMin* B $\rrbracket \Longrightarrow$
$(A \cup B) \rightarrow n = (if\ n < card\ A\ then\ A \rightarrow n\ else\ B \rightarrow (n - card\ A))$
⟨*proof*⟩

lemma *inext-nth-insert-Suc*:
$\llbracket I \neq \{\}; a < iMin\ I \rrbracket \Longrightarrow (insert\ a\ I) \rightarrow Suc\ n = I \rightarrow n$
⟨*proof*⟩

lemma *inext-nth-cut-less-eq*:
$n < card\ (I \downarrow< t) \Longrightarrow (I \downarrow< t) \rightarrow n = I \rightarrow n$
⟨*proof*⟩

lemma *less-card-cut-less-imp-inext-nth-less*:
$n < card\ (I \downarrow< t) \Longrightarrow I \rightarrow n < t$
⟨*proof*⟩

lemma *inext-nth-less-less-card-conv*:
$I \downarrow\geq t \neq \{\} \Longrightarrow (I \rightarrow n < t) = (n < card\ (I \downarrow< t))$
⟨*proof*⟩

lemma *cut-less-inext-nth-card-eq1*:
$n < card\ I \vee infinite\ I \Longrightarrow card\ (I \downarrow< (I \rightarrow n)) = n$
⟨*proof*⟩

lemma *cut-less-inext-nth-card-eq2*:
$\llbracket finite\ I; card\ I \leq Suc\ n \rrbracket \Longrightarrow card\ (I \downarrow< (I \rightarrow n)) = card\ I - Suc\ 0$
⟨*proof*⟩

lemma *cut-less-inext-nth-card-if*:
$card\ (I \downarrow< (I \rightarrow n)) = ($
$if\ (n < card\ I \vee infinite\ I)\ then\ n\ else\ card\ I - Suc\ 0)$
⟨*proof*⟩

lemma *cut-le-inext-nth-card-eq1*:
$n < card\ I \vee infinite\ I \Longrightarrow card\ (I \downarrow\leq (I \rightarrow n)) = Suc\ n$
⟨*proof*⟩

lemma *cut-le-inext-nth-card-eq2*:
$[\![$ *finite* I; *card* $I \leq$ *Suc* n $]\!] \Longrightarrow$ *card* $(I \downarrow\leq (I \rightarrow n)) =$ *card* I
$\langle proof \rangle$

lemma *cut-le-inext-nth-card-if*:
card $(I \downarrow\leq (I \rightarrow n)) = ($
if $(n <$ *card* $I \vee$ *infinite* $I)$ *then Suc* n *else card* $I)$
$\langle proof \rangle$

primrec *iprev-nth* :: *nat set* $\Rightarrow$ *nat* $\Rightarrow$ *nat* ((- $\leftarrow$ -) [*100*, *100*] *60*)
where
$I \leftarrow 0 =$ *Max* I
$|$ $I \leftarrow$ *Suc* $n =$ *iprev* (*iprev-nth* I n) I

lemma *iprev-nth-closed*: $[\![$ *finite* I; $I \neq \{\}$ $]\!] \Longrightarrow I \leftarrow n \in I$
$\langle proof \rangle$

lemma *iprev-nth-image*:
$[\![$ *finite* I; $I \neq \{\}$; *strict-mono-on* f I $]\!] \Longrightarrow (f$ ` $I) \leftarrow n = f\ (I \leftarrow n)$
$\langle proof \rangle$

lemma *iprev-nth-Suc-mono*: $I \leftarrow (Suc\ n) \leq I \leftarrow n$
$\langle proof \rangle$

lemma *iprev-nth-mono*: $a \leq b \Longrightarrow I \leftarrow b \leq I \leftarrow a$
$\langle proof \rangle$

lemma *iprev-nth-Suc-mono2*:
$[\![$ *finite* I; $\exists x \in I.\ x < I \leftarrow n$ $]\!] \Longrightarrow I \leftarrow (Suc\ n) < I \leftarrow n$
$\langle proof \rangle$

lemma *iprev-nth-mono2*:
$[\![$ *finite* I; $\exists x \in I.\ x < I \leftarrow a$ $]\!] \Longrightarrow (I \leftarrow b < I \leftarrow a) = (a < b)$
$\langle proof \rangle$

lemma *iprev-nth-iMin-fix*:
$[\![$ $I \neq \{\}$; $I \leftarrow a =$ *iMin* I; $a \leq b$ $]\!] \Longrightarrow I \leftarrow b =$ *iMin* I
$\langle proof \rangle$

lemma *iprev-nth-singleton*: $\{a\} \leftarrow n = a$
$\langle proof \rangle$

8.3 Induction over arbitrary natural sets using the functions *inext* and *iprev*

lemma *inext-nth-surj-aux1*:
$\{x \in I.\ \neg(\exists n.\ I \rightarrow n = x)\} = \{\}$
(**is** $?S = \{\}$

is $\{ x \in I.\ ?P\ x\} = \{\})$
⟨*proof*⟩

lemma *inext-nth-surj-on*:*surj-on* $(\lambda n.\ I \to n)$ *UNIV I*
⟨*proof*⟩

corollary *in-imp-ex-inext-nth*: $x \in I \Longrightarrow \exists\, n.\ x = I \to n$
⟨*proof*⟩

lemma *inext-induct*:
$[\![\ P\ (iMin\ I);\ \bigwedge n.\ [\![\ n \in I;\ P\ n\ ]\!] \Longrightarrow P\ (inext\ n\ I);\ n \in I\ ]\!] \Longrightarrow P\ n$
⟨*proof*⟩

lemma *iprev-nth-surj-aux1*:
$finite\ I \Longrightarrow \{\ x \in I.\ \neg(\exists\, n.\ I \leftarrow n = x)\} = \{\}$
⟨*proof*⟩

lemma *iprev-nth-surj-on*: $finite\ I \Longrightarrow surj\text{-}on\ (\lambda n.\ I \leftarrow n)\ UNIV\ I$
⟨*proof*⟩

corollary *in-imp-ex-iprev-nth*:
$[\![\ finite\ I;\ \ x \in I\ ]\!] \Longrightarrow \exists\, n.\ x = I \leftarrow n$
⟨*proof*⟩

lemma *iprev-induct*:
$[\![\ P\ (Max\ I);\ \bigwedge n.\ [\![\ n \in I;\ P\ n\ ]\!] \Longrightarrow P\ (iprev\ n\ I);\ finite\ I;\ n \in I\ ]\!] \Longrightarrow P\ n$
⟨*proof*⟩

8.4 Natural intervals with *inext* and *iprev*

lemma *inext-atLeast*: $n \le t \Longrightarrow inext\ t\ \{n..\} = Suc\ t$
⟨*proof*⟩

lemma *iprev-atLeast'*: $n \le t \Longrightarrow iprev\ (Suc\ t)\ \{n..\} = t$
⟨*proof*⟩

lemma *iprev-atLeast*: $n < t \ \Longrightarrow iprev\ t\ \{n..\} = t - Suc\ 0$
⟨*proof*⟩

lemma *inext-atMost*: $t < n \Longrightarrow inext\ t\ \{..n\} = Suc\ t$
⟨*proof*⟩

lemma *iprev-atMost*: $t \le n \Longrightarrow iprev\ t\ \{..n\} = t - Suc\ 0$
⟨*proof*⟩

lemma *inext-lessThan*: $Suc\ t < n \Longrightarrow inext\ t\ \{..{<}n\} = Suc\ t$
⟨*proof*⟩
lemma *iprev-lessThan*: $t < n \Longrightarrow iprev\ t\ \{..{<}n\} = t - Suc\ 0$
⟨*proof*⟩

lemma *inext-atLeastAtMost*: $\llbracket\ m \le t;\ t < n\ \rrbracket \Longrightarrow inext\ t\ \{m..n\} = Suc\ t$
⟨*proof*⟩
lemma *iprev-atLeastAtMost*: $\llbracket\ m < t;\ t \le n\ \rrbracket \Longrightarrow iprev\ t\ \{m..n\} = t - Suc\ 0$
⟨*proof*⟩
lemma *iprev-atLeastAtMost'*: $\llbracket\ m \le t;\ t < n\ \rrbracket \Longrightarrow iprev\ (Suc\ t)\ \{m..n\} = t$
⟨*proof*⟩

lemma *inext-nth-atLeast* : $\{n..\} \rightarrow a = n + a$
⟨*proof*⟩
lemma *inext-nth-atLeastAtMost*: $\llbracket\ a \le n - m;\ m \le n\ \rrbracket \Longrightarrow \{m..n\} \rightarrow a = m + a$
⟨*proof*⟩
lemma *iprev-nth-atLeastAtMost*: $\llbracket\ a \le n - m;\ m \le n\ \rrbracket \Longrightarrow \{m..n\} \leftarrow a = n - a$
⟨*proof*⟩
lemma *inext-nth-atMost*: $a \le n \Longrightarrow \{..n\} \rightarrow a = a$
⟨*proof*⟩
lemma *iprev-nth-atMost*: $a \le n \Longrightarrow \{..n\} \leftarrow a = n - a$
⟨*proof*⟩

lemma *inext-nth-lessThan* : $a < n \Longrightarrow \{..<n\} \rightarrow a = a$
⟨*proof*⟩
lemma *iprev-nth-lessThan*: $a < n \Longrightarrow \{..<n\} \leftarrow a = n - Suc\ a$
⟨*proof*⟩

lemma *inext-nth-UNIV*: $UNIV \rightarrow a = a$
⟨*proof*⟩

8.5 Further result for *inext-nth* and *iprev-nth*

lemma *inext-iprev-nth-Suc*:
$iMin\ I \ne I \leftarrow n \Longrightarrow inext\ (I \leftarrow Suc\ n)\ I = I \leftarrow n$
⟨*proof*⟩

lemma *inext-iprev-nth-pred*:
$\llbracket\ finite\ I;\ iMin\ I \ne I \leftarrow (n - Suc\ 0)\ \rrbracket \Longrightarrow$
$inext\ (I \leftarrow n)\ I = I \leftarrow (n - Suc\ 0)$
⟨*proof*⟩

lemma *iprev-inext-nth-Suc*:
$I \rightarrow n \ne Max\ I \lor infinite\ I \Longrightarrow iprev\ (I \rightarrow Suc\ n)\ I = I \rightarrow n$
⟨*proof*⟩
lemma *iprev-inext-nth-pred*:
$I \rightarrow (n - Suc\ 0) \ne Max\ I \lor infinite\ I \Longrightarrow$
$iprev\ (I \rightarrow n)\ I = I \rightarrow (n - Suc\ 0)$
⟨*proof*⟩

lemma *inext-nth-imirror-iprev-nth-conv*:

$[\![$ *finite* I; $I \neq \{\}$ $]\!] \Longrightarrow$
$(imirror\ I) \rightarrow n = mirror\text{-}elem\ (I \leftarrow n)\ I$
$\langle proof \rangle$

corollary *inext-nth-imirror-iprev-nth-conv2*:
$[\![$ *finite* I; $I \neq \{\}$ $]\!] \Longrightarrow$
$mirror\text{-}elem\ ((imirror\ I) \leftarrow n)\ I = I \rightarrow n$
$\langle proof \rangle$

lemma *iprev-nth-imirror-inext-nth-conv*:
$[\![$ *finite* I; $I \neq \{\}$ $]\!] \Longrightarrow$
$(imirror\ I) \leftarrow n = mirror\text{-}elem\ (I \rightarrow n)\ I$
$\langle proof \rangle$

corollary *iprev-nth-imirror-inext-nth-conv2*:
$[\![$ *finite* I; $I \neq \{\}$ $]\!] \Longrightarrow$
$mirror\text{-}elem\ ((imirror\ I) \rightarrow n)\ I = (I \leftarrow n)$
$\langle proof \rangle$

lemma *iprev-nth-card-greater-iMin*: $Suc\ n < card\ I \Longrightarrow iMin\ I < I \leftarrow n$
$\langle proof \rangle$

lemma *iprev-nth-card-iMin*:
$[\![$ *finite* I; $I \neq \{\}$; $card\ I \leq Suc\ n$ $]\!] \Longrightarrow I \leftarrow n = iMin\ I$
$\langle proof \rangle$

lemma *iprev-nth-card-iMin'*:
$[\![$ *finite* I; $I \neq \{\}$; $card\ I - Suc\ 0 \leq n$ $]\!] \Longrightarrow I \leftarrow n = iMin\ I$
$\langle proof \rangle$

end

9 Additional definitions and results for lists

theory *List2*
imports *../CommonSet/SetIntervalCut*
begin

9.1 Additional definitions and results for lists

Infix syntactical abbreviations for operators *take* and *drop*. The abbreviations resemble to the operator symbols used later for take and drop operators on infinite lists in ListInf.

abbreviation *f-take'* :: *'a list* $\Rightarrow$ *nat* $\Rightarrow$ *'a list* (**infixl** $\downarrow$ *100*)
where $xs \downarrow n \equiv take\ n\ xs$
abbreviation *f-drop'* :: *'a list* $\Rightarrow$ *nat* $\Rightarrow$ *'a list* (**infixl** $\uparrow$ *100*)
where $xs \uparrow n \equiv drop\ n\ xs$

lemma *append-eq-Cons*: $[x] @ xs = x \# xs$
$\langle proof \rangle$

lemma *length-Cons*: $length\ (x \# xs) = Suc\ (length\ xs)$
$\langle proof \rangle$

lemma *length-snoc*: $length\ (xs @ [x]) = Suc\ (length\ xs)$
$\langle proof \rangle$

9.1.1 Additional lemmata about list emptiness

lemma *length-greater-imp-not-empty*:$n < length\ xs \Longrightarrow xs \neq []$
$\langle proof \rangle$

lemma *length-ge-Suc-imp-not-empty*:$Suc\ n \leq length\ xs \Longrightarrow xs \neq []$
$\langle proof \rangle$

lemma *length-take-le*: $length\ (xs \downarrow n) \leq length\ xs$
$\langle proof \rangle$

lemma *take-not-empty-conv*:$(xs \downarrow n \neq []) = (0 < n \wedge xs \neq [])$
$\langle proof \rangle$

lemma *drop-not-empty-conv*:$(xs \uparrow n \neq []) = (n < length\ xs)$
$\langle proof \rangle$

lemma *zip-eq-Nil*: $(zip\ xs\ ys = []) = (xs = [] \vee ys = [])$
$\langle proof \rangle$

lemma *zip-not-empty-conv*: $(zip\ xs\ ys \neq []) = (xs \neq [] \wedge ys \neq [])$
$\langle proof \rangle$

9.1.2 Additional lemmata about *take*, *drop*, *hd*, *last*, *nth* and *filter*

lemma *nth-tl-eq-nth-Suc*:
$Suc\ n \leq length\ xs \Longrightarrow (tl\ xs) ! n = xs ! Suc\ n$
$\langle proof \rangle$
corollary *nth-tl-eq-nth-Suc2*:
$n < length\ xs \Longrightarrow (tl\ xs) ! n = xs ! Suc\ n$
$\langle proof \rangle$

lemma *hd-eq-first*: $xs \neq [] \Longrightarrow xs ! 0 = hd\ xs$
$\langle proof \rangle$
corollary *take-first*:$xs \neq [] \Longrightarrow xs \downarrow (Suc\ 0) = [xs ! 0]$
$\langle proof \rangle$
corollary *take-hd*:$xs \neq [] \Longrightarrow xs \downarrow (Suc\ 0) = [hd\ xs]$
$\langle proof \rangle$

theorem *last-nth*: $xs \neq [] \Longrightarrow last\ xs = xs ! (length\ xs - Suc\ 0)$

$\langle proof \rangle$

lemma *last-take*: $n < length\ xs \Longrightarrow last\ (xs \downarrow Suc\ n) = xs\ !\ n$
$\langle proof \rangle$
corollary *last-take2*:
$[\![\ 0 < n;\ n \leq length\ xs\ ]\!] \Longrightarrow last\ (xs \downarrow n) = xs\ !\ (n - Suc\ 0)$
$\langle proof \rangle$

corollary *nth-0-drop*: $n \leq length\ xs \Longrightarrow (xs \uparrow n)\ !\ 0 = xs\ !\ n$
$\langle proof \rangle$

lemma *drop-eq-tl*: $xs \uparrow (Suc\ 0) = tl\ xs$
$\langle proof \rangle$

lemma *drop-take-1*:
$n < length\ xs \Longrightarrow xs \uparrow n \downarrow (Suc\ 0) = [xs\ !\ n]$
$\langle proof \rangle$

lemma *upt-append*: $m \leq n \Longrightarrow [0..<m]\ @\ [m..<n] = [0..<n]$
$\langle proof \rangle$

lemma *nth-append1*: $n < length\ xs \Longrightarrow (xs\ @\ ys)\ !\ n = xs\ !\ n$
$\langle proof \rangle$
lemma *nth-append2*: $length\ xs \leq n \Longrightarrow (xs\ @\ ys)\ !\ n = ys\ !\ (n - length\ xs)$
$\langle proof \rangle$

lemma *list-all-conv*: $list\text{-}all\ P\ xs = (\forall\, i<length\ xs.\ P\ (xs\ !\ i))$
$\langle proof \rangle$

lemma *expand-list-eq*:
$\bigwedge ys.\ (xs = ys) = (length\ xs = length\ ys \wedge (\forall\, i<length\ xs.\ xs\ !\ i = ys\ !\ i))$
$\langle proof \rangle$
lemmas *list-eq-iff* = *expand-list-eq*

lemma *list-take-drop-imp-eq*:
$[\![\ xs \downarrow n = ys \downarrow n;\ \ xs \uparrow n = ys \uparrow n\ ]\!] \Longrightarrow xs = ys$
$\langle proof \rangle$

lemma *list-take-drop-eq-conv*:
$(xs = ys) = (\exists\, n.\ (xs \downarrow n = ys \downarrow n \wedge xs \uparrow n = ys \uparrow n))$
$\langle proof \rangle$

lemma *list-take-eq-conv*: $(xs = ys) = (\forall\, n.\ xs \downarrow n = ys \downarrow n)$
$\langle proof \rangle$

lemma *list-drop-eq-conv*: $(xs = ys) = (\forall\, n.\ xs \uparrow n = ys \uparrow n)$

$\langle proof \rangle$

abbreviation *replicate'* :: *'a* ⇒ *nat* ⇒ *'a list* (-⁻ [*1000*,*65*])
where $x^n \equiv$ *replicate n x*

lemma *replicate-snoc*: x^n @ $[x]$ = $x^{Suc\ n}$
$\langle proof \rangle$

lemma *eq-replicate-conv*: $(\forall\, i{<}length\ xs.\ xs\ !\ i = m) = (xs = m^{length\ xs})$
$\langle proof \rangle$

lemma *replicate-Cons-length*: $length\ (x\ \#\ a^n) = Suc\ n$
$\langle proof \rangle$
lemma *replicate-pred-Cons-length*: $0 < n \Longrightarrow length\ (x\ \#\ a^{n\ -\ Suc\ 0}) = n$
$\langle proof \rangle$

lemma *replicate-le-diff*: $m \leq n \Longrightarrow x^m$ @ $x^{n\ -\ m} = x^n$
$\langle proof \rangle$
lemma *replicate-le-diff2*: $[\![\ k \leq m;\ m \leq n\ ]\!] \Longrightarrow x^{m\ -\ k}$ @ $x^{n\ -\ m} = x^{n\ -\ k}$
$\langle proof \rangle$

lemma *append-constant-length-induct-aux*: $\bigwedge xs.$
$[\![\ length\ xs\ div\ k = n;\ \bigwedge ys.\ k = 0 \vee length\ ys < k \Longrightarrow P\ ys;$
$\bigwedge xs\ ys.\ [\![\ length\ xs = k;\ P\ ys\ ]\!] \Longrightarrow P\ (xs$ @ $ys)\ ]\!] \Longrightarrow P\ xs$
$\langle proof \rangle$

lemma *append-constant-length-induct*:
$[\![\ \bigwedge ys.\ k = 0 \vee length\ ys < k \Longrightarrow P\ ys;$
$\bigwedge xs\ ys.\ [\![\ length\ xs = k;\ P\ ys\ ]\!] \Longrightarrow P\ (xs$ @ $ys)\ ]\!] \Longrightarrow P\ xs$
$\langle proof \rangle$

lemma *zip-swap*: $map\ (\lambda(y,x).\ (x,y))\ (zip\ ys\ xs) = (zip\ xs\ ys)$
$\langle proof \rangle$

lemma *zip-takeL*: $(zip\ xs\ ys) \downarrow n = zip\ (xs \downarrow n)\ ys$
$\langle proof \rangle$

lemma *zip-takeR*: $(zip\ xs\ ys) \downarrow n = zip\ xs\ (ys \downarrow n)$
$\langle proof \rangle$

lemma *zip-take*: $(zip\ xs\ ys) \downarrow n = zip\ (xs \downarrow n)\ (ys \downarrow n)$
$\langle proof \rangle$

lemma *hd-zip*: $[\![\ xs \neq [];\ ys \neq []\ ]\!] \Longrightarrow hd\ (zip\ xs\ ys) = (hd\ xs,\ hd\ ys)$
$\langle proof \rangle$

lemma *map-id*: $map\ id\ xs = xs$
$\langle proof \rangle$

lemma *map-id-subst*: $P\ (map\ id\ xs) \Longrightarrow P\ xs$
⟨*proof*⟩

lemma *map-one*: $map\ f\ [x] = [f\ x]$
⟨*proof*⟩

lemma *map-last*: $xs \neq [] \Longrightarrow last\ (map\ f\ xs) = f\ (last\ xs)$
⟨*proof*⟩

lemma *filter-list-all*: $list\text{-}all\ P\ xs \Longrightarrow filter\ P\ xs = xs$
⟨*proof*⟩

lemma *filter-snoc*: $filter\ P\ (xs\ @\ [x]) = (if\ P\ x\ then\ (filter\ P\ xs)\ @\ [x]\ else\ filter\ P\ xs)$
⟨*proof*⟩

lemma *filter-filter-eq*: $list\text{-}all\ (\lambda x.\ P\ x = Q\ x)\ xs \Longrightarrow filter\ P\ xs = filter\ Q\ xs$
⟨*proof*⟩

lemma *filter-nth*: $\bigwedge n.$
$n < length\ (filter\ P\ xs) \Longrightarrow$
$(filter\ P\ xs)\ !\ n =$
$xs\ !\ (LEAST\ k.$
$k < length\ xs \wedge$
$n < card\ \{i.\ i \leq k \wedge i < length\ xs \wedge P\ (xs\ !\ i)\})$
⟨*proof*⟩

9.1.3 Ordered lists

fun *list-ord* :: $('a \Rightarrow 'a \Rightarrow bool) \Rightarrow ('a::ord)\ list \Rightarrow bool$
where
$list\text{-}ord\ ord\ (x1\ \#\ x2\ \#\ xs) = (ord\ x1\ x2 \wedge list\text{-}ord\ ord\ (x2\ \#\ xs))$
$|\ list\text{-}ord\ ord\ xs = True$

definition *list-asc* :: $('a::ord)\ list \Rightarrow bool$ **where**
$list\text{-}asc\ xs \equiv list\text{-}ord\ (\leq)\ xs$
definition *list-strict-asc* :: $('a::ord)\ list \Rightarrow bool$ **where**
$list\text{-}strict\text{-}asc\ xs \equiv list\text{-}ord\ (<)\ xs$
value $list\text{-}asc\ [1::nat,\ 2,\ 2]$
value $list\text{-}strict\text{-}asc\ [1::nat,\ 2,\ 2]$
definition *list-desc* :: $('a::ord)\ list \Rightarrow bool$ **where**
$list\text{-}desc\ xs \equiv list\text{-}ord\ (\geq)\ xs$
definition *list-strict-desc* :: $('a::ord)\ list \Rightarrow bool$ **where**
$list\text{-}strict\text{-}desc\ xs \equiv list\text{-}ord\ (>)\ xs$

lemma *list-ord-Nil*: $list\text{-}ord\ ord\ []$
⟨*proof*⟩

lemma *list-ord-one*: *list-ord ord* $[x]$
⟨*proof*⟩
lemma *list-ord-Cons*:
list-ord ord $(x \# xs) =$
$(xs = [] \vee (ord\ x\ (hd\ xs) \wedge list\text{-}ord\ ord\ xs))$
⟨*proof*⟩
lemma *list-ord-Cons-imp*: $[\![$ *list-ord ord xs*; *ord x* $(hd\ xs)$ $]\!] \Longrightarrow$ *list-ord ord* $(x \#$ $xs)$
⟨*proof*⟩
lemma *list-ord-append*: $\bigwedge ys.$
list-ord ord $(xs\ @\ ys) =$
$(list\text{-}ord\ ord\ xs \wedge$
$(ys = [] \vee (list\text{-}ord\ ord\ ys \wedge (xs = [] \vee ord\ (last\ xs)\ (hd\ ys)))))$
⟨*proof*⟩
lemma *list-ord-snoc*:
list-ord ord $(xs\ @\ [x]) =$
$(xs = [] \vee (ord\ (last\ xs)\ x \wedge list\text{-}ord\ ord\ xs))$
⟨*proof*⟩

lemma *list-ord-all-conv*:
$(list\text{-}ord\ ord\ xs) = (\forall n < length\ xs - 1.\ ord\ (xs\ !\ n)\ (xs\ !\ Suc\ n))$
⟨*proof*⟩

lemma *list-ord-imp*:
$[\![\bigwedge x\ y.\ ord\ x\ y \Longrightarrow ord'\ x\ y;\ list\text{-}ord\ ord\ xs\ ]\!] \Longrightarrow$
list-ord ord′ xs
⟨*proof*⟩
corollary *list-strict-asc-imp-list-asc*:
list-strict-asc $(xs{::}'a{::}preorder\ list) \Longrightarrow$ *list-asc xs*
⟨*proof*⟩
corollary *list-strict-desc-imp-list-desc*:
list-strict-desc $(xs{::}'a{::}preorder\ list) \Longrightarrow$ *list-desc xs*
⟨*proof*⟩

lemma *list-ord-trans-imp*: $\bigwedge i.$
$[\![$ *transP ord*; *list-ord ord xs*; $j < length\ xs$; $i < j$ $]\!] \Longrightarrow$
$ord\ (xs\ !\ i)\ (xs\ !\ j)$
⟨*proof*⟩

lemma *list-ord-trans*:
transP ord $\Longrightarrow$
$(list\text{-}ord\ ord\ xs) =$
$(\forall j < length\ xs.\ \forall i < j.\ ord\ (xs\ !\ i)\ (xs\ !\ j))$
⟨*proof*⟩

lemma *list-ord-trans-refl-le*:
$[\![$ *transP ord*; *reflP ord* $]\!] \Longrightarrow$
$(list\text{-}ord\ ord\ xs) =$
$(\forall j < length\ xs.\ \forall i \le j.\ ord\ (xs\ !\ i)\ (xs\ !\ j))$

$\langle proof \rangle$

lemma *list-ord-trans-refl-le-imp*:
$[\![$ *transP ord*; $\bigwedge x\ y.\ ord\ x\ y \Longrightarrow ord'\ x\ y$; *reflP ord'*;
list-ord ord xs $]\!] \Longrightarrow$
$(\forall j < length\ xs.\ \forall i \leq j.\ ord'\ (xs\ !\ i)\ (xs\ !\ j))$
$\langle proof \rangle$

corollary
list-asc-trans:
$(list\text{-}asc\ (xs::'a::preorder\ list)) =$
$(\forall j < length\ xs.\ \forall i < j.\ xs\ !\ i \leq xs\ !\ j)$ **and**
list-strict-asc-trans:
$(list\text{-}strict\text{-}asc\ (xs::'a::preorder\ list)) =$
$(\forall j < length\ xs.\ \forall i < j.\ xs\ !\ i < xs\ !\ j)$ **and**
list-desc-trans:
$(list\text{-}desc\ (xs::'a::preorder\ list)) =$
$(\forall j < length\ xs.\ \forall i < j.\ xs\ !\ j \leq xs\ !\ i)$ **and**
list-strict-desc-trans:
$(list\text{-}strict\text{-}desc\ (xs::'a::preorder\ list)) =$
$(\forall j < length\ xs.\ \forall i < j.\ xs\ !\ j < xs\ !\ i)$
$\langle proof \rangle$

corollary
list-asc-trans-le:
$(list\text{-}asc\ (xs::'a::preorder\ list)) =$
$(\forall j < length\ xs.\ \forall i \leq j.\ xs\ !\ i \leq xs\ !\ j)$ **and**
list-desc-trans-le:
$(list\text{-}desc\ (xs::'a::preorder\ list)) =$
$(\forall j < length\ xs.\ \forall i \leq j.\ xs\ !\ j \leq xs\ !\ i)$
$\langle proof \rangle$

corollary
list-strict-asc-trans-le:
$(list\text{-}strict\text{-}asc\ (xs::'a::preorder\ list)) \Longrightarrow$
$(\forall j < length\ xs.\ \forall i \leq j.\ xs\ !\ i \leq xs\ !\ j)$
$\langle proof \rangle$

lemma *list-ord-le-sorted-eq*: *list-asc xs* = *sorted xs*
$\langle proof \rangle$

corollary *list-asc-upto*: *list-asc* $[m..n]$
$\langle proof \rangle$

lemma *list-strict-asc-upt*: *list-strict-asc* $[m..{<}n]$
$\langle proof \rangle$

lemma *list-ord-distinct-aux*:

$\llbracket$ *irrefl* $\{(a,\ b).\ ord\ a\ b\}$; *transP ord*; *list-ord ord xs*;
$i <$ *length xs*; $j <$ *length xs*; $i < j$ $\rrbracket \Longrightarrow$
xs ! $i \neq$ *xs* ! j
⟨*proof*⟩

lemma *list-ord-distinct*:
$\llbracket$ *irrefl* $\{(a,b).\ ord\ a\ b\}$; *transP ord*; *list-ord ord xs* $\rrbracket \Longrightarrow$
distinct xs
⟨*proof*⟩

lemma *list-strict-asc-distinct*: *list-strict-asc* (*xs*::'*a*::*preorder list*) $\Longrightarrow$ *distinct xs*
⟨*proof*⟩

lemma *list-strict-desc-distinct*: *list-strict-desc* (*xs*::'*a*::*preorder list*) $\Longrightarrow$ *distinct xs*
⟨*proof*⟩

9.1.4 Additional definitions and results for sublists

primrec *sublist-list* :: '*a list* $\Rightarrow$ *nat list* $\Rightarrow$ '*a list*
where
sublist-list xs [] = []
| *sublist-list xs* (*y* # *ys*) = (*xs* ! *y*) # (*sublist-list xs ys*)

value *sublist-list* [*0*::*int*,*10*::*int*,*20*,*30*,*40*,*50*] [*1*::*nat*,*2*,*3*]
value *sublist-list* [*0*::*int*,*10*::*int*,*20*,*30*,*40*,*50*] [*1*::*nat*,*1*,*2*,*3*]
value [*nbe*] *sublist-list* [*0*::*int*,*10*::*int*,*20*,*30*,*40*,*50*] [*1*::*nat*,*1*,*2*,*3*,*10*]

lemma *sublist-list-length*: *length* (*sublist-list xs ys*) = *length ys*
⟨*proof*⟩

lemma *sublist-list-append*:
$\bigwedge$*zs*. *sublist-list xs* (*ys* @ *zs*) = *sublist-list xs ys* @ *sublist-list xs zs*
⟨*proof*⟩

lemma *sublist-list-Nil*: *sublist-list xs* [] =[]
⟨*proof*⟩

lemma *sublist-list-is-Nil-conv*:
(*sublist-list xs ys* = []) = (*ys* = [])
⟨*proof*⟩

lemma *sublist-list-eq-imp-length-eq*:
sublist-list xs ys = *sublist-list xs zs* $\Longrightarrow$ *length ys* = *length zs*
⟨*proof*⟩

lemma *sublist-list-nth*:
$\bigwedge$*n*. $n <$ *length ys* $\Longrightarrow$ *sublist-list xs ys* ! n = *xs* ! (*ys* ! n)
⟨*proof*⟩

lemma *take-drop-eq-sublist-list*:
$m + n \le length\ xs \Longrightarrow xs \uparrow m \downarrow n = sublist\text{-}list\ xs\ [m..{<}m{+}n]$
⟨*proof*⟩

primrec *sublist-list-if* :: $'a\ list \Rightarrow nat\ list \Rightarrow 'a\ list$
where
sublist-list-if xs [] = []
| *sublist-list-if xs* (*y* # *ys*) =
(*if y* < *length xs then* (*xs* ! *y*) # (*sublist-list-if xs ys*)
else (*sublist-list-if xs ys*))

value *sublist-list-if* [*0*::*int*,*10*::*int*,*20*,*30*,*40*,*50*] [*1*::*nat*,*2*,*3*]
value *sublist-list-if* [*0*::*int*,*10*::*int*,*20*,*30*,*40*,*50*] [*1*::*nat*,*1*,*2*,*3*]
value *sublist-list-if* [*0*::*int*,*10*::*int*,*20*,*30*,*40*,*50*] [*1*::*nat*,*1*,*2*,*3*,*10*]

lemma *sublist-list-if-sublist-list-filter-conv*: $\bigwedge xs.$
sublist-list-if xs ys = *sublist-list xs* (*filter* (λ*i*. *i* < *length xs*) *ys*)
⟨*proof*⟩

corollary *sublist-list-if-sublist-list-eq*: $\bigwedge xs.$
list-all (λ*i*. *i* < *length xs*) *ys* ⟹
sublist-list-if xs ys = *sublist-list xs ys*
⟨*proof*⟩

corollary *sublist-list-if-sublist-list-eq2*: $\bigwedge xs.$
∀ *n*<*length ys*. *ys* ! *n* < *length xs* ⟹
sublist-list-if xs ys = *sublist-list xs ys*
⟨*proof*⟩

lemma *sublist-list-if-Nil-left*: *sublist-list-if* [] *ys* = []
⟨*proof*⟩
lemma *sublist-list-if-Nil-right*: *sublist-list-if xs* [] = []
⟨*proof*⟩

lemma *sublist-list-if-length*:
length (*sublist-list-if xs ys*) = *length* (*filter* (λ*i*. *i* < *length xs*) *ys*)
⟨*proof*⟩
lemma *sublist-list-if-append*:
sublist-list-if xs (*ys* @ *zs*) = *sublist-list-if xs ys* @ *sublist-list-if xs zs*
⟨*proof*⟩
lemma *sublist-list-if-snoc*:
sublist-list-if xs (*ys* @ [*y*]) = *sublist-list-if xs ys* @ (*if y* < *length xs then* [*xs* ! *y*]
else [])
⟨*proof*⟩

lemma *sublist-list-if-is-Nil-conv*:
(*sublist-list-if xs ys* = []) = (*list-all* (λ*i*. *length xs* ≤ *i*) *ys*)

⟨*proof*⟩

lemma *sublist-list-if-nth*:
$n < length\ ((filter\ (\lambda i.\ i < length\ xs)\ ys)) \Longrightarrow$
$sublist\text{-}list\text{-}if\ xs\ ys\ !\ n = xs\ !\ ((filter\ (\lambda i.\ i < length\ xs)\ ys)\ !\ n)$
⟨*proof*⟩

lemma *take-drop-eq-sublist-list-if*:
$m + n \le length\ xs \Longrightarrow xs \uparrow m \downarrow n = sublist\text{-}list\text{-}if\ xs\ [m..<m+n]$
⟨*proof*⟩

lemma *nths-empty-conv*: $(nths\ xs\ I = []) = (\forall\, i{\in}I.\ length\ xs \le i)$
⟨*proof*⟩

lemma *nths-singleton2*: $nths\ xs\ \{y\} = (if\ y < length\ xs\ then\ [xs\ !\ y]\ else\ [])$
⟨*proof*⟩

lemma *nths-take-eq*:
$[\![\ finite\ I;\ Max\ I < n\ ]\!] \Longrightarrow nths\ (xs \downarrow n)\ I = nths\ xs\ I$
⟨*proof*⟩

lemma *nths-drop-eq*:
$n \le iMin\ I \Longrightarrow nths\ (xs \uparrow n)\ \{j.\ j + n \in I\} = nths\ xs\ I$
⟨*proof*⟩

lemma *nths-cut-less-eq*:
$length\ xs \le n \Longrightarrow nths\ xs\ (I \downarrow< n) = nths\ xs\ I$
⟨*proof*⟩

lemma *nths-disjoint-Un*:
$[\![\ finite\ A;\ Max\ A < iMin\ B\ ]\!] \Longrightarrow nths\ xs\ (A \cup B) = nths\ xs\ A\ @\ nths\ xs\ B$
⟨*proof*⟩

corollary *nths-disjoint-insert-left*:
$[\![\ finite\ I;\ x < iMin\ I\ ]\!] \Longrightarrow nths\ xs\ (insert\ x\ I) = nths\ xs\ \{x\}\ @\ nths\ xs\ I$
⟨*proof*⟩

corollary *nths-disjoint-insert-right*:
$[\![\ finite\ I;\ Max\ I < x\ ]\!] \Longrightarrow nths\ xs\ (insert\ x\ I) = nths\ xs\ I\ @\ nths\ xs\ \{x\}$
⟨*proof*⟩

lemma *nths-all*: $\{..<length\ xs\} \subseteq I \Longrightarrow nths\ xs\ I = xs$
⟨*proof*⟩

corollary *nths-UNIV*: $nths\ xs\ UNIV = xs$
⟨*proof*⟩

lemma *sublist-list-nths-eq*: $\bigwedge xs.$
$list\text{-}strict\text{-}asc\ ys \Longrightarrow sublist\text{-}list\text{-}if\ xs\ ys = nths\ xs\ (set\ ys)$

$\langle proof \rangle$

lemma *set-sublist-list-if*: $\bigwedge xs.\ set\ (sublist\text{-}list\text{-}if\ xs\ ys) = \{xs\ !\ i\ |i.\ i < length\ xs \wedge i \in set\ ys\}$
$\langle proof \rangle$

lemma *set-sublist-list*:
$list\text{-}all\ (\lambda i.\ i < length\ xs)\ ys \Longrightarrow$
$set\ (sublist\text{-}list\ xs\ ys) = \{xs\ !\ i\ |i.\ i < length\ xs \wedge i \in set\ ys\}$
$\langle proof \rangle$

lemma *set-sublist-list-if-eq-set-sublist*: $set\ (sublist\text{-}list\text{-}if\ xs\ ys) = set\ (nths\ xs\ (set\ ys))$
$\langle proof \rangle$

lemma *set-sublist-list-eq-set-sublist*:
$list\text{-}all\ (\lambda i.\ i < length\ xs)\ ys \Longrightarrow$
$set\ (sublist\text{-}list\ xs\ ys) = set\ (nths\ xs\ (set\ ys))$
$\langle proof \rangle$

9.1.5 Natural set images with lists

definition *f-image* :: $'a\ list \Rightarrow nat\ set \Rightarrow 'a\ set$ (**infixr** $`^f$ *90*)
where $xs\ `^f\ A \equiv \{y.\ \exists n\in A.\ n < length\ xs \wedge y = xs\ !\ n\}$

abbreviation *f-range* :: $'a\ list \Rightarrow 'a\ set$
where $f\text{-}range\ xs \equiv f\text{-}image\ xs\ UNIV$

lemma *f-image-eqI*[*simp*, *intro*]:
$[\![\ x = xs\ !\ n;\ n \in A;\ n < length\ xs\ ]\!] \Longrightarrow x \in xs\ `^f\ A$
$\langle proof \rangle$

lemma *f-imageI*: $[\![\ n \in A;\ n < length\ xs\ ]\!] \Longrightarrow xs\ !\ n \in xs\ `^f\ A$
$\langle proof \rangle$

lemma *rev-f-imageI*: $[\![\ n \in A;\ n < length\ xs;\ x = xs\ !\ n\ ]\!] \Longrightarrow x \in xs\ `^f\ A$
$\langle proof \rangle$

lemma *f-imageE*[*elim*!]:
$[\![\ x \in xs\ `^f\ A;\ \bigwedge n.\ [\![\ x = xs\ !\ n;\ n \in A;\ n < length\ xs\ ]\!] \Longrightarrow P\ ]\!] \Longrightarrow P$
$\langle proof \rangle$

lemma *f-image-Un*: $xs\ `^f\ (A \cup B) = xs\ `^f\ A \cup xs\ `^f\ B$
$\langle proof \rangle$

lemma *f-image-mono*: $A \subseteq B ==> xs\ `^f\ A \subseteq xs\ `^f\ B$
$\langle proof \rangle$

lemma *f-image-iff*: $(x \in xs\ `^f\ A) = (\exists n\in A.\ n < length\ xs \wedge x = xs\ !\ n)$

$\langle proof \rangle$

lemma *f-image-subset-iff*:
$(xs \text{ `}^f\ A \subseteq B) = (\forall n \in A.\ n < length\ xs \longrightarrow xs\ !\ n \in B)$
$\langle proof \rangle$

lemma *subset-f-image-iff*: $(B \subseteq xs \text{ `}^f\ A) = (\exists A' \subseteq A.\ B = xs \text{ `}^f\ A')$
$\langle proof \rangle$

lemma *f-image-subsetI*:
$[\![\bigwedge n.\ n \in A \wedge n < length\ xs \Longrightarrow xs\ !\ n \in B]\!] \Longrightarrow xs \text{ `}^f\ A \subseteq B$
$\langle proof \rangle$

lemma *f-image-empty*: $xs \text{ `}^f\ \{\} = \{\}$
$\langle proof \rangle$

lemma *f-image-insert-if*:
$xs \text{ `}^f\ (insert\ n\ A) = ($
$if\ n < length\ xs\ then\ insert\ (xs\ !\ n)\ (xs \text{ `}^f\ A)\ else\ (xs \text{ `}^f\ A))$
$\langle proof \rangle$

lemma *f-image-insert-eq1*:
$n < length\ xs \Longrightarrow xs \text{ `}^f\ (insert\ n\ A) = insert\ (xs\ !\ n)\ (xs \text{ `}^f\ A)$
$\langle proof \rangle$

lemma *f-image-insert-eq2*:
$length\ xs \le n \Longrightarrow xs \text{ `}^f\ (insert\ n\ A) = (xs \text{ `}^f\ A)$
$\langle proof \rangle$

lemma *insert-f-image*:
$[\![n \in A;\ n < length\ xs]\!] \Longrightarrow insert\ (xs\ !\ n)\ (xs \text{ `}^f\ A) = (xs \text{ `}^f\ A)$
$\langle proof \rangle$

lemma *f-image-is-empty*: $(xs \text{ `}^f\ A = \{\}) = (\{x.\ x \in A \wedge x < length\ xs\} = \{\})$
$\langle proof \rangle$

lemma *f-image-Collect*: $xs \text{ `}^f\ \{n.\ P\ n\} = \{xs\ !\ n\ |n.\ P\ n \wedge n < length\ xs\}$
$\langle proof \rangle$

lemma *f-image-eq-set*: $\forall n<length\ xs.\ n \in A \Longrightarrow xs \text{ `}^f\ A = set\ xs$
$\langle proof \rangle$
lemma *f-range-eq-set*: $f\text{-}range\ xs = set\ xs$
$\langle proof \rangle$

lemma *f-image-eq-set-nths*: $xs \text{ `}^f\ A = set\ (nths\ xs\ A)$
$\langle proof \rangle$
lemma *f-image-eq-set-sublist-list-if*: $xs \text{ `}^f\ (set\ ys) = set\ (sublist\text{-}list\text{-}if\ xs\ ys)$
$\langle proof \rangle$

lemma *f-image-eq-set-sublist-list*:
list-all $(\lambda i.\ i < \mathit{length}\ \mathit{xs})\ \mathit{ys} \Longrightarrow \mathit{xs}\ `^f\ (\mathit{set}\ \mathit{ys}) = \mathit{set}\ (\mathit{sublist\text{-}list}\ \mathit{xs}\ \mathit{ys})$
⟨*proof*⟩

lemma *f-range-eqI*: $[\![\ x = \mathit{xs}\ !\ n;\ n < \mathit{length}\ \mathit{xs}\ ]\!] \Longrightarrow x \in \mathit{f\text{-}range}\ \mathit{xs}$
⟨*proof*⟩

lemma *f-rangeI*: $n < \mathit{length}\ \mathit{xs} \Longrightarrow \mathit{xs}\ !\ n \in \mathit{f\text{-}range}\ \mathit{xs}$
⟨*proof*⟩

lemma *f-rangeE*[*elim?*]:
$[\![\ x \in \mathit{f\text{-}range}\ \mathit{xs};\ \bigwedge n.\ [\![\ n < \mathit{length}\ \mathit{xs};\ x = \mathit{xs}\ !\ n\ ]\!] \Longrightarrow P\ ]\!] \Longrightarrow P$
⟨*proof*⟩

9.1.6 Mapping lists of functions to lists

primrec *map-list* :: $('a \Rightarrow 'b)\ \mathit{list} \Rightarrow 'a\ \mathit{list} \Rightarrow 'b\ \mathit{list}$
where
map-list [] *xs* = []
| *map-list* $(f\ \#\ \mathit{fs})\ \mathit{xs} = f\ (\mathit{hd}\ \mathit{xs})\ \#\ \mathit{map\text{-}list}\ \mathit{fs}\ (\mathit{tl}\ \mathit{xs})$

lemma *map-list-Nil*: *map-list* [] *xs* = []
⟨*proof*⟩

lemma *map-list-Cons-Cons*:
map-list $(f\ \#\ \mathit{fs})\ (x\ \#\ \mathit{xs}) =$
$(f\ x)\ \#\ \mathit{map\text{-}list}\ \mathit{fs}\ \mathit{xs}$
⟨*proof*⟩

lemma *map-list-length*: $\bigwedge \mathit{xs}.$
length (*map-list fs xs*) = *length fs*
⟨*proof*⟩
corollary *map-list-empty-conv*:
(*map-list fs xs* = []) = (*fs* = [])
⟨*proof*⟩
corollary *map-list-not-empty-conv*:
(*map-list fs xs* $\neq$ []) = (*fs* $\neq$ [])
⟨*proof*⟩

lemma *map-list-nth*: $\bigwedge n\ \mathit{xs}.$
$[\![\ n < \mathit{length}\ \mathit{fs};\ n < \mathit{length}\ \mathit{xs}\ ]\!] \Longrightarrow$
$(\mathit{map\text{-}list}\ \mathit{fs}\ \mathit{xs}\ !\ n) =$
$(\mathit{fs}\ !\ n)\ (\mathit{xs}\ !\ n)$
⟨*proof*⟩

lemma *map-list-xs-take*: $\bigwedge n\ \mathit{xs}.$
$\mathit{length}\ \mathit{fs} \le n \Longrightarrow$
$\mathit{map\text{-}list}\ \mathit{fs}\ (\mathit{xs} \downarrow n) =$

$map\text{-}list\ fs\ xs$
$\langle proof \rangle$

lemma $map\text{-}list\text{-}take$: $\bigwedge n\ xs.$
$(map\text{-}list\ fs\ xs) \downarrow n =$
$(map\text{-}list\ (fs \downarrow n)\ xs)$
$\langle proof \rangle$
lemma $map\text{-}list\text{-}take\text{-}take$: $\bigwedge n\ xs.$
$(map\text{-}list\ fs\ xs) \downarrow n =$
$(map\text{-}list\ (fs \downarrow n)\ (xs \downarrow n))$
$\langle proof \rangle$
lemma $map\text{-}list\text{-}drop$: $\bigwedge n\ xs.$
$(map\text{-}list\ fs\ xs) \uparrow n =$
$(map\text{-}list\ (fs \uparrow n)\ (xs \uparrow n))$
$\langle proof \rangle$

lemma $map\text{-}list\text{-}append\text{-}append$: $\bigwedge xs1\ .$
$length\ fs1 = length\ xs1 \Longrightarrow$
$map\text{-}list\ (fs1\ @\ fs2)\ (xs1\ @\ xs2) =$
$map\text{-}list\ fs1\ xs1\ @$
$map\text{-}list\ fs2\ xs2$
$\langle proof \rangle$
lemma $map\text{-}list\text{-}snoc\text{-}snoc$:
$length\ fs = length\ xs \Longrightarrow$
$map\text{-}list\ (fs\ @\ [f])\ (xs\ @\ [x]) =$
$map\text{-}list\ fs\ xs\ @\ [f\ x]$
$\langle proof \rangle$
lemma $map\text{-}list\text{-}snoc$: $\bigwedge xs.$
$length\ fs < length\ xs \Longrightarrow$
$map\text{-}list\ (fs\ @\ [f])\ xs =$
$map\text{-}list\ fs\ xs\ @\ \ [f\ (xs\ !\ (length\ fs))]$
$\langle proof \rangle$

lemma $map\text{-}list\text{-}Cons\text{-}if$:
$map\text{-}list\ fs\ (x\ \#\ xs) =$
$(if\ (fs = [])\ then\ []\ else\ ($
$((hd\ fs)\ x)\ \#\ map\text{-}list\ (tl\ fs)\ xs))$
$\langle proof \rangle$
lemma $map\text{-}list\text{-}Cons\text{-}not\text{-}empty$:
$fs \neq [] \Longrightarrow$
$map\text{-}list\ fs\ (x\ \#\ xs) =$
$((hd\ fs)\ x)\ \#\ map\text{-}list\ (tl\ fs)\ xs$
$\langle proof \rangle$

lemma $map\text{-}eq\text{-}map\text{-}list\text{-}take$: $\bigwedge xs.$
$[\![\ length\ fs \le length\ xs;\ list\text{-}all\ (\lambda x.\ x = f)\ fs\ ]\!] \Longrightarrow$

$map\text{-}list\ fs\ xs = map\ f\ (xs \downarrow length\ fs)$
$\langle proof \rangle$
lemma *map-eq-map-list-take2*:
$[\![\ length\ fs = length\ xs;\ list\text{-}all\ (\lambda x.\ x = f)\ fs\ ]\!] \Longrightarrow$
$map\text{-}list\ fs\ xs = map\ f\ xs$
$\langle proof \rangle$
lemma *map-eq-map-list-replicate*:
$map\text{-}list\ (f^{length\ xs})\ xs = map\ f\ xs$
$\langle proof \rangle$

9.1.7 Mapping functions with two arguments to lists

primrec *map2* ::
— Function taking two parameters
$('a \Rightarrow 'b \Rightarrow 'c) \Rightarrow$
— Lists of parameters
$'a\ list \Rightarrow 'b\ list \Rightarrow$
$'c\ list$
where
$map2\ f\ []\ ys = []$
$|\ map2\ f\ (x \# xs)\ ys = f\ x\ (hd\ ys) \# map2\ f\ xs\ (tl\ ys)$

lemma *map2-map-list-conv*: $\bigwedge ys.\ map2\ f\ xs\ ys = map\text{-}list\ (map\ f\ xs)\ ys$
$\langle proof \rangle$

lemma *map2-Nil*: $map2\ f\ []\ ys = []$
$\langle proof \rangle$
lemma *map2-Cons-Cons*:
$map2\ f\ (x \# xs)\ (y \# ys) =$
$(f\ x\ y) \# map2\ f\ xs\ ys$
$\langle proof \rangle$

lemma *map2-length*: $\bigwedge ys.\ length\ (map2\ f\ xs\ ys) = length\ xs$
$\langle proof \rangle$
corollary *map2-empty-conv*:
$(map2\ f\ xs\ ys = []) = (xs = [])$
$\langle proof \rangle$
corollary *map2-not-empty-conv*:
$(map2\ f\ xs\ ys \neq []) = (xs \neq [])$
$\langle proof \rangle$

lemma *map2-nth*: $\bigwedge n\ ys.$
$[\![\ n < length\ xs;\ n < length\ ys\ ]\!] \Longrightarrow$
$(map2\ f\ xs\ ys\ !\ n) =$
$f\ (xs\ !\ n)\ (ys\ !\ n)$
$\langle proof \rangle$

lemma *map2-ys-take*: $\bigwedge n\ ys.$

$length\ xs \leq n \Longrightarrow$
$map2\ f\ xs\ (ys \downarrow n) =$
$map2\ f\ xs\ ys$
$\langle proof \rangle$

lemma *map2-take*: $\bigwedge n\ ys.$
$(map2\ f\ xs\ ys) \downarrow n =$
$(map2\ f\ (xs \downarrow n)\ ys)$
$\langle proof \rangle$

lemma *map2-take-take*: $\bigwedge n\ ys.$
$(map2\ f\ xs\ ys) \downarrow n =$
$(map2\ f\ (xs \downarrow n)\ (ys \downarrow n))$
$\langle proof \rangle$

lemma *map2-drop*: $\bigwedge n\ ys.$
$(map2\ f\ xs\ ys) \uparrow n =$
$(map2\ f\ (xs \uparrow n)\ (ys \uparrow n))$
$\langle proof \rangle$

lemma *map2-append-append*: $\bigwedge ys1\ .$
$length\ xs1\ = length\ ys1\ \Longrightarrow$
$map2\ f\ (xs1\ @\ xs2)\ (ys1\ @\ ys2) =$
$map2\ f\ xs1\ ys1\ @$
$map2\ f\ xs2\ ys2$
$\langle proof \rangle$

lemma *map2-snoc-snoc*:
$length\ xs = length\ ys \Longrightarrow$
$map2\ f\ (xs\ @\ [x])\ (ys\ @\ [y]) =$
$map2\ f\ xs\ ys\ @$
$[f\ x\ y]$
$\langle proof \rangle$

lemma *map2-snoc*: $\bigwedge ys.$
$length\ xs < length\ ys \Longrightarrow$
$map2\ f\ (xs\ @\ [x])\ ys =$
$map2\ f\ xs\ ys\ @$
$[f\ x\ (ys\ !\ (length\ xs))]$
$\langle proof \rangle$

lemma *map2-Cons-if*:
$map2\ f\ xs\ (y\ \#\ ys) =$
$(if\ (xs = [])\ then\ []\ else\ ($
$(f\ (hd\ xs)\ y)\ \#\ map2\ f\ (tl\ xs)\ ys))$
$\langle proof \rangle$
lemma *map2-Cons-not-empty*:
$xs \neq [] \Longrightarrow$

map2 f xs (*y* # *ys*) =
(*f* (*hd xs*) *y*) # *map2 f* (*tl xs*) *ys*
⟨*proof*⟩

lemma *map2-append1-take-drop*:
length xs1 ≤ *length ys* ⟹
map2 f (*xs1* @ *xs2*) *ys* =
map2 f xs1 (*ys* ↓ *length xs1*) @
map2 f xs2 (*ys* ↑ *length xs1*)
⟨*proof*⟩

lemma *map2-append2-take-drop*:
length ys1 ≤ *length xs* ⟹
map2 f xs (*ys1* @ *ys2*) =
map2 f (*xs* ↓ *length ys1*) *ys1* @
map2 f (*xs* ↑ *length ys1*) *ys2*
⟨*proof*⟩

lemma *map2-cong*:
⟦ *xs1* = *xs2*; *ys1* = *ys2*; *length xs2* ≤ *length ys2*;
⋀*x y*. ⟦ *x* ∈ *set xs2*; *y* ∈ *set ys2* ⟧ ⟹ *f x y* = *g x y* ⟧ ⟹
map2 f xs1 ys1 = *map2 g xs2 ys2*
⟨*proof*⟩

lemma *map2-eq-conv*:
length xs ≤ *length ys* ⟹
(*map2 f xs ys* = *map2 g xs ys*) = (∀ *i*<*length xs*. *f* (*xs* ! *i*) (*ys* ! *i*) = *g* (*xs* ! *i*) (*ys* ! *i*))
⟨*proof*⟩

lemma *map2-replicate*: *map2 f* x^n y^n = $(f\ x\ y)^n$
⟨*proof*⟩

lemma *map2-zip-conv*: ⋀*ys*.
length xs ≤ *length ys* ⟹
map2 f xs ys = *map* (λ(*x*,*y*). *f x y*) (*zip xs ys*)
⟨*proof*⟩

lemma *map2-rev*: ⋀*ys*.
length xs = *length ys* ⟹
rev (*map2 f xs ys*) = *map2 f* (*rev xs*) (*rev ys*)
⟨*proof*⟩

hide-const (**open**) *map2*

end

10 Set operations with results of type enat

theory *InfiniteSet2*
imports *SetInterval2*
begin

10.1 Set operations with *enat*

10.1.1 Basic definitions

definition $icard :: 'a\ set \Rightarrow enat$
where $icard\ A \equiv if\ finite\ A\ then\ enat\ (card\ A)\ else\ \infty$

10.2 Results for *icard*

lemma *icard-UNIV-nat*: $icard\ (UNIV::nat\ set) = \infty$
⟨*proof*⟩

lemma *icard-finite-conv*: $(icard\ A = enat\ (card\ A)) = finite\ A$
⟨*proof*⟩
lemma *icard-infinite-conv*: $(icard\ A = \infty) = infinite\ A$
⟨*proof*⟩

corollary *icard-finite*: $finite\ A \Longrightarrow icard\ A = enat\ (card\ A)$
⟨*proof*⟩
corollary *icard-infinite*[*simp*]: $infinite\ A \Longrightarrow icard\ A = \infty$
⟨*proof*⟩

lemma *icard-eq-enat-imp*: $icard\ A = enat\ n \Longrightarrow finite\ A$
⟨*proof*⟩
lemma *icard-eq-Infty-imp*: $icard\ A = \infty \Longrightarrow infinite\ A$
⟨*proof*⟩

lemma *icard-the-enat*: $finite\ A \Longrightarrow the\text{-}enat\ (icard\ A) = card\ A$
⟨*proof*⟩

lemma *icard-eq-enat-imp-card*: $icard\ A = enat\ n \Longrightarrow card\ A = n$
⟨*proof*⟩

lemma *icard-eq-enat-card-conv*: $0 < n \Longrightarrow (icard\ A = enat\ n) = (card\ A = n)$
⟨*proof*⟩

lemma *icard-empty*[*simp*]: $icard\ \{\} = 0$
⟨*proof*⟩
lemma *icard-empty-iff*: $(icard\ A = 0) = (A = \{\})$
⟨*proof*⟩
lemmas *icard-empty-iff-enat* = *icard-empty-iff*[*unfolded zero-enat-def*]

lemma *icard-not-empty-iff*: $(0 < icard\ A) = (A \neq \{\})$
⟨*proof*⟩

lemmas *icard-not-empty-iff-enat* = *icard-not-empty-iff*[*unfolded zero-enat-def*]

lemma *icard-singleton*: *icard* $\{a\}$ = *eSuc 0*
⟨*proof*⟩
lemmas *icard-singleton-enat*[*simp*] = *icard-singleton*[*unfolded zero-enat-def*]
lemma *icard-1-imp-singleton*: *icard* A = *eSuc 0* $\Longrightarrow \exists\, a.\ A = \{a\}$
⟨*proof*⟩
lemma *icard-1-singleton-conv*: (*icard* A = *eSuc 0*) = ($\exists\, a.\ A = \{a\}$)
⟨*proof*⟩

lemma *icard-insert-disjoint*: $x \notin A \Longrightarrow$ *icard* (*insert* x A) = *eSuc* (*icard* A)
⟨*proof*⟩

lemma *icard-insert-if*: *icard* (*insert* x A) = (*if* $x \in A$ *then icard* A *else eSuc* (*icard* A))
⟨*proof*⟩

lemmas *icard-0-eq* = *icard-empty-iff*

lemma *icard-Suc-Diff1*: $x \in A \Longrightarrow$ *eSuc* (*icard* ($A - \{x\}$)) = *icard* A
⟨*proof*⟩

lemma *icard-Diff-singleton*: $x \in A \Longrightarrow$ *icard* ($A - \{x\}$) = *icard* A − *1*
⟨*proof*⟩

lemma *icard-Diff-singleton-if*: *icard* ($A - \{x\}$) = (*if* $x \in A$ *then icard* A − *1 else icard* A)
⟨*proof*⟩

lemma *icard-insert*: *icard* (*insert* x A) = *eSuc* (*icard* ($A - \{x\}$))
⟨*proof*⟩

lemma *icard-insert-le*: *icard* A ≤ *icard* (*insert* x A)
⟨*proof*⟩

lemma *icard-mono*: $A \subseteq B \Longrightarrow$ *icard* A ≤ *icard* B
⟨*proof*⟩

lemma *not-icard-seteq*: $\exists$ (A::*nat set*) B. ($A \subseteq B \wedge$ *icard* B ≤ *icard* A $\wedge \neg\, A = B$)
⟨*proof*⟩

lemma *not-psubset-icard-mono*: $\exists$ (A::*nat set*) B. $A \subset B \wedge \neg$ *icard* A < *icard* B
⟨*proof*⟩

lemma *icard-Un-Int*: *icard* A + *icard* B = *icard* ($A \cup B$) + *icard* ($A \cap B$)
⟨*proof*⟩

lemma *icard-Un-disjoint*: $A \cap B = \{\} \Longrightarrow$ *icard* ($A \cup B$) = *icard* A + *icard* B

$\langle proof \rangle$

lemma *not-icard-Diff-subset*: $\exists (A::nat\ set)\ B.\ B \subseteq A \wedge \neg\ icard\ (A - B) = icard\ A - icard\ B$
$\langle proof \rangle$

lemma *not-icard-Diff1-less*: $\exists (A::nat\ set)x.\ x \in A \wedge \neg\ icard\ (A - \{x\}) < icard\ A$
$\langle proof \rangle$

lemma *not-icard-Diff2-less*: $\exists (A::nat\ set)x\ y.\ x \in A \wedge y \in A \wedge \neg\ icard\ (A - \{x\} - \{y\}) < icard\ A$
$\langle proof \rangle$

lemma *icard-Diff1-le*: $icard\ (A - \{x\}) \leq icard\ A$
$\langle proof \rangle$

lemma *icard-psubset*: $[\![\ A \subseteq B;\ icard\ A < icard\ B\ ]\!] \Longrightarrow A \subset B$
$\langle proof \rangle$

lemma *icard-partition*:
$[\![\ \bigwedge c.\ c \in C \Longrightarrow icard\ c = k;\ \bigwedge c1\ c2.\ [\![c1 \in C;\ c2 \in C;\ c1 \neq c2]\!] \Longrightarrow c1 \cap c2 = \{\}\ ]\!] \Longrightarrow$
$icard\ (\bigcup C) = k * icard\ C$
$\langle proof \rangle$

lemma *icard-image-le*: $icard\ (f\ `\ A) \leq icard\ A$
$\langle proof \rangle$

lemma *icard-image*: $inj\text{-}on\ f\ A \Longrightarrow icard\ (f\ `\ A) = icard\ A$
$\langle proof \rangle$

lemma *not-eq-icard-imp-inj-on*: $\exists (f::nat \Rightarrow nat)\ (A::nat\ set).\ icard\ (f\ `\ A) = icard\ A \wedge \neg\ inj\text{-}on\ f\ A$
$\langle proof \rangle$

lemma *not-inj-on-iff-eq-icard*: $\exists (f::nat \Rightarrow nat)\ (A::nat\ set).\ \neg\ (inj\text{-}on\ f\ A = (icard\ (f\ `\ A) = icard\ A))$
$\langle proof \rangle$

lemma *icard-inj-on-le*: $[\![\ inj\text{-}on\ f\ A;\ f\ `\ A \subseteq B\ ]\!] \Longrightarrow icard\ A \leq icard\ B$
$\langle proof \rangle$

lemma *icard-bij-eq*:
$[\![\ inj\text{-}on\ f\ A;\ f\ `\ A \subseteq B;\ inj\text{-}on\ g\ B;\ g\ `\ B \subseteq A\ ]\!] \Longrightarrow$
$icard\ A = icard\ B$
$\langle proof \rangle$

lemma *icard-cartesian-product*: $icard\ (A \times B) = icard\ A * icard\ B$
$\langle proof \rangle$

lemma *icard-cartesian-product-singleton*: $icard\ (\{x\} \times A) = icard\ A$
⟨*proof*⟩

lemma *icard-cartesian-product-singleton-right*: $icard\ (A \times \{x\}) = icard\ A$
⟨*proof*⟩

lemma
icard-lessThan: $icard\ \{..<u\} = enat\ u$ **and**
icard-atMost: $icard\ \{..u\} = enat\ (Suc\ u)$ **and**
icard-atLeastLessThan: $icard\ \{l..<u\} = enat\ (u - l)$ **and**
icard-atLeastAtMost: $icard\ \{l..u\} = enat\ (Suc\ u - l)$ **and**
icard-greaterThanAtMost: $icard\ \{l<..u\} = enat\ (u - l)$ **and**
icard-greaterThanLessThan: $icard\ \{l<..<u\} = enat\ (u - Suc\ l)$
⟨*proof*⟩

lemma *icard-atLeast*: $icard\ \{(u::nat)..\} = \infty$
⟨*proof*⟩

lemma *icard-greaterThan*: $icard\ \{(u::nat)<..\} = \infty$
⟨*proof*⟩

lemma
icard-atLeastZeroLessThan-int: $icard\ \{0..<u\} = enat\ (nat\ u)$ **and**
icard-atLeastLessThan-int: $icard\ \{l..<u\} = enat\ (nat\ (u - l))$ **and**
icard-atLeastAtMost-int: $icard\ \{l..u\} = enat\ (nat\ (u - l + 1))$ **and**
icard-greaterThanAtMost-int: $icard\ \{l<..u\} = enat\ (nat\ (u - l))$
⟨*proof*⟩

lemma *icard-atLeast-int*: $icard\ \{(u::int)..\} = \infty$
⟨*proof*⟩

lemma *icard-greaterThan-int*: $icard\ \{(u::int)<..\} = \infty$
⟨*proof*⟩

lemma *icard-atMost-int*: $icard\ \{..(u::int)\} = \infty$
⟨*proof*⟩

lemma *icard-lessThan-int*: $icard\ \{..<(u::int)\} = \infty$
⟨*proof*⟩

end

11 Additional definitions and results for lists

theory *ListInf*
imports *List2 ../CommonSet/InfiniteSet2*
begin

11.1 Infinite lists

We define infinite lists as functions over natural numbers, i. e., we use functions $nat \Rightarrow 'a$ as infinite lists over elements of $'a$. Mapping functions to intervals lists $[m..<n]$ yiels common finite lists.

11.1.1 Appending a functions to a list

type-synonym $'a\ ilist = nat \Rightarrow 'a$

definition *i-append* :: $'a\ list \Rightarrow 'a\ ilist \Rightarrow 'a\ ilist$ (**infixr** $\frown$ *65*)
where $xs \frown f \equiv \lambda n.\ if\ n < length\ xs\ then\ xs\ !\ n\ else\ f\ (n - length\ xs)$

Synonym for the lemma *fun-eq-iff* from the HOL library to unify lemma names for finite and infinite lists, providing *list-eq-iff* for finite and *ilist-eq-iff* for infinite lists.

lemmas *expand-ilist-eq* = *fun-eq-iff*
lemmas *ilist-eq-iff* = *expand-ilist-eq*

lemma *i-append-nth*: $(xs \frown f)\ n = (if\ n < length\ xs\ then\ xs\ !\ n\ else\ f\ (n - length\ xs))$
⟨*proof*⟩
lemma *i-append-nth1*[*simp*]: $n < length\ xs \Longrightarrow (xs \frown f)\ n = xs\ !\ n$
⟨*proof*⟩
lemma *i-append-nth2*[*simp*]: $length\ xs \le n \Longrightarrow (xs \frown f)\ n = f\ (n - length\ xs)$
⟨*proof*⟩
lemma *i-append-Nil*[*simp*]: $[] \frown f = f$
⟨*proof*⟩

lemma *i-append-assoc*[*simp*]: $xs \frown (ys \frown f) = (xs\ @\ ys) \frown f$
⟨*proof*⟩

lemma *i-append-Cons*: $(x\ \#\ xs) \frown f = [x] \frown (xs \frown f)$
⟨*proof*⟩

lemma *i-append-eq-i-append-conv*[*simp*]:
$length\ xs = length\ ys \Longrightarrow$
$(xs \frown f = ys \frown g) = (xs = ys \land f = g)$
⟨*proof*⟩

lemma *i-append-eq-i-append-conv2-aux*:
$[\![\ xs \frown f = ys \frown g;\ length\ xs \le length\ ys\ ]\!] \Longrightarrow$
$\exists\,zs.\ xs\ @\ zs = ys \land f = zs \frown g$
⟨*proof*⟩

lemma *i-append-eq-i-append-conv2*:
$(xs \frown f = ys \frown g) =$
$(\exists\,zs.\ xs = ys\ @\ zs \land zs \frown f = g \lor xs\ @\ zs = ys \land f = zs \frown g)$
⟨*proof*⟩

lemma *same-i-append-eq*[*iff*]: $(xs \frown f = xs \frown g) = (f = g)$
⟨*proof*⟩

lemma *NOT-i-append-same-eq*:
$\neg(\forall\, xs\ ys\ f.\ (xs \frown (f{::}(nat \Rightarrow nat)) = ys \frown f) = (xs = ys))$
⟨*proof*⟩

lemma *i-append-hd*: $(xs \frown f)\ 0 = (\mathit{if}\ xs = []\ \mathit{then}\ f\ 0\ \mathit{else}\ hd\ xs)$
⟨*proof*⟩

lemma *i-append-hd2*[*simp*]: $xs \neq [] \Longrightarrow (xs \frown f)\ 0 = hd\ xs$
⟨*proof*⟩

lemma *eq-Nil-i-appendI*: $f = g \Longrightarrow f = [] \frown g$
⟨*proof*⟩

lemma *i-append-eq-i-appendI*:
$[\![\ xs\ @\ xs' = ys;\ f = xs' \frown g\ ]\!] \Longrightarrow xs \frown f = ys \frown g$
⟨*proof*⟩

lemma *o-ext*:
$(\forall\, x.\ (x \in range\ h \longrightarrow f\ x = g\ x)) \Longrightarrow f \circ h = g \circ h$
⟨*proof*⟩

lemma *i-append-o*[*simp*]: $g \circ (xs \frown f) = (map\ g\ xs) \frown (g \circ f)$
⟨*proof*⟩

lemma *o-eq-conv*: $(f \circ h = g \circ h) = (\forall\, x{\in}range\ h.\ f\ x = g\ x)$
⟨*proof*⟩

lemma *o-cong*:
$[\![\ h = i;\ \bigwedge x.\ x \in range\ i \Longrightarrow f\ x = g\ x\ ]\!] \Longrightarrow f \circ h = f \circ i$
⟨*proof*⟩

lemma *ex-o-conv*: $(\exists\, h.\ g = f \circ h) = (\forall\, y{\in}range\ g.\ \exists\, x.\ y = f\ x)$
⟨*proof*⟩

lemma *o-inj-on*:
$[\![\ f \circ g = f \circ h;\ inj\text{-}on\ f\ (range\ g \cup range\ h)\ ]\!] \Longrightarrow g = h$
⟨*proof*⟩

lemma *inj-on-o-eq-o*:
$inj\text{-}on\ f\ (range\ g \cup range\ h) \Longrightarrow$
$(f \circ g = f \circ h) = (g = h)$
⟨*proof*⟩

lemma *o-injective*: $[\![f \circ g = f \circ h;\ inj\ f]\!] \Longrightarrow g = h$
⟨*proof*⟩

lemma *inj-o-eq-o*: $inj\ f \Longrightarrow (f \circ g = f \circ h) = (g = h)$
⟨*proof*⟩

lemma *inj-oI*: $inj\ f \Longrightarrow inj\ (\lambda g.\ f \circ g)$
⟨*proof*⟩

lemma *inj-oD*: $inj\ (\lambda g.\ f \circ g) \Longrightarrow inj\ f$
⟨*proof*⟩

lemma *inj-o*[*iff*]: $inj\ (\lambda g.\ f \circ g) = inj\ f$
⟨*proof*⟩

lemma *inj-on-oI*:
$inj\text{-}on\ f\ (\bigcup\ ((\lambda f.\ range\ f)\ `\ A)) \Longrightarrow inj\text{-}on\ (\lambda g.\ f \circ g)\ A$
⟨*proof*⟩

lemma *o-idI*: $\forall x.\ x \in range\ g \longrightarrow f\ x = x \Longrightarrow f \circ g = g$
⟨*proof*⟩

lemma *o-fun-upd*[*simp*]: $y \notin range\ g \Longrightarrow f\ (y := x) \circ g = f \circ g$
⟨*proof*⟩

lemma *range-i-append*[*simp*]: $range\ (xs \frown f) = set\ xs \cup range\ f$
⟨*proof*⟩

lemma *set-subset-i-append*: $set\ xs \subseteq range\ (xs \frown f)$
⟨*proof*⟩

lemma *range-subset-i-append*: $range\ f \subseteq range\ (xs \frown f)$
⟨*proof*⟩

lemma *range-ConsD*: $y \in range\ ([x] \frown f) \Longrightarrow y = x \lor y \in range\ f$
⟨*proof*⟩

lemma *range-o* [*simp*]: $range\ (f \circ g) = f\ `\ range\ g$
⟨*proof*⟩

lemma *in-range-conv-decomp*:
$(x \in range\ f) = (\exists xs\ g.\ f = xs \frown ([x] \frown g))$
⟨*proof*⟩

nth

lemma *i-append-nth-Cons-0*[*simp*]: $((x \# xs) \frown f)\ 0 = x$
⟨*proof*⟩

lemma *i-append-nth-Cons-Suc*[*simp*]:

$((x \# xs) \frown f)\ (Suc\ n) = (xs \frown f)\ n$
⟨*proof*⟩

lemma *i-append-nth-Cons*:
$([x] \frown f)\ n = (case\ n\ of\ 0 \Rightarrow x \mid Suc\ k \Rightarrow f\ k)$
⟨*proof*⟩

lemma *i-append-nth-Cons'*:
$([x] \frown f)\ n = (if\ n = 0\ then\ x\ else\ f\ (n - Suc\ 0))$
⟨*proof*⟩

lemma *i-append-nth-length*[*simp*]: $(xs \frown f)\ (length\ xs) = f\ 0$
⟨*proof*⟩

lemma *i-append-nth-length-plus*[*simp*]: $(xs \frown f)\ (length\ xs + n) = f\ n$
⟨*proof*⟩

lemma *range-iff*: $(y \in range\ f) = (\exists x.\ y = f\ x)$
⟨*proof*⟩

lemma *range-ball-nth*: $\forall y \in range\ f.\ P\ y \Longrightarrow P\ (f\ x)$
⟨*proof*⟩

lemma *all-nth-imp-all-range*: $[\![\ \forall x.\ P\ (f\ x); y \in range\ f\ ]\!] \Longrightarrow P\ y$
⟨*proof*⟩

lemma *all-range-conv-all-nth*: $(\forall y \in range\ f.\ P\ y) = (\forall x.\ P\ (f\ x))$
⟨*proof*⟩

lemma *i-append-update1*:
$n < length\ xs \Longrightarrow (xs \frown f)\ (n := x) = xs[n := x] \frown f$
⟨*proof*⟩

lemma *i-append-update2*:
$length\ xs \le n \Longrightarrow (xs \frown f)\ (n := x) = xs \frown (f(n - length\ xs := x))$
⟨*proof*⟩

lemma *i-append-update*:
$(xs \frown f)\ (n := x) =$
$(if\ n < length\ xs\ then\ xs[n := x] \frown f$
$else\ xs \frown (f(n - length\ xs := x)))$
⟨*proof*⟩

lemma *i-append-update-length*[*simp*]:
$(xs \frown f)\ (length\ xs := y) = xs \frown (f(0 := y))$
⟨*proof*⟩

lemma *range-update-subset-insert*:
$range\ (f(n := x)) \subseteq insert\ x\ (range\ f)$

⟨*proof*⟩

lemma *range-update-subsetI*:
$\llbracket$ *range* $f \subseteq A$; $x \in A$ $\rrbracket \Longrightarrow$ *range* $(f(n := x)) \subseteq A$
⟨*proof*⟩

lemma *range-update-memI*: $x \in$ *range* $(f(n := x))$
⟨*proof*⟩

11.1.2 *take* and *drop* for infinite lists

The *i-take* operator takes the first n elements of an infinite list, i.e. *i-take* f $n = [f\ 0, f\ 1, \ldots, f\ (n-1)]$. The *i-drop* operator drops the first n elements of an infinite list, i.e. (*i-take* f n) $0 = f\ n$, (*i-take* f n) $1 = f\ (n + 1)$, ...

definition *i-take* :: *nat* ⇒ *'a ilist* ⇒ *'a list*
where *i-take* n f ≡ *map* f $[0..<n]$
definition *i-drop* :: *nat* ⇒ *'a ilist* ⇒ *'a ilist*
where *i-drop* n f ≡ $(\lambda x.\ f\ (n + x))$

abbreviation *i-take'* :: *'a ilist* ⇒ *nat* ⇒ *'a list* (**infixl** ⇓ *100*)
where $f \Downarrow n$ ≡ *i-take* n f
abbreviation *i-drop'* :: *'a ilist* ⇒ *nat* ⇒ *'a ilist* (**infixl** ⇑ *100*)
where $f \Uparrow n$ ≡ *i-drop* n f

lemma $f \Downarrow n =$ *map* f $[0..<n]$
⟨*proof*⟩
lemma $f \Uparrow n = (\lambda x.\ f\ (n + x))$
⟨*proof*⟩

Basic results for *i-take* and *i-drop*

lemma *i-take-first*: $f \Downarrow Suc\ 0 = [f\ 0]$
⟨*proof*⟩

lemma *i-drop-i-take-1*: $f \Uparrow n \Downarrow Suc\ 0 = [f\ n]$
⟨*proof*⟩

lemma *i-take-take-eq1*: $m \le n \Longrightarrow (f \Downarrow n) \downarrow m = f \Downarrow m$
⟨*proof*⟩

lemma *i-take-take-eq2*: $n \le m \Longrightarrow (f \Downarrow n) \downarrow m = f \Downarrow n$
⟨*proof*⟩

lemma *i-take-take*[*simp*]: $(f \Downarrow n) \downarrow m = f \Downarrow min\ n\ m$
⟨*proof*⟩

lemma *i-drop-nth*[*simp*]: $(s \Uparrow n)\ x = s\ (n + x)$
⟨*proof*⟩

lemma *i-drop-nth-sub*: $n \le x \Longrightarrow (s \Uparrow n)\ (x - n) = s\ x$
⟨*proof*⟩

theorem *i-take-nth*[*simp*]: $i < n \Longrightarrow (f \Downarrow n)\ !\ i = f\ i$
⟨*proof*⟩

lemma *i-take-length*[*simp*]: $length\ (f \Downarrow n) = n$
⟨*proof*⟩

lemma *i-take-0*[*simp*]: $f \Downarrow 0 = []$
⟨*proof*⟩

lemma *i-drop-0*[*simp*]: $f \Uparrow 0 = f$
⟨*proof*⟩

lemma *i-take-eq-Nil*[*simp*]: $(f \Downarrow n = []) = (n = 0)$
⟨*proof*⟩

lemma *i-take-not-empty-conv*: $(f \Downarrow n \neq []) = (0 < n)$
⟨*proof*⟩

lemma *last-i-take*: $last\ (f \Downarrow Suc\ n) = f\ n$
⟨*proof*⟩

lemma *last-i-take2*: $0 < n \Longrightarrow last\ (f \Downarrow n) = f\ (n - Suc\ 0)$
⟨*proof*⟩

lemma *nth-0-i-drop*: $(f \Uparrow n)\ 0 = f\ n$
⟨*proof*⟩

lemma *i-take-const*[*simp*]: $(\lambda n.\ x) \Downarrow i = replicate\ i\ x$
⟨*proof*⟩

lemma *i-drop-const*[*simp*]: $(\lambda n.\ x) \Uparrow i = (\lambda n.\ x)$
⟨*proof*⟩

lemma *i-append-i-take-eq1*:
$n \le length\ xs \Longrightarrow (xs \frown f) \Downarrow n = xs \downarrow n$
⟨*proof*⟩

lemma *i-append-i-take-eq2*:
$length\ xs \le n \Longrightarrow (xs \frown f) \Downarrow n = xs\ @\ (f \Downarrow (n - length\ xs))$
⟨*proof*⟩

lemma *i-append-i-take-if*:
$(xs \frown f) \Downarrow n = (if\ n \le length\ xs\ then\ xs \downarrow n\ else\ xs\ @\ (f \Downarrow (n - length\ xs)))$
⟨*proof*⟩

lemma *i-append-i-take*[*simp*]:

$(xs \frown f) \Downarrow n = (xs \downarrow n) @ (f \Downarrow (n - \mathit{length}\ xs))$
⟨*proof*⟩

lemma *i-append-i-drop-eq1*:
$n \le \mathit{length}\ xs \Longrightarrow (xs \frown f) \Uparrow n = (xs \uparrow n) \frown f$
⟨*proof*⟩

lemma *i-append-i-drop-eq2*:
$\mathit{length}\ xs \le n \Longrightarrow (xs \frown f) \Uparrow n = f \Uparrow (n - \mathit{length}\ xs)$
⟨*proof*⟩

lemma *i-append-i-drop-if*:
$(xs \frown f) \Uparrow n = (\mathit{if}\ n < \mathit{length}\ xs\ \mathit{then}\ (xs \uparrow n) \frown f\ \mathit{else}\ f \Uparrow (n - \mathit{length}\ xs))$
⟨*proof*⟩

lemma *i-append-i-drop*[*simp*]: $(xs \frown f) \Uparrow n = (xs \uparrow n) \frown (f \Uparrow (n - \mathit{length}\ xs))$
⟨*proof*⟩

lemma *i-append-i-take-i-drop-id*[*simp*]: $(f \Downarrow n) \frown (f \Uparrow n) = f$
⟨*proof*⟩

lemma *ilist-i-take-i-drop-imp-eq*:
$[\![\, f \Downarrow n = g \Downarrow n;\ f \Uparrow n = g \Uparrow n \,]\!] \Longrightarrow f = g$
⟨*proof*⟩

lemma *ilist-i-take-i-drop-eq-conv*:
$(f = g) = (\exists\, n.\ (f \Downarrow n = g \Downarrow n \wedge f \Uparrow n = g \Uparrow n))$
⟨*proof*⟩

lemma *ilist-i-take-eq-conv*: $(f = g) = (\forall\, n.\ f \Downarrow n = g \Downarrow n)$
⟨*proof*⟩

lemma *ilist-i-drop-eq-conv*: $(f = g) = (\forall\, n.\ f \Uparrow n = g \Uparrow n)$
⟨*proof*⟩

lemma *i-take-the-conv*:
$f \Downarrow k = (\mathit{THE}\ xs.\ \mathit{length}\ xs = k \wedge (\exists\, g.\ xs \frown g = f))$
⟨*proof*⟩

lemma *i-drop-the-conv*:
$f \Uparrow k = (\mathit{THE}\ g.\ (\exists\, xs.\ \mathit{length}\ xs = k \wedge xs \frown g = f))$
⟨*proof*⟩

lemma *i-take-Suc-append*[*simp*]:
$((x \# xs) \frown f) \Downarrow \mathit{Suc}\ n = x \# ((xs \frown f) \Downarrow n)$
⟨*proof*⟩

corollary *i-take-Suc-Cons*: $([x] \frown f) \Downarrow \mathit{Suc}\ n = x \# (f \Downarrow n)$
⟨*proof*⟩

lemma *i-drop-Suc-append*[*simp*]: $((x \# xs) \frown f) \Uparrow Suc\ n = ((xs \frown f) \Uparrow n)$
⟨*proof*⟩

corollary *i-drop-Suc-Cons*: $([x] \frown f) \Uparrow Suc\ n = f \Uparrow n$
⟨*proof*⟩

lemma *i-take-Suc*: $f \Downarrow Suc\ n = f\ 0 \# (f \Uparrow Suc\ 0 \Downarrow n)$
⟨*proof*⟩

lemma *i-take-Suc-conv-app-nth*: $f \Downarrow Suc\ n = (f \Downarrow n)\ @\ [f\ n]$
⟨*proof*⟩

lemma *i-drop-i-drop*[*simp*]: $s \Uparrow a \Uparrow b = s \Uparrow (a + b)$
⟨*proof*⟩

corollary *i-drop-Suc*: $f \Uparrow Suc\ 0 \Uparrow n = f \Uparrow Suc\ n$
⟨*proof*⟩

lemma *i-take-commute*: $s \Downarrow a \downarrow b = s \Downarrow b \downarrow a$
⟨*proof*⟩

lemma *i-drop-commute*: $s \Uparrow a \Uparrow b = s \Uparrow b \Uparrow a$
⟨*proof*⟩

corollary *i-drop-tl*: $f \Uparrow Suc\ 0 \Uparrow n = f \Uparrow n \Uparrow Suc\ 0$
⟨*proof*⟩

lemma *nth-via-i-drop*: $(f \Uparrow n)\ 0 = x \Longrightarrow f\ n = x$
⟨*proof*⟩

lemma *i-drop-Suc-conv-tl*: $[f\ n] \frown (f \Uparrow Suc\ n) = f \Uparrow n$
⟨*proof*⟩

lemma *i-drop-Suc-conv-tl'*: $([f\ n] \frown f) \Uparrow Suc\ n = f \Uparrow n$
⟨*proof*⟩

lemma *i-take-i-drop*: $f \Uparrow m \Downarrow n = f \Downarrow (n + m) \uparrow m$
⟨*proof*⟩

Appending an interval of a function

lemma *i-take-int-append*:
$m \le n \Longrightarrow (f \Downarrow m)\ @\ map\ f\ [m..<n] = f \Downarrow n$
⟨*proof*⟩

lemma *i-take-drop-map-empty-iff*: $(f \Downarrow n \uparrow m = []) = (n \le m)$
⟨*proof*⟩

lemma *i-take-drop-map*: $f \Downarrow n \uparrow m = map\ f\ [m..<n]$

$\langle proof \rangle$

corollary *i-take-drop-append*[*simp*]:
$m \le n \Longrightarrow (f \Downarrow m) @ (f \Downarrow n \uparrow m) = f \Downarrow n$
$\langle proof \rangle$

lemma *i-take-drop*: $f \Downarrow n \uparrow m = f \Uparrow m \Downarrow (n - m)$
$\langle proof \rangle$

lemma *i-take-o*[*simp*]: $(f \circ g) \Downarrow n = map\ f\ (g \Downarrow n)$
$\langle proof \rangle$

lemma *i-drop-o*[*simp*]: $(f \circ g) \Uparrow n = f \circ (g \Uparrow n)$
$\langle proof \rangle$

lemma *set-i-take-subset*: $set\ (f \Downarrow n) \subseteq range\ f$
$\langle proof \rangle$

lemma *range-i-drop-subset*: $range\ (f \Uparrow n) \subseteq range\ f$
$\langle proof \rangle$

lemma *in-set-i-takeD*: $x \in set\ (f \Downarrow n) \Longrightarrow x \in range\ f$
$\langle proof \rangle$

lemma *in-range-i-takeD*: $x \in range\ (f \Uparrow n) \Longrightarrow x \in range\ f$
$\langle proof \rangle$

lemma *i-append-eq-conv-conj*:
$((xs \frown f) = g) = (xs = g \Downarrow length\ xs \wedge f = g \Uparrow length\ xs)$
$\langle proof \rangle$

lemma *i-take-add*: $f \Downarrow (i + j) = (f \Downarrow i) @ (f \Uparrow i \Downarrow j)$
$\langle proof \rangle$

lemma *i-append-eq-i-append-conv-if-aux*:
$length\ xs \le length\ ys \Longrightarrow$
$(xs \frown f = ys \frown g) = (xs = ys \downarrow length\ xs \wedge f = (ys \uparrow length\ xs) \frown g)$
$\langle proof \rangle$

lemma *i-append-eq-i-append-conv-if*:
$(xs \frown f = ys \frown g) =$
$(if\ length\ xs \le length\ ys$
$\ then\ xs = ys \downarrow length\ xs \wedge f = (ys \uparrow length\ xs) \frown g$
$\ else\ xs \downarrow length\ ys = ys \wedge (xs \uparrow length\ ys) \frown f = g)$
$\langle proof \rangle$

lemma *i-take-hd-i-drop*: $(f \Downarrow n) @ [(f \Uparrow n)\ 0] = f \Downarrow Suc\ n$
$\langle proof \rangle$

lemma *id-i-take-nth-i-drop*: $f = (f \Downarrow n) \frown (([f\ n] \frown f) \Uparrow Suc\ n)$
⟨*proof*⟩

lemma *upd-conv-i-take-nth-i-drop*:
$f\ (n := x) = (f \Downarrow n) \frown ([x] \frown (f \Uparrow Suc\ n))$
⟨*proof*⟩

theorem *i-take-induct*:
$[\![\ P\ (f \Downarrow 0);\ \bigwedge n.\ P\ (f \Downarrow n) \Longrightarrow P\ (\ f \Downarrow Suc\ n)\ ]\!] \Longrightarrow P\ (\ f \Downarrow n)$
⟨*proof*⟩

theorem *take-induct*[*rule-format*]:
$[\![\ P\ (s \downarrow 0);$
$\bigwedge n.\ [\![\ Suc\ n < length\ s;\ P\ (s \downarrow n)\ ]\!] \Longrightarrow P\ (\ s \downarrow Suc\ n);$
$i < length\ s]\!]$
$\Longrightarrow P\ (s \downarrow i)$
⟨*proof*⟩

theorem *i-drop-induct*:
$[\![\ P\ (f \Uparrow 0);\ \bigwedge n.\ P\ (f \Uparrow n) \Longrightarrow P\ (\ f \Uparrow Suc\ n)\ ]\!] \Longrightarrow P\ (\ f \Uparrow n)$
⟨*proof*⟩

theorem *f-drop-induct*[*rule-format*]:
$[\![\ P\ (s \uparrow 0);$
$\bigwedge n.\ [\![\ Suc\ n < length\ s;\ P\ (s \uparrow n)\ ]\!] \Longrightarrow P\ (\ s \uparrow Suc\ n);$
$i < length\ s]\!]$
$\Longrightarrow P\ (s \uparrow i)$
⟨*proof*⟩

lemma *i-take-drop-eq-map*: $f \Uparrow m \Downarrow n = map\ f\ [m..{<}m{+}n]$
⟨*proof*⟩

lemma *o-eq-i-append-imp*:
$f \circ g = ys \frown i \Longrightarrow$
$\exists xs\ h.\ g = xs \frown h \wedge map\ f\ xs = ys \wedge f \circ h = i$
⟨*proof*⟩

corollary *o-eq-i-append-conv*:
$(f \circ g = ys \frown i) =$
$(\exists xs\ h.\ g = xs \frown h \wedge map\ f\ xs = ys \wedge f \circ h = i)$
⟨*proof*⟩
corollary *i-append-eq-o-conv*:
$(ys \frown i = f \circ g) =$
$(\exists xs\ h.\ g = xs \frown h \wedge map\ f\ xs = ys \wedge f \circ h = i)$
⟨*proof*⟩

11.1.3 *zip* for infinite lists

definition $\mathit{i\text{-}zip} :: {}'a\ \mathit{ilist} \Rightarrow {}'b\ \mathit{ilist} \Rightarrow ({}'a \times {}'b)\ \mathit{ilist}$
where $\mathit{i\text{-}zip}\ f\ g \equiv \lambda n.\ (f\ n,\ g\ n)$

lemma *i-zip-nth*: $(\mathit{i\text{-}zip}\ f\ g)\ n = (f\ n,\ g\ n)$
⟨*proof*⟩

lemma *i-zip-swap*: $(\lambda(y,\ x).\ (x,\ y)) \circ \mathit{i\text{-}zip}\ g\ f = \mathit{i\text{-}zip}\ f\ g$
⟨*proof*⟩

lemma *i-zip-i-take*: $(\mathit{i\text{-}zip}\ f\ g) \Downarrow n = \mathit{zip}\ (f \Downarrow n)\ (g \Downarrow n)$
⟨*proof*⟩

lemma *i-zip-i-drop*: $(\mathit{i\text{-}zip}\ f\ g) \Uparrow n = \mathit{i\text{-}zip}\ (f \Uparrow n)\ (g \Uparrow n)$
⟨*proof*⟩

lemma *fst-o-izip*: $\mathit{fst} \circ (\mathit{i\text{-}zip}\ f\ g) = f$
⟨*proof*⟩

lemma *snd-o-i-zip*: $\mathit{snd} \circ (\mathit{i\text{-}zip}\ f\ g) = g$
⟨*proof*⟩

lemma *update-i-zip*:
$(\mathit{i\text{-}zip}\ f\ g)(n := \mathit{xy}) = \mathit{i\text{-}zip}\ (f(n := \mathit{fst}\ \mathit{xy}))\ (g(n := \mathit{snd}\ \mathit{xy}))$
⟨*proof*⟩

lemma *i-zip-Cons-Cons*:
$\mathit{i\text{-}zip}\ ([x] \frown f)\ ([y] \frown g) = [(x,\ y)] \frown (\mathit{i\text{-}zip}\ f\ g)$
⟨*proof*⟩

lemma *i-zip-i-append1*:
$\mathit{i\text{-}zip}\ (\mathit{xs} \frown f)\ g = \mathit{zip}\ \mathit{xs}\ (g \Downarrow \mathit{length}\ \mathit{xs}) \frown (\mathit{i\text{-}zip}\ f\ (g \Uparrow \mathit{length}\ \mathit{xs}))$
⟨*proof*⟩

lemma *i-zip-i-append2*:
$\mathit{i\text{-}zip}\ f\ (\mathit{ys} \frown g) = \mathit{zip}\ (f \Downarrow \mathit{length}\ \mathit{ys})\ \mathit{ys} \frown (\mathit{i\text{-}zip}\ (f \Uparrow \mathit{length}\ \mathit{ys})\ g)$
⟨*proof*⟩

lemma *i-zip-append*:
$\mathit{length}\ \mathit{xs} = \mathit{length}\ \mathit{ys} \Longrightarrow$
$\mathit{i\text{-}zip}\ (\mathit{xs} \frown f)\ (\mathit{ys} \frown g) = \mathit{zip}\ \mathit{xs}\ \mathit{ys} \frown (\mathit{i\text{-}zip}\ f\ g)$
⟨*proof*⟩

lemma *i-zip-range*: $\mathit{range}\ (\mathit{i\text{-}zip}\ f\ g) = \{\ (f\ n,\ g\ n)|\ n.\ \mathit{True}\ \}$
⟨*proof*⟩

lemma *i-zip-update*:
$\mathit{i\text{-}zip}\ (f(n := x))\ (g(n := y)) = (\mathit{i\text{-}zip}\ f\ g)(\ n := (x,\ y))$
⟨*proof*⟩

lemma *i-zip-const*: $i\text{-}zip\ (\lambda n.\ x)\ (\lambda n.\ y) = (\lambda n.\ (x,\ y))$
⟨*proof*⟩

11.1.4 Mapping functions with two arguments to infinite lists

definition *i-map2* ::
— Function taking two parameters
$('a \Rightarrow 'b \Rightarrow 'c) \Rightarrow$
— Lists of parameters
$'a\ ilist \Rightarrow 'b\ ilist \Rightarrow$
$'c\ ilist$
where
$i\text{-}map2\ f\ xs\ ys \equiv \lambda n.\ f\ (xs\ n)\ (ys\ n)$

lemma *i-map2-nth*: $(i\text{-}map2\ f\ xs\ ys)\ n = f\ (xs\ n)\ (ys\ n)$
⟨*proof*⟩

lemma *i-map2-Cons-Cons*:
$i\text{-}map2\ f\ ([x] \frown xs)\ ([y] \frown ys) =$
$[f\ x\ y] \frown (i\text{-}map2\ f\ xs\ ys)$
⟨*proof*⟩

lemma *i-map2-take-ge*:
$n \le n1 \Longrightarrow$
$i\text{-}map2\ f\ xs\ ys \Downarrow n =$
$map2\ f\ (xs \Downarrow n)\ (ys \Downarrow n1)$
⟨*proof*⟩
lemma *i-map2-take-take*:
$i\text{-}map2\ f\ xs\ ys \Downarrow n =$
$map2\ f\ (xs \Downarrow n)\ (ys \Downarrow n)$
⟨*proof*⟩

lemma *i-map2-drop*:
$(i\text{-}map2\ f\ xs\ ys) \Uparrow n =$
$(i\text{-}map2\ f\ (xs \Uparrow n)\ (ys \Uparrow n))$
⟨*proof*⟩

lemma *i-map2-append-append*:
$length\ xs1 = length\ ys1 \Longrightarrow$
$i\text{-}map2\ f\ (xs1 \frown xs)\ (ys1 \frown ys) =$
$map2\ f\ xs1\ ys1 \frown i\text{-}map2\ f\ xs\ ys$
⟨*proof*⟩

lemma *i-map2-Cons-left*:
$i\text{-}map2\ f\ ([x] \frown xs)\ ys =$
$[f\ x\ (ys\ 0)] \frown i\text{-}map2\ f\ xs\ (ys \Uparrow Suc\ 0)$
⟨*proof*⟩
lemma *i-map2-Cons-right*:

$i\text{-}map2\ f\ xs\ ([y] \frown ys) =$
$[f\ (xs\ 0)\ y] \frown i\text{-}map2\ f\ (xs \Uparrow Suc\ 0)\ ys$
$\langle proof \rangle$

lemma *i-map2-append-take-drop-left*:
$i\text{-}map2\ f\ (xs1 \frown xs)\ ys =$
$map2\ f\ xs1\ (ys \Downarrow length\ xs1) \frown$
$i\text{-}map2\ f\ xs\ (ys \Uparrow length\ xs1)$
$\langle proof \rangle$
lemma *i-map2-append-take-drop-right*:
$i\text{-}map2\ f\ xs\ (ys1 \frown ys) =$
$map2\ f\ (xs \Downarrow length\ ys1)\ ys1 \frown$
$i\text{-}map2\ f\ (xs \Uparrow length\ ys1)\ ys$
$\langle proof \rangle$

lemma *i-map2-cong*:
$[\![xs1 = xs2;\ ys1 = ys2;$
$\bigwedge x\ y.\ [\![x \in range\ xs2;\ y \in range\ ys2]\!] \Longrightarrow f\ x\ y = g\ x\ y]\!] \Longrightarrow$
$i\text{-}map2\ f\ xs1\ ys1 = i\text{-}map2\ g\ xs2\ ys2$
$\langle proof \rangle$

lemma *i-map2-eq-conv*:
$(i\text{-}map2\ f\ xs\ ys = i\text{-}map2\ g\ xs\ ys) = (\forall i.\ f\ (xs\ i)\ (ys\ i) = g\ (xs\ i)\ (ys\ i))$
$\langle proof \rangle$

lemma *i-map2-replicate*: $i\text{-}map2\ f\ (\lambda n.\ x)\ (\lambda n.\ y) = (\lambda n.\ f\ x\ y)$
$\langle proof \rangle$

lemma *i-map2-i-zip-conv*:
$i\text{-}map2\ f\ xs\ ys = (\lambda(x,y).\ f\ x\ y) \circ (i\text{-}zip\ xs\ ys)$
$\langle proof \rangle$

11.2 Generalised lists as combination of finite and infinite lists

11.2.1 Basic definitions

datatype $(gset: 'a)\ glist = FL\ 'a\ list \mid IL\ 'a\ ilist$ **for** $map: gmap$

definition $glength :: 'a\ glist \Rightarrow enat$
where
$glength\ a \equiv case\ a\ of$
$FL\ xs \Rightarrow enat\ (length\ xs) \mid$
$IL\ f \Rightarrow \infty$

definition $gCons :: 'a \Rightarrow 'a\ glist \Rightarrow 'a\ glist$ (**infixr** $\#_g$ *65*)
where
$x\ \#_g\ a \equiv case\ a\ of$
$FL\ xs \Rightarrow FL\ (x\ \#\ xs) \mid$

$IL\ g\ \Rightarrow IL\ ([x] \frown g)$

definition *gappend* :: *'a glist* ⇒ *'a glist* ⇒ *'a glist* (**infixr** $@_g$ *65*)
where
$gappend\ a\ b \equiv case\ a\ of$
$FL\ xs \Rightarrow (case\ b\ of\ FL\ ys \Rightarrow FL\ (xs\ @\ ys) \mid IL\ f \Rightarrow IL\ (xs \frown f)) \mid$
$IL\ f\ \Rightarrow IL\ f$

definition *gtake* :: *enat* ⇒ *'a glist* ⇒ *'a glist*
where
$gtake\ n\ a \equiv case\ n\ of$
$enat\ m \Rightarrow FL\ (case\ a\ of$
$FL\ xs \Rightarrow xs \downarrow m \mid$
$IL\ f\ \Rightarrow f \Downarrow m) \mid$
$\infty \Rightarrow a$

definition *gdrop* :: *enat* ⇒ *'a glist* ⇒ *'a glist*
where
$gdrop\ n\ a \equiv case\ n\ of$
$enat\ m \Rightarrow (case\ a\ of$
$FL\ xs \Rightarrow FL\ (xs \uparrow m) \mid$
$IL\ f\ \Rightarrow IL\ (f \Uparrow m)) \mid$
$\infty \Rightarrow FL\ []$

definition *gnth* :: *'a glist* ⇒ *nat* ⇒ *'a* (**infixl** $!_g$ *100*)
where
$a\ !_g\ n \equiv case\ a\ of$
$FL\ xs \Rightarrow xs\ !\ n \mid$
$IL\ f\ \Rightarrow f\ n$

abbreviation *g-take'* :: *'a glist* ⇒ *enat* ⇒ *'a glist* (**infixl** $\downarrow_g$ *100*)
where $a \downarrow_g n \equiv gtake\ n\ a$

abbreviation *g-drop'* :: *'a glist* ⇒ *enat* ⇒ *'a glist* (**infixl** $\uparrow_g$ *100*)
where $a \uparrow_g n \equiv gdrop\ n\ a$

11.2.2 *glength*

lemma *glength-fin*[*simp*]: $glength\ (FL\ xs) = enat\ (length\ xs)$
⟨*proof*⟩

lemma *glength-infin*[*simp*]: $glength\ (IL\ f) = \infty$
⟨*proof*⟩

lemma *gappend-glength*[*simp*]: $glength\ (a\ @_g\ b) = glength\ a + glength\ b$
⟨*proof*⟩

lemma *gmap-glength*[*simp*]: $glength\ (gmap\ f\ a) = glength\ a$
⟨*proof*⟩

lemma *glength-0-conv*[*simp*]: $(glength\ a = 0) = (a = FL\ [])$
⟨*proof*⟩

lemma *glength-greater-0-conv*[*simp*]: $(0 < glength\ a) = (a \neq FL\ [])$
⟨*proof*⟩

lemma *glength-gSuc-conv*:
$(glength\ a = eSuc\ n) =$
$(\exists\, x\ b.\ a = x\ \#_g\ b \wedge glength\ b = n)$
⟨*proof*⟩

lemma *gSuc-glength-conv*:
$(eSuc\ n = glength\ a) =$
$(\exists\, x\ b.\ a = x\ \#_g\ b \wedge glength\ b = n)$
⟨*proof*⟩

11.2.3 $@_g$ – gappend

lemma *gappend-Nil*[*simp*]: $(FL\ [])\ @_g\ a = a$
⟨*proof*⟩

lemma *gappend-Nil2*[*simp*]: $a\ @_g\ (FL\ [])= a$
⟨*proof*⟩

lemma *gappend-is-Nil-conv*[*simp*]: $(a\ @_g\ b = FL\ []) = (a = FL\ [] \wedge b = FL\ [])$
⟨*proof*⟩

lemma *Nil-is-gappend-conv*[*simp*]: $(FL\ [] = a\ @_g\ b) = (a = FL\ [] \wedge b = FL\ [])$
⟨*proof*⟩

lemma *gappend-assoc*[*simp*]: $(a\ @_g\ b)\ @_g\ c = a\ @_g\ b\ @_g\ c$
⟨*proof*⟩

lemma *gappend-infin*[*simp*]: $IL\ f\ @_g\ b = IL\ f$
⟨*proof*⟩

lemma *same-gappend-eq-disj*[*simp*]: $(a\ @_g\ b = a\ @_g\ c) = (glength\ a = \infty \vee b = c)$
⟨*proof*⟩
lemma *same-gappend-eq*:
$glength\ a < \infty \Longrightarrow (a\ @_g\ b = a\ @_g\ c) = (b = c)$
⟨*proof*⟩

11.2.4 *gmap*

lemma *gmap-gappend*[*simp*]: $gmap\ f\ (a\ @_g\ b) = gmap\ f\ a\ @_g\ gmap\ f\ b$
⟨*proof*⟩

lemmas *gmap-gmap*[*simp*] = *glist.map-comp*

lemma *gmap-eq-conv*[*simp*]: $(gmap\ f\ a = gmap\ g\ a) = (\forall\, x{\in}gset\ a.\ f\ x = g\ x)$
$\langle proof \rangle$

lemmas *gmap-cong* = *glist.map-cong*

lemma *gmap-is-Nil-conv*: $(gmap\ f\ a = FL\ []) = (a = FL\ [])$
$\langle proof \rangle$

lemma *gmap-eq-imp-glength-eq*:
$gmap\ f\ a = gmap\ f\ b \Longrightarrow glength\ a = glength\ b$
$\langle proof \rangle$

11.2.5 *gset*

lemma *gset-gappend*[*simp*]:
$gset\ (a\ @_g\ b) =$
$(case\ a\ of\ FL\ a' \Rightarrow set\ a' \cup gset\ b \mid IL\ a' \ \Rightarrow range\ a')$
$\langle proof \rangle$

lemma *gset-gappend-if*:
$gset\ (a\ @_g\ b) =$
$(if\ glength\ a < \infty\ then\ gset\ a \cup gset\ b\ else\ gset\ a)$
$\langle proof \rangle$

lemma *gset-empty*[*simp*]: $(gset\ a = \{\}) = (a = FL\ [])$
$\langle proof \rangle$

lemmas *gset-gmap*[*simp*] = *glist.set-map*

lemma *icard-glength*: $icard\ (gset\ a) \le glength\ a$
$\langle proof \rangle$

11.2.6 $!_g$ − gnth

lemma *gnth-gCons-0*[*simp*]: $(x\ \#_g\ a)\ !_g\ 0 = x$
$\langle proof \rangle$

lemma *gnth-gCons-Suc*[*simp*]: $(x\ \#_g\ a)\ !_g\ Suc\ n = a\ !_g\ n$
$\langle proof \rangle$

lemma *gnth-gappend*:
$(a\ @_g\ b)\ !_g\ n =$
$(if\ enat\ n < glength\ a\ then\ a\ !_g\ n$
$else\ b\ !_g\ (n - the\text{-}enat\ (glength\ a)))$
$\langle proof \rangle$

lemma *gnth-gappend-length-plus*[*simp*]: $(FL\ xs\ @_g\ b)\ !_g\ (length\ xs + n) = b\ !_g\ n$
$\langle proof \rangle$

lemma *gmap-gnth*[*simp*]: $enat\ n < glength\ a \Longrightarrow gmap\ f\ a\ !_g\ n = f\ (a\ !_g\ n)$
⟨*proof*⟩

lemma *in-gset-cong-gnth*: $(x \in gset\ a) = (\exists\, i.\ enat\ i < glength\ a \wedge a\ !_g\ i = x)$
⟨*proof*⟩

11.2.7 *gtake* and *gdrop*

lemma *gtake-0*[*simp*]: $a \downarrow_g 0 = FL\ []$
⟨*proof*⟩

lemma *gdrop-0*[*simp*]: $a \uparrow_g 0 = a$
⟨*proof*⟩

lemma *gtake-Infty*[*simp*]: $a \downarrow_g \infty = a$
⟨*proof*⟩

lemma *gdrop-Infty*[*simp*]: $a \uparrow_g \infty = FL\ []$
⟨*proof*⟩

lemma *gtake-all*[*simp*]: $glength\ a \le n \Longrightarrow a \downarrow_g n = a$
⟨*proof*⟩

lemma *gdrop-all*[*simp*]: $glength\ a \le n \Longrightarrow a \uparrow_g n = FL\ []$
⟨*proof*⟩

lemma *gtake-eSuc-gCons*[*simp*]: $(x\ \#_g\ a) \downarrow_g (eSuc\ n) = x\ \#_g\ a \downarrow_g n$
⟨*proof*⟩

lemma *gdrop-eSuc-gCons*[*simp*]: $(x\ \#_g\ a) \uparrow_g (eSuc\ n) = a \uparrow_g n$
⟨*proof*⟩

lemma *gtake-eSuc*: $a \ne FL\ [] \Longrightarrow a \downarrow_g (eSuc\ n) = a\ !_g\ 0\ \#_g\ (a \uparrow_g (eSuc\ 0) \downarrow_g n)$
⟨*proof*⟩

lemma *gdrop-eSuc*: $a \uparrow_g (eSuc\ n) = a \uparrow_g (eSuc\ 0) \uparrow_g n$
⟨*proof*⟩

lemma *gnth-via-grop*: $a \uparrow_g (enat\ n) = x\ \#_g\ b \Longrightarrow a\ !_g\ n = x$
⟨*proof*⟩

lemma *gtake-eSuc-conv-gapp-gnth*:
$enat\ n < glength\ a \Longrightarrow a \downarrow_g enat\ (Suc\ n) = a \downarrow_g (enat\ n)\ @_g\ FL\ [a\ !_g\ n]$
⟨*proof*⟩

lemma *gdrop-eSuc-conv-tl*:
$enat\ n < glength\ a \Longrightarrow a\ !_g\ n\ \#_g\ a \uparrow_g enat\ (Suc\ n) = a \uparrow_g enat\ n$
⟨*proof*⟩

lemma *glength-gtake*[*simp*]: $glength\ (a \downarrow_g\ n) = min\ (glength\ a)\ n$
⟨*proof*⟩

lemma *glength-drop*[*simp*]: $glength\ (a \uparrow_g\ (enat\ n)) = glength\ a - (enat\ n)$
⟨*proof*⟩

end

12 Prefices on finite and infinite lists

theory *ListInf-Prefix*
imports *HOL−Library.Sublist ListInf*
begin

12.1 Additional list prefix results

lemma *prefix-eq-prefix-take-ex*: $prefix\ xs\ ys = (\exists\, n.\ ys \downarrow n = xs)$
⟨*proof*⟩

lemma *prefix-take-eq-prefix-take-ex*: $(ys \downarrow (length\ xs) = xs) = (\exists\, n.\ ys \downarrow n = xs)$
⟨*proof*⟩

lemma *prefix-eq-prefix-take*: $prefix\ xs\ ys = (ys \downarrow (length\ xs) = xs)$
⟨*proof*⟩

lemma *strict-prefix-take-eq-strict-prefix-take-ex*:
$(ys \downarrow (length\ xs) = xs \wedge xs \neq ys) =$
$((\exists\, n.\ ys \downarrow n = xs) \wedge xs \neq ys)$
⟨*proof*⟩

lemma *strict-prefix-eq-strict-prefix-take-ex*: $strict\text{-}prefix\ xs\ ys = ((\exists\, n.\ ys \downarrow n = xs) \wedge xs \neq ys)$
⟨*proof*⟩
lemma *strict-prefix-eq-strict-prefix-take*: $strict\text{-}prefix\ xs\ ys = (ys \downarrow (length\ xs) = xs \wedge xs \neq ys)$
⟨*proof*⟩

lemma *take-imp-prefix*: $prefix\ (xs \downarrow n)\ xs$
⟨*proof*⟩

lemma *eq-imp-prefix*: $xs = (ys::'a\ list) \Longrightarrow prefix\ xs\ ys$
⟨*proof*⟩

lemma *le-take-imp-prefix*: $a \leq b \Longrightarrow prefix\ (xs \downarrow a)\ (xs \downarrow b)$
⟨*proof*⟩

lemma *take-prefix-imp-le*:
$\llbracket$ $a \le$ *length xs*; *prefix* $(xs \downarrow a)$ $(xs \downarrow b)$ $\rrbracket \Longrightarrow a \le b$
$\langle proof \rangle$

lemma *take-prefixeq-le-conv*:
$a \le$ *length xs* $\Longrightarrow$ *prefix* $(xs \downarrow a)$ $(xs \downarrow b) = (a \le b)$
$\langle proof \rangle$
lemma *append-imp-prefix*[*simp*, *intro*]: *prefix a* $(a$ @ $b)$
$\langle proof \rangle$

lemma *prefix-imp-take-eq*:
$\llbracket$ $n \le$ *length xs*; *prefix xs ys* $\rrbracket \Longrightarrow xs \downarrow n = ys \downarrow n$
$\langle proof \rangle$

lemma *prefix-length-le-eq-conv*: (*prefix xs ys* $\wedge$ *length ys* $\le$ *length xs*) $= (xs = ys)$
$\langle proof \rangle$

lemma *take-length-prefix-conv*:
length xs $\le$ *length ys* $\Longrightarrow$ *prefix* $(ys \downarrow$ *length xs*$)$ *xs* $=$ *prefix xs ys*
$\langle proof \rangle$

lemma *append-eq-imp-take*:
$\llbracket$ $k \le$ *length xs*; *length r1* $= k$; *r1* @ *r2* $= xs$ $\rrbracket \Longrightarrow r1 = xs \downarrow k$
$\langle proof \rangle$

lemma *take-the-conv*:
$xs \downarrow k =$ (*if length xs* $\le k$ *then xs else* (*THE r. length r* $= k \wedge (\exists\, r2.\ r$ @ $r2 =$ $xs)))$
$\langle proof \rangle$

lemma *prefix-refl*: *prefix xs* (*xs*::$'a$ *list*)
$\langle proof \rangle$

lemma *prefix-trans*: $\llbracket$ *prefix xs ys*; *prefix* (*ys*::$'a$ *list*) *zs* $\rrbracket \Longrightarrow$ *prefix xs zs*
$\langle proof \rangle$

lemma *prefixeq-antisym*: $\llbracket$ *prefix xs ys*; *prefix* (*ys*::$'a$ *list*) *xs* $\rrbracket \Longrightarrow xs = ys$
$\langle proof \rangle$

12.2 Counting equal pairs

Counting number of equal elements in two lists

definition *mirror-pair* :: $('a \times 'b) \Rightarrow ('b \times 'a)$
where *mirror-pair p* $\equiv$ (*snd p*, *fst p*)

lemma *zip-mirror*[*rule-format*]:
$\llbracket$ $i <$ *min* (*length xs*) (*length ys*);
p1 $=$ (*zip xs ys*) ! i; *p2* $=$ (*zip ys xs*) ! i $\rrbracket \Longrightarrow$

$mirror\text{-}pair\ p1 = p2$
⟨*proof*⟩

definition *equal-pair* :: $('a \times 'a) \Rightarrow bool$
where $equal\text{-}pair\ p \equiv (fst\ p = snd\ p)$

lemma *mirror-pair-equal*: $equal\text{-}pair\ (mirror\text{-}pair\ p) = (equal\text{-}pair\ p)$
⟨*proof*⟩

primrec
equal-pair-count :: $('a \times 'a)\ list \Rightarrow nat$
where
$equal\text{-}pair\text{-}count\ [] = 0$
$|\ equal\text{-}pair\text{-}count\ (p\ \#\ ps) = ($
$\quad if\ (fst\ p = snd\ p)$
$\quad\quad then\ Suc\ (equal\text{-}pair\text{-}count\ ps)$
$\quad\quad else\ 0)$

lemma *equal-pair-count-le*: $equal\text{-}pair\text{-}count\ xs \le length\ xs$
⟨*proof*⟩

lemma *equal-pair-count-0*:
$fst\ (hd\ ps) \ne snd\ (hd\ ps) \Longrightarrow equal\text{-}pair\text{-}count\ ps = 0$
⟨*proof*⟩

lemma *equal-pair-count-Suc*:
$equal\text{-}pair\text{-}count\ ((a,\ a)\ \#\ ps) = Suc\ (equal\text{-}pair\text{-}count\ ps)$
⟨*proof*⟩

lemma *equal-pair-count-eq-pairwise*[*rule-format*]:
$[\![\ length\ ps1 = length\ ps2;$
$\quad \forall i{<}length\ ps2.\ equal\text{-}pair\ (ps1\ !\ i) = equal\text{-}pair(ps2\ !\ i)\ ]\!]$
$\Longrightarrow equal\text{-}pair\text{-}count\ ps1 = equal\text{-}pair\text{-}count\ ps2$
⟨*proof*⟩

lemma *equal-pair-count-mirror-pairwise*[*rule-format*]:
$[\![\ length\ ps1 = length\ ps2;$
$\quad \forall i{<}length\ ps2.\ ps1\ !\ i = mirror\text{-}pair\ (ps2\ !\ i)\ ]\!]$
$\Longrightarrow equal\text{-}pair\text{-}count\ ps1 = equal\text{-}pair\text{-}count\ ps2$
⟨*proof*⟩

lemma *equal-pair-count-correct*:
$\bigwedge i.\ i < equal\text{-}pair\text{-}count\ ps \Longrightarrow equal\text{-}pair\ (ps\ !\ i)$
⟨*proof*⟩

lemma *equal-pair-count-maximality-aux*[*rule-format*]: $\bigwedge i.$

$i = equal\text{-}pair\text{-}count\ ps \Longrightarrow length\ ps = i \vee \neg\ equal\text{-}pair\ (ps\ !\ i)$
⟨*proof*⟩

corollary *equal-pair-count-maximality1a*[*rule-format*]:
$equal\text{-}pair\text{-}count\ ps = length\ ps \vee \neg\ equal\text{-}pair\ (ps!equal\text{-}pair\text{-}count\ ps)$
⟨*proof*⟩

corollary *equal-pair-count-maximality1b*[*rule-format*]:
$equal\text{-}pair\text{-}count\ ps \neq length\ ps \Longrightarrow$
$\neg\ equal\text{-}pair\ (ps!equal\text{-}pair\text{-}count\ ps)$
⟨*proof*⟩

lemma *equal-pair-count-maximality2a*[*rule-format*]:
$equal\text{-}pair\text{-}count\ ps = length\ ps \vee$ — either all pairs are equal
$(\forall\, i{\geq}equal\text{-}pair\text{-}count\ ps.(\exists\, j{\leq}i.\ \neg equal\text{-}pair\ (ps\ !\ j)))$
⟨*proof*⟩

corollary *equal-pair-count-maximality2b*[*rule-format*]:
$equal\text{-}pair\text{-}count\ ps \neq length\ ps \Longrightarrow$
$\forall\, i{\geq}equal\text{-}pair\text{-}count\ ps.(\exists\, j{\leq}i.\ \neg equal\text{-}pair\ (ps!j))$
⟨*proof*⟩

lemmas *equal-pair-count-maximality* =
equal-pair-count-maximality1a equal-pair-count-maximality1b
equal-pair-count-maximality2a equal-pair-count-maximality2b

12.3 Prefix length

Length of the prefix infimum

definition $inf\text{-}prefix\text{-}length :: 'a\ list \Rightarrow 'a\ list \Rightarrow nat$
where $inf\text{-}prefix\text{-}length\ xs\ ys \equiv equal\text{-}pair\text{-}count\ (zip\ xs\ ys)$

value $int\ (inf\text{-}prefix\text{-}length\ [1::int,2,3,4,7,8,15]\ [1::int,2,3,4,7,15])$
value $int\ (inf\text{-}prefix\text{-}length\ [1::int,2,3,4]\ [1::int,2,3,4,7,15])$
value $int\ (inf\text{-}prefix\text{-}length\ []\ [1::int,2,3,4,7,15])$
value $int\ (inf\text{-}prefix\text{-}length\ [1::int,2,3,4,5]\ [1::int,2,3,4,5])$

lemma *inf-prefix-length-commute*[*rule-format*]:
$inf\text{-}prefix\text{-}length\ xs\ ys = inf\text{-}prefix\text{-}length\ ys\ xs$
⟨*proof*⟩

lemma *inf-prefix-length-leL*[*intro*]:
$inf\text{-}prefix\text{-}length\ xs\ ys \leq length\ xs$
⟨*proof*⟩

corollary *inf-prefix-length-leR*[*intro*]:
$inf\text{-}prefix\text{-}length\ xs\ ys \leq length\ ys$
⟨*proof*⟩

lemmas *inf-prefix-length-le* =
inf-prefix-length-leL
inf-prefix-length-leR

lemma *inf-prefix-length-le-min*[*rule-format*]:
inf-prefix-length xs ys $\leq$ *min* (*length xs*) (*length ys*)
⟨*proof*⟩

lemma *hd-inf-prefix-length-0*:
hd xs $\neq$ *hd ys* $\Longrightarrow$ *inf-prefix-length xs ys* = *0*
⟨*proof*⟩

lemma *inf-prefix-length-NilL*[*simp*]: *inf-prefix-length* [] *ys* = *0*
⟨*proof*⟩
lemma *inf-prefix-length-NilR*[*simp*]: *inf-prefix-length xs* [] = *0*
⟨*proof*⟩

lemma *inf-prefix-length-Suc*[*simp*]:
inf-prefix-length (*a* # *xs*) (*a* # *ys*) = *Suc* (*inf-prefix-length xs ys*)
⟨*proof*⟩

lemma *inf-prefix-length-correct*:
i < *inf-prefix-length xs ys* $\Longrightarrow$ *xs* ! *i* = *ys* ! *i*
⟨*proof*⟩

corollary *nth-neq-imp-inf-prefix-length-le*:
xs ! *i* $\neq$ *ys* ! *i* $\Longrightarrow$ *inf-prefix-length xs ys* $\leq$ *i*
⟨*proof*⟩

lemma *inf-prefix-length-maximality1*[*rule-format*]:
inf-prefix-length xs ys $\neq$ *min* (*length xs*) (*length ys*) $\Longrightarrow$
xs ! (*inf-prefix-length xs ys*) $\neq$ *ys* ! (*inf-prefix-length xs ys*)
⟨*proof*⟩

corollary *inf-prefix-length-maximality2*[*rule-format*]:
⟦ *inf-prefix-length xs ys* $\neq$ *min* (*length xs*) (*length ys*);
inf-prefix-length xs ys $\leq$ *i* ⟧ $\Longrightarrow$
$\exists j \leq i.$ *xs* ! *j* $\neq$ *ys* ! *j*
⟨*proof*⟩

lemmas *inf-prefix-length-maximality* =
inf-prefix-length-maximality1 inf-prefix-length-maximality2

lemma *inf-prefix-length-append*[*simp*]:
inf-prefix-length (*zs* @ *xs*) (*zs* @ *ys*) =
length zs + *inf-prefix-length xs ys*
⟨*proof*⟩

lemma *inf-prefix-length-take-correct*:

$n \le$ *inf-prefix-length xs ys* $\Longrightarrow xs \downarrow n = ys \downarrow n$
⟨*proof*⟩

lemma *inf-prefix-length-0-imp-hd-neq*:
$[\![$ $xs \neq []$; $ys \neq []$; *inf-prefix-length xs ys* $= 0$ $]\!] \Longrightarrow hd\ xs \neq hd\ ys$
⟨*proof*⟩

12.4 Prefix infimum

definition *inf-prefix* :: $'a\ list \Rightarrow 'a\ list \Rightarrow 'a\ list$ (**infixl** $\sqcap$ *70*)
where $xs \sqcap ys \equiv xs \downarrow$ (*inf-prefix-length xs ys*)

lemma *length-inf-prefix*: *length* $(xs \sqcap ys) =$ *inf-prefix-length xs ys*
⟨*proof*⟩

lemma *inf-prefix-commute*: $xs \sqcap ys = ys \sqcap xs$
⟨*proof*⟩

lemma *inf-prefix-takeL*: $xs \sqcap ys = xs \downarrow$ (*inf-prefix-length xs ys*)
⟨*proof*⟩

lemma *inf-prefix-takeR*: $xs \sqcap ys = ys \downarrow$ (*inf-prefix-length xs ys*)
⟨*proof*⟩

lemma *inf-prefix-correct*: $i <$ *length* $(xs \sqcap ys) \Longrightarrow xs\ !\ i = ys\ !\ i$
⟨*proof*⟩

corollary *inf-prefix-correctL*:
$i <$ *length* $(xs \sqcap ys) \Longrightarrow (xs \sqcap ys)\ !\ i = xs\ !\ i$
⟨*proof*⟩

corollary *inf-prefix-correctR*:
$i <$ *length* $(xs \sqcap ys) \Longrightarrow (xs \sqcap ys)\ !\ i = ys\ !\ i$
⟨*proof*⟩

lemma *inf-prefix-take-correct*:
$n \le$ *length* $(xs \sqcap ys) \Longrightarrow xs \downarrow n = ys \downarrow n$
⟨*proof*⟩

lemma *is-inf-prefix*[*rule-format*]:
$[\![$ *length zs* $=$ *length* $(xs \sqcap ys)$;
$\bigwedge i.\ i <$ *length* $(xs \sqcap ys) \Longrightarrow zs\ !\ i = xs\ !\ i \wedge zs\ !\ i = ys\ !\ i$ $]\!] \Longrightarrow$
$zs = xs \sqcap ys$
⟨*proof*⟩

lemma *hd-inf-prefix-Nil*: $hd\ xs \neq hd\ ys \Longrightarrow xs \sqcap ys = []$
⟨*proof*⟩

lemma *inf-prefix-Nil-imp-hd-neq*:

$\llbracket\ xs \neq []; ys \neq []; xs \sqcap ys = []\ \rrbracket \Longrightarrow hd\ xs \neq hd\ ys$
⟨*proof*⟩

lemma *length-inf-prefix-append*[*simp*]:
$length\ ((zs\ @\ xs) \sqcap (zs\ @\ ys)) =$
$length\ zs + length\ (xs \sqcap ys)$
⟨*proof*⟩

lemma *inf-prefix-append*[*simp*]: $(zs\ @\ xs) \sqcap (zs\ @\ ys) = zs\ @\ (xs \sqcap ys)$
⟨*proof*⟩

lemma *hd-neq-inf-prefix-append*:
$hd\ xs \neq hd\ ys \Longrightarrow (zs\ @\ xs) \sqcap (zs\ @\ ys) = zs$
⟨*proof*⟩

lemma *inf-prefix-NilL*[*simp*]: $[] \sqcap ys = []$
⟨*proof*⟩

corollary *inf-prefix-NilR*[*simp*]: $xs \sqcap [] = []$
⟨*proof*⟩

lemmas *inf-prefix-Nil* = *inf-prefix-NilL inf-prefix-NilR*

lemma *inf-prefix-Cons*[*simp*]: $(a \# xs) \sqcap (a \# ys) = a \# xs \sqcap ys$
⟨*proof*⟩

corollary *inf-prefix-hd*[*simp*]: $hd\ ((a \# xs) \sqcap (a \# ys)) = a$
⟨*proof*⟩

lemma *inf-prefix-le1*: $prefix\ (xs \sqcap ys)\ xs$
⟨*proof*⟩

lemma *inf-prefix-le2*: $prefix\ (xs \sqcap ys)\ ys$
⟨*proof*⟩

lemma *le-inf-prefix-iff*: $prefix\ x\ (y \sqcap z) = (prefix\ x\ y \wedge prefix\ x\ z)$
⟨*proof*⟩

lemma *le-imp-le-inf-prefix*: $\llbracket\ prefix\ x\ y; prefix\ x\ z\ \rrbracket \Longrightarrow prefix\ x\ (y \sqcap z)$
⟨*proof*⟩

interpretation *prefix*:
semilattice-inf
$(\sqcap) :: 'a\ list \Rightarrow 'a\ list \Rightarrow 'a\ list$
prefix
strict-prefix
⟨*proof*⟩

12.5 Prefices for infinite lists

definition *iprefix* :: $'a\ list \Rightarrow 'a\ ilist \Rightarrow bool$ (**infixl** $\sqsubseteq$ *50*)
where $xs \sqsubseteq f \equiv \exists\, g.\ f = xs \frown g$

lemma *iprefix-eq-iprefix-take*: $(xs \sqsubseteq f) = (f \Downarrow length\ xs = xs)$
$\langle proof \rangle$

lemma *iprefix-take-eq-iprefix-take-ex*:
$(f \Downarrow length\ xs = xs) = (\exists\, n.\ f \Downarrow n = xs)$
$\langle proof \rangle$

lemma *iprefix-eq-iprefix-take-ex*: $(xs \sqsubseteq f) = (\exists\, n.\ f \Downarrow n = xs)$
$\langle proof \rangle$

lemma *i-take-imp-iprefix*[*intro*]: $f \Downarrow n \sqsubseteq f$
$\langle proof \rangle$

lemma *i-take-prefix-le-conv*: $prefix\ (f \Downarrow a)\ (f \Downarrow b) = (a \le b)$
$\langle proof \rangle$

lemma *i-append-imp-iprefix*[*simp*,*intro*]: $xs \sqsubseteq xs \frown f$
$\langle proof \rangle$

lemma *iprefix-imp-take-eq*:
$[\![\ n \le length\ xs;\ xs \sqsubseteq f\ ]\!] \Longrightarrow xs \downarrow n = f \Downarrow n$
$\langle proof \rangle$

lemma *prefixeq-iprefix-trans*: $[\![\ prefix\ xs\ ys;\ ys \sqsubseteq f\ ]\!] \Longrightarrow xs \sqsubseteq f$
$\langle proof \rangle$

lemma *i-take-length-prefix-conv*: $prefix\ (f \Downarrow length\ xs)\ xs = (xs \sqsubseteq f)$
$\langle proof \rangle$

lemma *iprefixI*[*intro?*]: $f = xs \frown g \Longrightarrow xs \sqsubseteq f$
$\langle proof \rangle$

lemma *iprefixE*[*elim?*]: $[\![\ xs \sqsubseteq f;\ \bigwedge g.\ f = xs \frown g \Longrightarrow C\ ]\!] \Longrightarrow C$
$\langle proof \rangle$

lemma *Nil-iprefix*[*iff*]: $[] \sqsubseteq f$
$\langle proof \rangle$

lemma *same-prefix-iprefix*[*simp*]: $(xs\ @\ ys \sqsubseteq xs \frown f) = (ys \sqsubseteq f)$
$\langle proof \rangle$

lemma *prefix-iprefix*[*simp*]: $prefix\ xs\ ys \Longrightarrow xs \sqsubseteq ys \frown f$
$\langle proof \rangle$

lemma *append-iprefixD*: $xs\ @\ ys \sqsubseteq f \Longrightarrow xs \sqsubseteq f$

$\langle proof \rangle$

lemma *iprefix-length-le-imp-prefix*:
$[\![\ xs \sqsubseteq ys \frown f;\ length\ xs \le length\ ys\ ]\!] \Longrightarrow prefix\ xs\ ys$
$\langle proof \rangle$

lemma *iprefix-i-append*:
$(xs \sqsubseteq ys \frown f) = (prefix\ xs\ ys \vee (\exists zs.\ xs = ys\ @\ zs \wedge zs \sqsubseteq f))$
$\langle proof \rangle$

lemma *i-append-one-iprefix*:
$xs \sqsubseteq f \Longrightarrow xs\ @\ [f\ (length\ xs)] \sqsubseteq f$
$\langle proof \rangle$

lemma *iprefix-same-length-le*:
$[\![\ xs \sqsubseteq f;\ ys \sqsubseteq f;\ length\ xs \le length\ ys\ ]\!] \Longrightarrow prefix\ xs\ ys$
$\langle proof \rangle$

lemma *iprefix-same-cases*:
$[\![\ xs \sqsubseteq f;\ ys \sqsubseteq f\ ]\!] \Longrightarrow prefix\ xs\ ys \vee prefix\ ys\ xs$
$\langle proof \rangle$

lemma *set-mono-iprefix*: $xs \sqsubseteq f \Longrightarrow set\ xs \subseteq range\ f$
$\langle proof \rangle$

end
theory *ListInfinite*
imports
CommonSet/SetIntervalStep
ListInf/ListInf-Prefix
begin
end