Interval Temporal Logic on Natural Numbers

David Trachtenherz

March 17, 2025

Abstract

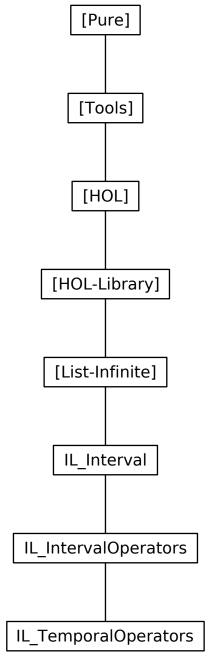
We introduce a theory of temporal logic operators using sets of natural numbers as time domain, formalized in a shallow embedding manner. The theory comprises special natural intervals (theory IL_Interval: open and closed intervals, continuous and modulo intervals, interval traversing results), operators for shifting intervals to left/right on the number axis as well as expanding/contracting intervals by constant factors (theory IL_IntervalOperators.thy), and ultimately definitions and results for unary and binary temporal operators on arbitrary natural sets (theory IL_TemporalOperators).

Contents

1 Intervals and operations for temporal logic declarations 2
- 1.1 Time intervals – definitions and basic lemmata 2
  - 1.1.1 Definitions 2
  - 1.1.2 Membership in an interval 3
  - 1.1.3 Interval conversions 5
  - 1.1.4 Finiteness and emptiness of intervals 6
  - 1.1.5 *Min* and *Max* element of an interval 7
- 1.2 Adding and subtracting constants to interval elements 8
- 1.3 Relations between intervals 11
  - 1.3.1 Auxiliary lemmata 11
  - 1.3.2 Subset relation between intervals 12
  - 1.3.3 Equality of intervals 18
  - 1.3.4 Inequality of intervals 18
- 1.4 Union and intersection of intervals 19
- 1.5 Cutting intervals 23
- 1.6 Cardinality of intervals 30
- 1.7 Functions *inext* and *iprev* with intervals 32
  - 1.7.1 Mirroring of intervals 35
  - 1.7.2 Functions *inext-nth* and *iprev-nth* on intervals 36
- 1.8 Induction with intervals 38

2 Arithmetic operators on natural intervals **39**
2.1 Arithmetic operations with intervals 39
2.1.1 Addition of and multiplication by constants 39
2.1.2 Some conversions between intervals using constant addition and multiplication 43
2.1.3 Subtraction of constants 44
2.1.4 Subtraction of intervals from constants 49
2.1.5 Division of intervals by constants 52
2.2 Interval cut operators with arithmetic interval operators . . . 59
2.3 *inext* and *iprev* with interval operators 61
2.4 Cardinality of intervals with interval operators 64
2.5 Results about sets of intervals 67
2.5.1 Set of intervals without and with empty interval . . . 67
2.5.2 Interval sets are closed under cutting 72
2.5.3 Interval sets are closed under addition and multiplication 73
2.5.4 Interval sets are closed with certain conditions under subtraction 73
2.5.5 Interval sets are not closed under division 74
2.5.6 Sets of intervals closed under division 74

3 Temporal logic operators on natural intervals **78**
3.1 Basic definitions 79
3.2 Basic lemmata for temporal operators 82
3.2.1 Intro/elim rules 82
3.2.2 Rewrite rules for trivial simplification 83
3.2.3 Empty sets and singletons 89
3.2.4 Conversions between temporal operators 90
3.2.5 Some implication results 93
3.2.6 Congruence rules for temporal operators' predicates . 94
3.2.7 Temporal operators with set unions/intersections and subsets 95
3.3 Further results for temporal operators 96
3.4 Temporal operators and arithmetic interval operators 98


[Pure]
[Tools]
[HOL]
[HOL-Library]
[List-Infinite]
IL_Interval
IL_IntervalOperators
IL_TemporalOperators

1 Intervals and operations for temporal logic declarations

theory *IL-Interval*
imports
List−Infinite.InfiniteSet2
List−Infinite.SetIntervalStep
begin

1.1 Time intervals – definitions and basic lemmata

1.1.1 Definitions

type-synonym *Time* = *nat*

type-synonym *iT* = *Time set*

Infinite interval starting at some natural *n*.

definition
iFROM :: *Time* ⇒ *iT* (‹[-. . .]›)
where
[*n*. . .] ≡ {*n*..}

Finite interval starting at *0* and ending at some natural *n*.

definition
iTILL :: *Time* ⇒ *iT* (‹[. . .-]›)
where
[. . .*n*] ≡ {..*n*}

Finite bounded interval containing the naturals between *n* and *n* + *d*. *d* denotes the difference between left and right interval bound. The number of elements is *d* + *1* so that an empty interval cannot be defined.

definition
iIN :: *Time* ⇒ *nat* ⇒ *iT* (‹[-. . .,-]›)
where
[*n*. . .,*d*] ≡ {*n*..*n*+*d*}

Infinite modulo interval containing all naturals having the same division remainder modulo *m* as *r*, and beginning at *n*.

definition
iMOD :: *Time* ⇒ *nat* ⇒ *iT* (‹[-, *mod* -]›)
where
[*r*, *mod m*] ≡ { *x*. *x mod m* = *r mod m* ∧ *r* ≤ *x*}

Finite bounded modulo interval containing all naturals having the same division remainder modulo *m* as *r*, beginning at *n*, and ending after *c* cycles at $r + m * c$. The number of elements is $c + 1$ so that an empty interval cannot be defined.

definition
  iMODb :: *Time* ⇒ *nat* ⇒ *nat* ⇒ *iT* (‹[-, *mod* -, -]›)
where
  $[r,\ mod\ m,\ c] \equiv \{\ x.\ x\ mod\ m = r\ mod\ m \wedge r \leq x \wedge x \leq r + m * c\}$

1.1.2 Membership in an interval

lemmas *iT-defs* = *iFROM-def iTILL-def iIN-def iMOD-def iMODb-def*

lemma *iFROM-iff*: $x \in [n\ldots] = (n \leq x)$
⟨*proof*⟩
lemma *iTILL-iff*: $x \in [\ldots n] = (x \leq n)$
⟨*proof*⟩
lemma *iIN-iff*:$x \in [n\ldots,d] = (n \leq x \wedge x \leq n + d)$
⟨*proof*⟩
lemma *iMOD-iff*: $x \in [r,\ mod\ m] = (x\ mod\ m = r\ mod\ m \wedge r \leq x)$
⟨*proof*⟩
lemma *iMODb-iff*: $x \in [r,\ mod\ m,\ c] =$
  $(x\ mod\ m = r\ mod\ m \wedge r \leq x \wedge x \leq r + m * c)$
⟨*proof*⟩

lemma *iFROM-D*: $x \in [n\ldots] \Longrightarrow (n \leq x)$
⟨*proof*⟩
lemma *iTILL-D*: $x \in [\ldots n] \Longrightarrow (x \leq n)$
⟨*proof*⟩
corollary *iIN-geD*: $x \in [n\ldots,d] \Longrightarrow n \leq x$
⟨*proof*⟩
corollary *iIN-leD*: $x \in [n\ldots,d] \Longrightarrow x \leq n + d$
⟨*proof*⟩
corollary *iMOD-modD*: $x \in [r,\ mod\ m] \Longrightarrow x\ mod\ m = r\ mod\ m$
⟨*proof*⟩
corollary *iMOD-geD*: $x \in [r,\ mod\ m] \Longrightarrow r \leq x$
⟨*proof*⟩
corollary *iMODb-modD*: $x \in [r,\ mod\ m,\ c] \Longrightarrow x\ mod\ m = r\ mod\ m$
⟨*proof*⟩
corollary *iMODb-geD*: $x \in [r,\ mod\ m,\ c] \Longrightarrow r \leq x$
⟨*proof*⟩
corollary *iMODb-leD*: $x \in [r,\ mod\ m,\ c] \Longrightarrow x \leq r + m * c$
⟨*proof*⟩

lemmas *iT-iff* = *iFROM-iff iTILL-iff iIN-iff iMOD-iff iMODb-iff*

lemmas *iT-drule* =
  iFROM-D
  iTILL-D
  iIN-geD iIN-leD
  iMOD-modD iMOD-geD

iMODb-modD iMODb-geD iMODb-leD

lemma
iFROM-I [*intro*]: $n \le x \Longrightarrow x \in [n\ldots]$ **and**
iTILL-I [*intro*]: $x \le n \Longrightarrow x \in [\ldots n]$ **and**
iIN-I [*intro*]: $n \le x \Longrightarrow x \le n + d \Longrightarrow x \in [n\ldots,d]$ **and**
iMOD-I [*intro*]: $x \ mod \ m = r \ mod \ m \Longrightarrow r \le x \Longrightarrow x \in [r, \ mod \ m]$ **and**
iMODb-I [*intro*]: $x \ mod \ m = r \ mod \ m \Longrightarrow r \le x \Longrightarrow x \le r + m * c \Longrightarrow x \in [r, \ mod \ m, \ c]$
⟨*proof*⟩

lemma
iFROM-E [*elim*]: $x \in [n\ldots] \Longrightarrow (n \le x \Longrightarrow P) \Longrightarrow P$ **and**
iTILL-E [*elim*]: $x \in [\ldots n] \Longrightarrow (x \le n \Longrightarrow P) \Longrightarrow P$ **and**
iIN-E [*elim*]: $x \in [n\ldots,d] \Longrightarrow (n \le x \Longrightarrow x \le n + d \Longrightarrow P) \Longrightarrow P$ **and**
iMOD-E [*elim*]: $x \in [r, \ mod \ m] \Longrightarrow (x \ mod \ m = r \ mod \ m \Longrightarrow r \le x \Longrightarrow P) \Longrightarrow P$ **and**
iMODb-E [*elim*]: $x \in [r, \ mod \ m, \ c] \Longrightarrow (x \ mod \ m = r \ mod \ m \Longrightarrow r \le x \Longrightarrow x \le r + m * c \Longrightarrow P) \Longrightarrow P$
⟨*proof*⟩

lemma *iIN-Suc-insert-conv*:
insert $(Suc \ (n + d)) \ [n\ldots,d] = [n\ldots,Suc \ d]$
⟨*proof*⟩

lemma *iTILL-Suc-insert-conv*: *insert* $(Suc \ n) \ [\ldots n] = [\ldots Suc \ n]$
⟨*proof*⟩

lemma *iMODb-Suc-insert-conv*:
insert $(r + m * Suc \ c) \ [r, \ mod \ m, \ c] = [r, \ mod \ m, \ Suc \ c]$
⟨*proof*⟩

lemma *iFROM-pred-insert-conv*: *insert* $(n - Suc \ 0) \ [n\ldots] = [n - Suc \ 0\ldots]$
⟨*proof*⟩

lemma *iIN-pred-insert-conv*:
$0 < n \Longrightarrow$ *insert* $(n - Suc \ 0) \ [n\ldots,d] = [n - Suc \ 0\ldots,Suc \ d]$
⟨*proof*⟩

lemma *iMOD-pred-insert-conv*:
$m \le r \Longrightarrow$ *insert* $(r - m) \ [r, \ mod \ m] = [r - m, \ mod \ m]$
⟨*proof*⟩

lemma *iMODb-pred-insert-conv*:
$m \le r \Longrightarrow$ *insert* $(r - m) \ [r, \ mod \ m, \ c] = [r - m, \ mod \ m, \ Suc \ c]$

⟨*proof*⟩

lemma *iFROM-Suc-pred-insert-conv*: *insert n* [*Suc n*...] = [*n*...]
⟨*proof*⟩
lemma *iIN-Suc-pred-insert-conv*: *insert n* [*Suc n*...,*d*] = [*n*...,*Suc d*]
⟨*proof*⟩
lemma *iMOD-Suc-pred-insert-conv*: *insert r* [*r* + *m*, *mod m*] = [*r*, *mod m*]
⟨*proof*⟩
lemma *iMODb-Suc-pred-insert-conv*: *insert r* [*r* + *m*, *mod m*, *c*] = [*r*, *mod m*, *Suc c*]
⟨*proof*⟩

lemmas *iT-Suc-insert* =
iIN-Suc-insert-conv
iTILL-Suc-insert-conv
iMODb-Suc-insert-conv
lemmas *iT-pred-insert* =
iFROM-pred-insert-conv
iIN-pred-insert-conv
iMOD-pred-insert-conv
iMODb-pred-insert-conv
lemmas *iT-Suc-pred-insert* =
iFROM-Suc-pred-insert-conv
iIN-Suc-pred-insert-conv
iMOD-Suc-pred-insert-conv
iMODb-Suc-pred-insert-conv

lemma *iMOD-mem-diff*: ⟦ *a* ∈ [*r*, *mod m*]; *b* ∈ [*r*, *mod m*] ⟧ ⟹ (*a* − *b*) *mod m* = *0*
⟨*proof*⟩
lemma *iMODb-mem-diff*: ⟦ *a* ∈ [*r*, *mod m*, *c*]; *b* ∈ [*r*, *mod m*, *c*] ⟧ ⟹ (*a* − *b*) *mod m* = *0*
⟨*proof*⟩

1.1.3 Interval conversions

lemma *iIN-0-iTILL-conv*:[*0*...,*n*] = [...*n*]
⟨*proof*⟩
lemma *iIN-iTILL-iTILL-conv*: *0* < *n* ⟹ [*n*...,*d*] = [...*n*+*d*] − [...*n* − *Suc 0*]
⟨*proof*⟩
lemma *iIN-iFROM-iTILL-conv*: [*n*...,*d*] = [*n*...] ∩ [...*n*+*d*]
⟨*proof*⟩
lemma *iMODb-iMOD-iTILL-conv*: [*r*, *mod m*, *c*] = [*r*, *mod m*] ∩ [...*r*+*m*∗*c*]
⟨*proof*⟩
lemma *iMODb-iMOD-iIN-conv*: [*r*, *mod m*, *c*] = [*r*, *mod m*] ∩ [*r*...,*m*∗*c*]
⟨*proof*⟩

lemma *iFROM-iTILL-iIN-conv*: *n* ≤ *n'* ⟹ [*n*...] ∩ [...*n'*] = [*n*...,*n'*−*n*]
⟨*proof*⟩

lemma *iMOD-iTILL-iMODb-conv*:
$r \le n \Longrightarrow [r,\ mod\ m] \cap [\ldots n] = [r,\ mod\ m,\ (n - r)\ div\ m]$
⟨*proof*⟩

lemma *iMOD-iIN-iMODb-conv*:
$[r,\ mod\ m] \cap [r\ldots,d] = [r,\ mod\ m,\ d\ div\ m]$
⟨*proof*⟩

lemma *iFROM-0*: $[0\ldots] = UNIV$
⟨*proof*⟩

lemma *iTILL-0*: $[\ldots 0] = \{0\}$
⟨*proof*⟩

lemma *iIN-0*: $[n\ldots,0] = \{n\}$
⟨*proof*⟩

lemma *iMOD-0*: $[r,\ mod\ 0] = [r\ldots,0]$
⟨*proof*⟩

lemma *iMODb-mod-0*: $[r,\ mod\ 0,\ c] = [r\ldots,0]$
⟨*proof*⟩

lemma *iMODb-0*: $[r,\ mod\ m,\ 0] = [r\ldots,0]$
⟨*proof*⟩

lemmas *iT-0* =
iFROM-0
iTILL-0
iIN-0
iMOD-0
iMODb-mod-0
iMODb-0

lemma *iMOD-1*: $[r,\ mod\ Suc\ 0] = [r\ldots]$
⟨*proof*⟩

lemma *iMODb-mod-1*: $[r,\ mod\ Suc\ 0,\ c] = [r\ldots,c]$
⟨*proof*⟩

1.1.4 Finiteness and emptiness of intervals

lemma
iFROM-not-empty: $[n\ldots] \neq \{\}$ **and**
iTILL-not-empty: $[\ldots n] \neq \{\}$ **and**
iIN-not-empty: $[n\ldots,d] \neq \{\}$ **and**
iMOD-not-empty: $[r,\ mod\ m] \neq \{\}$ **and**
iMODb-not-empty: $[r,\ mod\ m,\ c] \neq \{\}$
⟨*proof*⟩

lemmas *iT-not-empty* =
iFROM-not-empty
iTILL-not-empty
iIN-not-empty
iMOD-not-empty
iMODb-not-empty

lemma
iTILL-finite: *finite* [...*n*] **and**
iIN-finite: *finite* [*n*...,*d*] **and**
iMODb-finite: *finite* [*r*, *mod m*, *c*] **and**
iMOD-0-finite: *finite* [*r*, *mod 0*]
⟨*proof*⟩

lemma *iFROM-infinite*: *infinite* [*n*...]
⟨*proof*⟩

lemma *iMOD-infinite*: *0* < *m* ⟹ *infinite* [*r*, *mod m*]
⟨*proof*⟩

lemmas *iT-finite* =
iTILL-finite
iIN-finite
iMODb-finite iMOD-0-finite

lemmas *iT-infinite* =
iFROM-infinite
iMOD-infinite

1.1.5 *Min* and *Max* element of an interval

lemma
iTILL-Min: *iMin* [...*n*] = *0* **and**
iFROM-Min: *iMin* [*n*...] = *n* **and**
iIN-Min: *iMin* [*n*...,*d*] = *n* **and**
iMOD-Min: *iMin* [*r*, *mod m*] = *r* **and**
iMODb-Min: *iMin* [*r*, *mod m*, *c*] = *r*
⟨*proof*⟩

lemmas *iT-Min* =
iIN-Min
iTILL-Min
iFROM-Min
iMOD-Min
iMODb-Min

lemma
iTILL-Max: *Max* [...*n*] = *n* **and**

iIN-Max: *Max* $[n\ldots,d] = n+d$ **and**
iMODb-Max: *Max* $[r,\ mod\ m,\ c] = r + m * c$ **and**
iMOD-0-Max: *Max* $[r,\ mod\ 0] = r$
⟨*proof*⟩

lemmas *iT-Max* =
iTILL-Max
iIN-Max
iMODb-Max
iMOD-0-Max

lemma
iTILL-iMax: *iMax* $[\ldots n] =$ *enat* n **and**
iIN-iMax: *iMax* $[n\ldots,d] =$ *enat* $(n+d)$ **and**
iMODb-iMax: *iMax* $[r,\ mod\ m,\ c] =$ *enat* $(r + m * c)$ **and**
iMOD-0-iMax: *iMax* $[r,\ mod\ 0] =$ *enat* r **and**
iFROM-iMax: *iMax* $[n\ldots] = \infty$ **and**
iMOD-iMax: $0 < m \Longrightarrow$ *iMax* $[r,\ mod\ m] = \infty$
⟨*proof*⟩

lemmas *iT-iMax* =
iTILL-iMax
iIN-iMax
iMODb-iMax
iMOD-0-iMax
iFROM-iMax
iMOD-iMax

1.2 Adding and subtracting constants to interval elements

lemma
iFROM-plus: $x \in [n\ldots] \Longrightarrow x + k \in [n\ldots]$ **and**
iFROM-Suc: $x \in [n\ldots] \Longrightarrow Suc\ x \in [n\ldots]$ **and**
iFROM-minus: $[\![x \in [n\ldots];\ k \le x - n]\!] \Longrightarrow x - k \in [n\ldots]$ **and**
iFROM-pred: $n < x \Longrightarrow x - Suc\ 0 \in [n\ldots]$
⟨*proof*⟩

lemma
iTILL-plus: $[\![x \in [\ldots n];\ k \le n - x]\!] \Longrightarrow x + k \in [\ldots n]$ **and**
iTILL-Suc: $x < n \Longrightarrow Suc\ x \in [\ldots n]$ **and**
iTILL-minus: $x \in [\ldots n] \Longrightarrow x - k \in [\ldots n]$ **and**
iTILL-pred: $x \in [\ldots n] \Longrightarrow x - Suc\ 0 \in [\ldots n]$
⟨*proof*⟩

lemma *iIN-plus*: $[\![x \in [n\ldots,d];\ k \le n + d - x]\!] \Longrightarrow x + k \in [n\ldots,d]$
⟨*proof*⟩

lemma *iIN-Suc*: $[\![x \in [n\ldots,d];\ x < n + d]\!] \Longrightarrow Suc\ x \in [n\ldots,d]$
⟨*proof*⟩

lemma *iIN-minus*: $\llbracket x \in [n\ldots,d];\ k \le x - n \rrbracket \Longrightarrow x - k \in [n\ldots,d]$
⟨*proof*⟩

lemma *iIN-pred*: $\llbracket x \in [n\ldots,d];\ n < x \rrbracket \Longrightarrow x - Suc\ 0 \in [n\ldots,d]$
⟨*proof*⟩

lemma *iMOD-plus-divisor-mult*: $x \in [r,\ mod\ m] \Longrightarrow x + k * m \in [r,\ mod\ m]$
⟨*proof*⟩

corollary *iMOD-plus-divisor*: $x \in [r,\ mod\ m] \Longrightarrow x + m \in [r,\ mod\ m]$
⟨*proof*⟩

lemma *iMOD-minus-divisor-mult*:
$\llbracket x \in [r,\ mod\ m];\ k * m \le x - r \rrbracket \Longrightarrow x - k * m \in [r,\ mod\ m]$
⟨*proof*⟩

corollary *iMOD-minus-divisor-mult2*:
$\llbracket x \in [r,\ mod\ m];\ k \le (x - r)\ div\ m \rrbracket \Longrightarrow x - k * m \in [r,\ mod\ m]$
⟨*proof*⟩

corollary *iMOD-minus-divisor*:
$\llbracket x \in [r,\ mod\ m];\ m + r \le x \rrbracket \Longrightarrow x - m \in [r,\ mod\ m]$
⟨*proof*⟩

lemma *iMOD-plus*:
$x \in [r,\ mod\ m] \Longrightarrow (x + k \in [r,\ mod\ m]) = (k\ mod\ m = 0)$
⟨*proof*⟩
corollary *iMOD-Suc*:
$x \in [r,\ mod\ m] \Longrightarrow (Suc\ x \in [r,\ mod\ m]) = (m = Suc\ 0)$
⟨*proof*⟩

lemma *iMOD-minus*:
$\llbracket x \in [r,\ mod\ m];\ k \le x - r \rrbracket \Longrightarrow (x - k \in [r,\ mod\ m]) = (k\ mod\ m = 0)$
⟨*proof*⟩

corollary *iMOD-pred*:
$\llbracket x \in [r,\ mod\ m];\ r < x \rrbracket \Longrightarrow (x - Suc\ 0 \in [r,\ mod\ m]) = (m = Suc\ 0)$
⟨*proof*⟩

lemma *iMODb-plus-divisor-mult*:
$\llbracket x \in [r,\ mod\ m,\ c];\ k * m \le r + m * c - x \rrbracket \Longrightarrow x + k * m \in [r,\ mod\ m,\ c]$
⟨*proof*⟩

lemma *iMODb-plus-divisor-mult2*:
$\llbracket x \in [r,\ mod\ m,\ c];\ k \le c - (x - r)\ div\ m \rrbracket \Longrightarrow$
$x + k * m \in [r,\ mod\ m,\ c]$
⟨*proof*⟩
lemma *iMODb-plus-divisor*:

$[\![x \in [r, mod\ m, c]; x < r + m * c]\!] \Longrightarrow x + m \in [r, mod\ m, c]$
⟨*proof*⟩

lemma *iMODb-minus-divisor-mult*:
$[\![x \in [r, mod\ m, c]; r + k * m \leq x]\!] \Longrightarrow x - k * m \in [r, mod\ m, c]$
⟨*proof*⟩

lemma *iMODb-plus*:
$[\![x \in [r, mod\ m, c]; k \leq r + m * c - x]\!] \Longrightarrow$
$(x + k \in [r, mod\ m, c]) = (k\ mod\ m = 0)$
⟨*proof*⟩

corollary *iMODb-Suc*:
$[\![x \in [r, mod\ m, c]; x < r + m * c]\!] \Longrightarrow$
$(Suc\ x \in [r, mod\ m, c]) = (m = Suc\ 0)$
⟨*proof*⟩

lemma *iMODb-minus*:
$[\![x \in [r, mod\ m, c]; k \leq x - r]\!] \Longrightarrow$
$(x - k \in [r, mod\ m, c]) = (k\ mod\ m = 0)$
⟨*proof*⟩

corollary *iMODb-pred*:
$[\![x \in [r, mod\ m, c]; r < x]\!] \Longrightarrow$
$(x - Suc\ 0 \in [r, mod\ m, c]) = (m = Suc\ 0)$
⟨*proof*⟩

lemmas *iFROM-plus-minus* =
iFROM-plus
iFROM-Suc
iFROM-minus
iFROM-pred

lemmas *iTILL-plus-minus* =
iTILL-plus
iTILL-Suc
iTILL-minus
iTILL-pred

lemmas *iIN-plus-minus* =
iIN-plus
iIN-Suc
iTILL-minus
iIN-pred

lemmas *iMOD-plus-minus-divisor* =
iMOD-plus-divisor-mult
iMOD-plus-divisor
iMOD-minus-divisor-mult

iMOD-minus-divisor-mult2
iMOD-minus-divisor

lemmas *iMOD-plus-minus* =
iMOD-plus
iMOD-Suc
iMOD-minus
iMOD-pred

lemmas *iMODb-plus-minus-divisor* =
iMODb-plus-divisor-mult
iMODb-plus-divisor-mult2
iMODb-plus-divisor
iMODb-minus-divisor-mult

lemmas *iMODb-plus-minus* =
iMODb-plus
iMODb-Suc
iMODb-minus
iMODb-pred

lemmas *iT-plus-minus* =
iFROM-plus-minus
iTILL-plus-minus
iIN-plus-minus
iMOD-plus-minus-divisor
iMOD-plus-minus
iMODb-plus-minus-divisor
iMODb-plus-minus

1.3 Relations between intervals

1.3.1 Auxiliary lemmata

lemma *Suc-in-imp-not-subset-iMOD*:
$[\![n \in S;\ Suc\ n \in S;\ m \neq Suc\ 0]\!] \Longrightarrow \neg\ S \subseteq [r,\ mod\ m]$
⟨*proof*⟩

corollary *Suc-in-imp-neq-iMOD*:
$[\![n \in S;\ Suc\ n \in S;\ m \neq Suc\ 0]\!] \Longrightarrow S \neq [r,\ mod\ m]$
⟨*proof*⟩

lemma *Suc-in-imp-not-subset-iMODb*:
$[\![n \in S;\ Suc\ n \in S;\ m \neq Suc\ 0]\!] \Longrightarrow \neg\ S \subseteq [r,\ mod\ m,\ c]$
⟨*proof*⟩
corollary *Suc-in-imp-neq-iMODb*:
$[\![n \in S;\ Suc\ n \in S;\ m \neq Suc\ 0]\!] \Longrightarrow S \neq [r,\ mod\ m,\ c]$
⟨*proof*⟩

1.3.2 Subset relation between intervals

lemma
iIN-iFROM-subset-same: $[n\ldots,d] \subseteq [n\ldots]$ **and**
iIN-iTILL-subset-same: $[n\ldots,d] \subseteq [\ldots n+d]$ **and**
iMOD-iFROM-subset-same: $[r,\ mod\ m] \subseteq [r\ldots]$ **and**
iMODb-iTILL-subset-same: $[r,\ mod\ m,\ c] \subseteq [\ldots r+m*c]$ **and**
iMODb-iIN-subset-same: $[r,\ mod\ m,\ c] \subseteq [r\ldots,m*c]$ **and**
iMODb-iMOD-subset-same: $[r,\ mod\ m,\ c] \subseteq [r,\ mod\ m]$
⟨*proof*⟩

lemmas *iT-subset-same* =
iIN-iFROM-subset-same
iIN-iTILL-subset-same
iMOD-iFROM-subset-same
iMODb-iTILL-subset-same
iMODb-iIN-subset-same
iMODb-iTILL-subset-same
iMODb-iMOD-subset-same

lemma *iMODb-imp-iMOD*: $x \in [r,\ mod\ m,\ c] \Longrightarrow x \in [r,\ mod\ m]$
⟨*proof*⟩

lemma *iMOD-imp-iMODb*:
$[\![\ x \in [r,\ mod\ m];\ x \le r + m * c\ ]\!] \Longrightarrow x \in [r,\ mod\ m,\ c]$
⟨*proof*⟩

lemma *iMOD-singleton-subset-conv*: $([r,\ mod\ m] \subseteq \{a\}) = (r = a \wedge m = 0)$
⟨*proof*⟩
lemma *iMOD-singleton-eq-conv*: $([r,\ mod\ m] = \{a\}) = (r = a \wedge m = 0)$
⟨*proof*⟩

lemma *iMODb-singleton-subset-conv*:
$([r,\ mod\ m,\ c] \subseteq \{a\}) = (r = a \wedge (m = 0 \vee c = 0))$
⟨*proof*⟩
lemma *iMODb-singleton-eq-conv*:
$([r,\ mod\ m,\ c] = \{a\}) = (r = a \wedge (m = 0 \vee c = 0))$
⟨*proof*⟩

lemma *iMODb-subset-imp-divisor-mod-0*:
$[\![\ 0 < c';\ [r',\ mod\ m',\ c'] \subseteq [r,\ mod\ m,\ c]\ ]\!] \Longrightarrow m'\ mod\ m = 0$
⟨*proof*⟩

lemma *iMOD-subset-imp-divisor-mod-0*:
$[r',\ mod\ m'] \subseteq [r,\ mod\ m] \Longrightarrow m'\ mod\ m = 0$
⟨*proof*⟩

lemma *iMOD-subset-imp-iMODb-subset*:
$[\![\ [r',\ mod\ m'] \subseteq [r,\ mod\ m];\ r' + m' * c' \le r + m * c\ ]\!] \Longrightarrow$
$[r',\ mod\ m',\ c'] \subseteq [r,\ mod\ m,\ c]$

⟨*proof*⟩

lemma *iMODb-subset-imp-iMOD-subset*:
$[\![[r', mod\ m', c'] \subseteq [r, mod\ m, c]; 0 < c']\!] \Longrightarrow$
$[r', mod\ m'] \subseteq [r, mod\ m]$
⟨*proof*⟩

lemma *iMODb-0-iMOD-subset-conv*:
$([r', mod\ m', 0] \subseteq [r, mod\ m]) =$
$(r'\ mod\ m = r\ mod\ m \wedge r \le r')$
⟨*proof*⟩

lemma *iFROM-subset-conv*: $([n'\ldots] \subseteq [n\ldots]) = (n \le n')$
⟨*proof*⟩

lemma *iFROM-iMOD-subset-conv*: $([n'\ldots] \subseteq [r, mod\ m]) = (r \le n' \wedge m = Suc\ 0)$
⟨*proof*⟩

lemma *iIN-subset-conv*: $([n'\ldots,d'] \subseteq [n\ldots,d]) = (n \le n' \wedge n'+d' \le n+d)$
⟨*proof*⟩

lemma *iIN-iFROM-subset-conv*: $([n'\ldots,d'] \subseteq [n\ldots]) = (n \le n')$
⟨*proof*⟩

lemma *iIN-iTILL-subset-conv*: $([n'\ldots,d'] \subseteq [\ldots n]) = (n' + d' \le n)$
⟨*proof*⟩

lemma *iIN-iMOD-subset-conv*:
$0 < d' \Longrightarrow ([n'\ldots,d'] \subseteq [r, mod\ m]) = (r \le n' \wedge m = Suc\ 0)$
⟨*proof*⟩

lemma *iIN-iMODb-subset-conv*:
$0 < d' \Longrightarrow$
$([n'\ldots,d'] \subseteq [r, mod\ m, c]) =$
$(r \le n' \wedge m = Suc\ 0 \wedge n' + d' \le r + m * c)$
⟨*proof*⟩

lemma *iTILL-subset-conv*: $([\ldots n'] \subseteq [\ldots n]) = (n' \le n)$
⟨*proof*⟩

lemma *iTILL-iFROM-subset-conv*: $([\ldots n'] \subseteq [n\ldots]) = (n = 0)$
⟨*proof*⟩

lemma *iTILL-iIN-subset-conv*: $([\ldots n'] \subseteq [n\ldots,d]) = (n = 0 \wedge n' \le d)$
⟨*proof*⟩

lemma *iTILL-iMOD-subset-conv*:

$0 < n' \Longrightarrow ([\ldots n'] \subseteq [r, \ mod\ m]) = (r = 0 \wedge m = Suc\ 0)$
⟨proof⟩

lemma *iTILL-iMODb-subset-conv*:
$0 < n' \Longrightarrow ([\ldots n'] \subseteq [r, \ mod\ m, \ c]) = (r = 0 \wedge m = Suc\ 0 \wedge n' \leq r + m * c)$
⟨proof⟩

lemma *iMOD-iFROM-subset-conv*: $([r', \ mod\ m']) \subseteq [n\ldots] = (n \leq r')$
⟨proof⟩

lemma *iMODb-iFROM-subset-conv*: $([r', \ mod\ m', \ c'] \subseteq [n\ldots]) = (n \leq r')$
⟨proof⟩

lemma *iMODb-iIN-subset-conv*:
$([r', \ mod\ m', \ c'] \subseteq [n\ldots,d]) = (n \leq r' \wedge r' + m' * c' \leq n + d)$
⟨proof⟩

lemma *iMODb-iTILL-subset-conv*:
$([r', \ mod\ m', \ c'] \subseteq [\ldots n]) = (r' + m' * c' \leq n)$
⟨proof⟩

lemma *iMOD-0-subset-conv*: $([r', \ mod\ 0] \subseteq [r, \ mod\ m]) = (r'\ mod\ m = r\ mod\ m$
$\wedge \ \ r \leq r')$
⟨proof⟩

lemma *iMOD-subset-conv*: $0 < m \Longrightarrow$
$([r', \ mod\ m'] \subseteq [r, \ mod\ m]) =$
$(r'\ mod\ m = r\ mod\ m \ \wedge \ r \leq r' \ \wedge \ m'\ mod\ m = 0)$
⟨proof⟩

lemma *iMODb-subset-mod-0-conv*:
$([r', \ mod\ m', \ c'] \subseteq [r, \ mod\ 0, \ c\ ]) = (r'=r \wedge (m'=0 \vee c'=0))$
⟨proof⟩

lemma *iMODb-subset-0-conv*:
$([r', \ mod\ m', \ c'] \subseteq [r, \ mod\ m, \ 0\ ]) = (r'=r \wedge (m'=0 \vee c'=0))$
⟨proof⟩

lemma *iMODb-0-subset-conv*:
$([r', \ mod\ m', \ 0] \subseteq [r, \ mod\ m, \ c\ ]) = (r' \in [r, \ mod\ m, \ c])$
⟨proof⟩

lemma *iMODb-mod-0-subset-conv*:
$([r', \ mod\ 0, \ c'] \subseteq [r, \ mod\ m, \ c\ ]) = (r' \in [r, \ mod\ m, \ c])$
⟨proof⟩

lemma *iMODb-subset-conv'*: $[\![\ 0 < c;\ 0 < c'\ ]\!] \Longrightarrow$
$([r', \ mod\ m', \ c'] \subseteq [r, \ mod\ m, \ c]) =$
$(r'\ mod\ m = r\ mod\ m \ \wedge \ r \leq r' \ \wedge \ m'\ mod\ m = 0 \ \wedge$

$r' + m' * c' \le r + m * c$)
⟨*proof*⟩

lemma *iMODb-subset-conv*: $[\![\ 0 < m';\ 0 < c'\ ]\!] \Longrightarrow$
$([r', mod\ m', c'] \subseteq [r, mod\ m, c]) =$
$(r'\ mod\ m = r\ mod\ m\ \wedge\ r \le r'\ \wedge\ m'\ mod\ m = 0\ \wedge$
$r' + m' * c' \le r + m * c)$
⟨*proof*⟩

lemma *iMODb-iMOD-subset-conv*: $0 < c' \Longrightarrow$
$([r', mod\ m', c'] \subseteq [r, mod\ m]) =$
$(r'\ mod\ m = r\ mod\ m\ \wedge\ r \le r'\ \wedge\ m'\ mod\ m = 0)$
⟨*proof*⟩

lemmas *iT-subset-conv* =
iFROM-subset-conv
iFROM-iMOD-subset-conv
iTILL-subset-conv
iTILL-iFROM-subset-conv
iTILL-iIN-subset-conv
iTILL-iMOD-subset-conv
iTILL-iMODb-subset-conv
iIN-subset-conv
iIN-iFROM-subset-conv
iIN-iTILL-subset-conv
iIN-iMOD-subset-conv
iIN-iMODb-subset-conv
iMOD-subset-conv
iMOD-iFROM-subset-conv
iMODb-subset-conv'
iMODb-subset-conv
iMODb-iFROM-subset-conv
iMODb-iIN-subset-conv
iMODb-iTILL-subset-conv
iMODb-iMOD-subset-conv

lemma *iFROM-subset*: $n \le n' \Longrightarrow [n'\ldots] \subseteq [n\ldots]$
⟨*proof*⟩

lemma *not-iFROM-iIN-subset*: $\neg\ [n'\ldots] \subseteq [n\ldots,d]$
⟨*proof*⟩

lemma *not-iFROM-iTILL-subset*: $\neg\ [n'\ldots] \subseteq [\ldots n]$
⟨*proof*⟩

lemma *not-iFROM-iMOD-subset*: $m \ne Suc\ 0 \Longrightarrow \neg\ [n'\ldots] \subseteq [r, mod\ m]$
⟨*proof*⟩

lemma *not-iFROM-iMODb-subset*: $\neg\ [n'\ldots] \subseteq [r, mod\ m, c]$

⟨*proof*⟩

lemma *iIN-subset*: $[\![\ n \le n';\ n' + d' \le n + d\ ]\!] \Longrightarrow [n' \ldots, d'] \subseteq [n \ldots, d]$
⟨*proof*⟩

lemma *iIN-iFROM-subset*: $n \le n' \Longrightarrow [n' \ldots, d'] \subseteq [n \ldots]$
⟨*proof*⟩

lemma *iIN-iTILL-subset*: $n' + d' \le n \Longrightarrow [n' \ldots, d'] \subseteq [\ldots n]$
⟨*proof*⟩

lemma *not-iIN-iMODb-subset*: $[\![\ 0 < d';\ m \ne Suc\ 0\ ]\!] \Longrightarrow \neg\ [n' \ldots, d'] \subseteq [r,\ mod\ m,\ c]$
⟨*proof*⟩

lemma *not-iIN-iMOD-subset*: $[\![\ 0 < d';\ m \ne Suc\ 0\ ]\!] \Longrightarrow \neg\ [n' \ldots, d'] \subseteq [r,\ mod\ m]$
⟨*proof*⟩

lemma *iTILL-subset*: $n' \le n \Longrightarrow [\ldots n'] \subseteq [\ldots n]$
⟨*proof*⟩

lemma *iTILL-iFROM-subset*: $([\ldots n'] \subseteq [0 \ldots])$
⟨*proof*⟩

lemma *iTILL-iIN-subset*: $n' \le d \Longrightarrow ([\ldots n'] \subseteq [0 \ldots, d])$
⟨*proof*⟩

lemma *not-iTILL-iMOD-subset*:
$[\![\ 0 < n';\ m \ne Suc\ 0\ ]\!] \Longrightarrow \neg\ [\ldots n'] \subseteq [r,\ mod\ m]$
⟨*proof*⟩

lemma *not-iTILL-iMODb-subset*:
$[\![\ 0 < n';\ m \ne Suc\ 0\ ]\!] \Longrightarrow \neg\ [\ldots n'] \subseteq [r,\ mod\ m,\ c]$
⟨*proof*⟩

lemma *iMOD-iFROM-subset*: $n \le r' \Longrightarrow [r',\ mod\ m'] \subseteq [n \ldots]$
⟨*proof*⟩

lemma *not-iMOD-iIN-subset*: $0 < m' \Longrightarrow \neg\ [r',\ mod\ m'] \subseteq [n \ldots, d]$
⟨*proof*⟩

lemma *not-iMOD-iTILL-subset*: $0 < m' \Longrightarrow \neg\ [r',\ mod\ m'] \subseteq [\ldots n]$
⟨*proof*⟩

lemma *iMOD-subset*:
$[\![\ r \le r';\ r'\ mod\ m = r\ mod\ m;\ m'\ mod\ m = 0\ ]\!] \Longrightarrow [r',\ mod\ m'] \subseteq [r,\ mod\ m]$
⟨*proof*⟩

lemma *not-iMOD-iMODb-subset*: $0 < m' \Longrightarrow \neg\ [r', mod\ m'] \subseteq [r, mod\ m, c]$
⟨*proof*⟩

lemma *iMODb-iFROM-subset*: $n \le r' \Longrightarrow [r', mod\ m', c'] \subseteq [n\ldots]$
⟨*proof*⟩

lemma *iMODb-iTILL-subset*:
$r' + m' * c' \le n \Longrightarrow [r', mod\ m', c'] \subseteq [\ldots n]$
⟨*proof*⟩

lemma *iMODb-iIN-subset*:
$[\![\ n \le r';\ r' + m' * c' \le n + d\ ]\!] \Longrightarrow [r', mod\ m', c'] \subseteq [n\ldots,d]$
⟨*proof*⟩

lemma *iMODb-iMOD-subset*:
$[\![\ r \le r';\ r'\ mod\ m = r\ mod\ m;\ m'\ mod\ m = 0\ ]\!] \Longrightarrow [r', mod\ m', c'] \subseteq [r, mod\ m]$
⟨*proof*⟩

lemma *iMODb-subset*:
$[\![\ r \le r';\ r'\ mod\ m = r\ mod\ m;\ m'\ mod\ m = 0;\ r' + m' * c' \le r + m * c\ ]\!] \Longrightarrow$
$[r', mod\ m', c'] \subseteq [r, mod\ m, c]$
⟨*proof*⟩

lemma *iFROM-trans*: $[\![\ y \in [x\ldots];\ z \in [y\ldots]\ ]\!] \Longrightarrow z \in [x\ldots]$
⟨*proof*⟩

lemma *iTILL-trans*: $[\![\ y \in [\ldots x];\ z \in [\ldots y]\ ]\!] \Longrightarrow z \in [\ldots x]$
⟨*proof*⟩

lemma *iIN-trans*:
$[\![\ y \in [x\ldots,d];\ z \in [y\ldots,d'];\ d' \le x + d - y\ ]\!] \Longrightarrow z \in [x\ldots,d]$
⟨*proof*⟩

lemma *iMOD-trans*:
$[\![\ y \in [x, mod\ m];\ z \in [y, mod\ m]\ ]\!] \Longrightarrow z \in [x, mod\ m]$
⟨*proof*⟩

lemma *iMODb-trans*:
$[\![\ y \in [x, mod\ m, c];\ z \in [y, mod\ m, c'];\ m * c' \le x + m * c - y\ ]\!] \Longrightarrow$
$z \in [x, mod\ m, c]$
⟨*proof*⟩

lemma *iMODb-trans'*:
$[\![\ y \in [x, mod\ m, c];\ z \in [y, mod\ m, c'];\ c' \le x\ div\ m + c - y\ div\ m\ ]\!] \Longrightarrow$
$z \in [x, mod\ m, c]$
⟨*proof*⟩

1.3.3 Equality of intervals

lemma *iFROM-eq-conv*: $([n\ldots] = [n'\ldots]) = (n = n')$
⟨*proof*⟩

lemma *iIN-eq-conv*: $([n\ldots,d] = [n'\ldots,d']) = (n = n' \wedge d = d')$
⟨*proof*⟩

lemma *iTILL-eq-conv*: $([\ldots n] = [\ldots n']) = (n = n')$
⟨*proof*⟩

lemma *iMOD-0-eq-conv*: $([r,\ mod\ 0] = [r',\ mod\ m']) = (r = r' \wedge m' = 0)$
⟨*proof*⟩

lemma *iMOD-eq-conv*: $0 < m \Longrightarrow ([r,\ mod\ m] = [r',\ mod\ m']) = (r = r' \wedge m = m')$
⟨*proof*⟩

lemma *iMODb-mod-0-eq-conv*:
$([r,\ mod\ 0,\ c] = [r',\ mod\ m',\ c']) = (r = r' \wedge (m' = 0 \vee c' = 0))$
⟨*proof*⟩

lemma *iMODb-0-eq-conv*:
$([r,\ mod\ m,\ 0] = [r',\ mod\ m',\ c']) = (r = r' \wedge (m' = 0 \vee c' = 0))$
⟨*proof*⟩

lemma *iMODb-eq-conv*: $[\![\ 0 < m;\ 0 < c\ ]\!] \Longrightarrow$
$([r,\ mod\ m,\ c] = [r',\ mod\ m',\ c']) = (r = r' \wedge m = m' \wedge c = c')$
⟨*proof*⟩

lemma *iMOD-iFROM-eq-conv*: $([n\ldots] = [r,\ mod\ m]) = (n = r \wedge m = Suc\ 0)$
⟨*proof*⟩

lemma *iMODb-iIN-0-eq-conv*:
$([n\ldots,0] = [r,\ mod\ m,\ c]) = (n = r \wedge (m = 0 \vee c = 0))$
⟨*proof*⟩

lemma *iMODb-iIN-eq-conv*:
$0 < d \Longrightarrow ([n\ldots,d] = [r,\ mod\ m,\ c]) = (n = r \wedge m = Suc\ 0 \wedge c = d)$
⟨*proof*⟩

1.3.4 Inequality of intervals

lemma *iFROM-iIN-neq*: $[n'\ldots] \neq [n\ldots,d]$
⟨*proof*⟩

corollary *iFROM-iTILL-neq*: $[n'\ldots] \neq [\ldots n]$
⟨*proof*⟩

corollary *iFROM-iMOD-neq*: $m \neq Suc\ 0 \Longrightarrow [n\ldots] \neq [r,\ mod\ m]$

⟨*proof*⟩
corollary *iFROM-iMODb-neq*: $[n\ldots] \neq [r,\ mod\ m,\ c]$
⟨*proof*⟩

corollary *iIN-iMOD-neq*: $0 < m \Longrightarrow [n\ldots,d] \neq [r,\ mod\ m]$
⟨*proof*⟩

corollary *iIN-iMODb-neq2*: $[\![\ m \neq Suc\ 0;\ 0 < d\ ]\!] \Longrightarrow [n\ldots,d] \neq [r,\ mod\ m,\ c]$
⟨*proof*⟩

lemma *iIN-iMODb-neq*: $[\![\ 2 \leq m;\ 0 < c\ ]\!] \Longrightarrow [n\ldots,d] \neq [r,\ mod\ m,\ c]$
⟨*proof*⟩

lemma *iTILL-iIN-neq*: $0 < n \Longrightarrow [\ldots n'] \neq [n\ldots,d]$
⟨*proof*⟩

corollary *iTILL-iMOD-neq*: $0 < m \Longrightarrow [\ldots n] \neq [r,\ mod\ m]$
⟨*proof*⟩

corollary *iTILL-iMODb-neq*:
$[\![\ m \neq Suc\ 0;\ 0 < n\ ]\!] \Longrightarrow [\ldots n] \neq [r,\ mod\ m,\ c]$
⟨*proof*⟩

lemma *iMOD-iMODb-neq*: $0 < m \Longrightarrow [r,\ mod\ m] \neq [r',\ mod\ m',\ c']$
⟨*proof*⟩

lemmas *iT-neq* =
iFROM-iTILL-neq iFROM-iIN-neq iFROM-iMOD-neq iFROM-iMODb-neq
iTILL-iIN-neq iTILL-iMOD-neq iTILL-iMODb-neq
iIN-iMOD-neq iIN-iMODb-neq iIN-iMODb-neq2
iMOD-iMODb-neq

1.4 Union and intersection of intervals

lemma *iFROM-union'*: $[n\ldots] \cup [n'\ldots] = [min\ n\ n'\ldots]$
⟨*proof*⟩

corollary *iFROM-union*: $n \leq n' \Longrightarrow [n\ldots] \cup [n'\ldots] = [n\ldots]$
⟨*proof*⟩

lemma *iFROM-inter'*: $[n\ldots] \cap [n'\ldots] = [max\ n\ n'\ldots]$
⟨*proof*⟩

corollary *iFROM-inter*: $n' \leq n \Longrightarrow [n\ldots] \cap [n'\ldots] = [n\ldots]$
⟨*proof*⟩

lemma *iTILL-union'*: $[\ldots n] \cup [\ldots n'] = [\ldots max\ n\ n']$
⟨*proof*⟩

corollary *iTILL-union*: $n' \le n \Longrightarrow [\ldots n] \cup [\ldots n'] = [\ldots n]$
⟨*proof*⟩

lemma *iTILL-iFROM-union*: $n \le n' \Longrightarrow [\ldots n'] \cup [n\ldots] = UNIV$
⟨*proof*⟩

lemma *iTILL-inter'*: $[\ldots n] \cap [\ldots n'] = [\ldots min\ n\ n']$
⟨*proof*⟩

corollary *iTILL-inter*: $n \le n' \Longrightarrow [\ldots n] \cap [\ldots n'] = [\ldots n]$
⟨*proof*⟩

Union and intersection for iIN only when there intersection is not empty and none of them is other's subset, for instance: .. n .. n+d .. n' .. n'+d'

lemma *iIN-union*:
$[\![n \le n'; n' \le Suc\ (n + d); n + d \le n' + d']\!] \Longrightarrow$
$[n\ldots,d] \cup [n'\ldots,d'] = [n\ldots,n' - n + d']$
⟨*proof*⟩

lemma *iIN-append-union*:
$[n\ldots,d] \cup [n + d\ldots,d'] = [n\ldots,d + d']$
⟨*proof*⟩

lemma *iIN-append-union-Suc*:
$[n\ldots,d] \cup [Suc\ (n + d)\ldots,d'] = [n\ldots,Suc\ (d + d')]$
⟨*proof*⟩

lemma *iIN-append-union-pred*:
$0 < d \Longrightarrow [n\ldots,d - Suc\ 0] \cup [n + d\ldots,d'] = [n\ldots,d + d']$
⟨*proof*⟩

lemma *iIN-iFROM-union*:
$n' \le Suc\ (n + d) \Longrightarrow [n\ldots,d] \cup [n'\ldots] = [min\ n\ n'\ldots]$
⟨*proof*⟩

lemma *iIN-iFROM-append-union*:
$[n\ldots,d] \cup [n + d\ldots] = [n\ldots]$
⟨*proof*⟩

lemma *iIN-iFROM-append-union-Suc*:
$[n\ldots,d] \cup [Suc\ (n + d)\ldots] = [n\ldots]$
⟨*proof*⟩

lemma *iIN-iFROM-append-union-pred*:
$0 < d \Longrightarrow [n\ldots,d - Suc\ 0] \cup [n + d\ldots] = [n\ldots]$
⟨*proof*⟩

lemma *iIN-inter*:
$[\![n \le n'; n' \le n + d; n + d \le n' + d']\!] \Longrightarrow$
$[n\ldots,d] \cap [n'\ldots,d'] = [n'\ldots,n + d - n']$
⟨*proof*⟩

lemma *iMOD-union*:
$[\![r \le r'; r\ mod\ m = r'\ mod\ m]\!] \Longrightarrow$
$[r,\ mod\ m] \cup [r',\ mod\ m] = [r,\ mod\ m]$
⟨*proof*⟩
lemma *iMOD-union'*:
$r\ mod\ m = r'\ mod\ m \Longrightarrow$
$[r,\ mod\ m] \cup [r',\ mod\ m] = [min\ r\ r',\ mod\ m]$
⟨*proof*⟩

lemma *iMOD-inter*:
$[\![r \le r'; r\ mod\ m = r'\ mod\ m]\!] \Longrightarrow$
$[r,\ mod\ m] \cap [r',\ mod\ m] = [r',\ mod\ m]$
⟨*proof*⟩

lemma *iMOD-inter'*:
$r\ mod\ m = r'\ mod\ m \Longrightarrow$
$[r,\ mod\ m] \cap [r',\ mod\ m] = [max\ r\ r',\ mod\ m]$
⟨*proof*⟩

lemma *iMODb-union*:
$[\![r \le r'; r\ mod\ m = r'\ mod\ m; r' \le r + m * c; r + m * c \le r' + m * c']\!] \Longrightarrow$
$[r,\ mod\ m,\ c] \cup [r',\ mod\ m,\ c'] = [r,\ mod\ m,\ r'\ div\ m - r\ div\ m + c']$
⟨*proof*⟩

lemma *iMODb-append-union*:
$[r,\ mod\ m,\ c] \cup [\ r + m * c,\ mod\ m,\ c'] = [r,\ mod\ m,\ c + c']$
⟨*proof*⟩

lemma *iMODb-iMOD-append-union'*:
$[\![r\ mod\ m = r'\ mod\ m; r' \le r + m * Suc\ c]\!] \Longrightarrow$
$[r,\ mod\ m,\ c] \cup [\ r',\ mod\ m\ ] = [min\ r\ r',\ mod\ m]$
⟨*proof*⟩

lemma *iMODb-iMOD-append-union*:
$[\![r \le r'; r\ mod\ m = r'\ mod\ m; r' \le r + m * Suc\ c]\!] \Longrightarrow$
$[r,\ mod\ m,\ c] \cup [\ r',\ mod\ m\ ] = [r,\ mod\ m]$
⟨*proof*⟩

lemma *iMODb-append-union-Suc*:
$[r,\ mod\ m,\ c] \cup [\ r + m * Suc\ c,\ mod\ m,\ c'] = [r,\ mod\ m,\ Suc\ (c + c')]$
⟨*proof*⟩

lemma *iMODb-append-union-pred*:
$0 < c \Longrightarrow [r,\ mod\ m,\ c - Suc\ 0] \cup [\ r + m * c,\ mod\ m,\ c'] = [r,\ mod\ m,\ c +$

c']
⟨*proof*⟩

lemma *iMODb-inter*:
$[\![r \le r'; r\ mod\ m = r'\ mod\ m; r' \le r + m * c; r + m * c \le r' + m * c']\!] \Longrightarrow$
$[r,\ mod\ m,\ c] \cap [r',\ mod\ m,\ c'] = [r',\ mod\ m,\ c - (r' - r)\ div\ m]$
⟨*proof*⟩

lemmas *iT-union′* =
iFROM-union′
iTILL-union′
iMOD-union′
iMODb-iMOD-append-union′
lemmas *iT-union* =
iFROM-union
iTILL-union
iTILL-iFROM-union
iIN-union
iIN-iFROM-union
iMOD-union
iMODb-union
lemmas *iT-union-append* =
iIN-append-union
iIN-append-union-Suc
iIN-append-union-pred
iIN-iFROM-append-union
iIN-iFROM-append-union-Suc
iIN-iFROM-append-union-pred
iMODb-append-union
iMODb-iMOD-append-union
iMODb-append-union-Suc
iMODb-append-union-pred

lemmas *iT-inter′* =
iFROM-inter′
iTILL-inter′
iMOD-inter′
lemmas *iT-inter* =
iFROM-inter
iTILL-inter
iIN-inter
iMOD-inter
iMODb-inter

lemma *mod-partition-Union*:
$0 < m \Longrightarrow (\bigcup k.\ A \cap [k * m \ldots, m - Suc\ 0]) = A$

⟨*proof*⟩

lemma *finite-mod-partition-Union*:
$[\![\ 0 < m;\ finite\ A\ ]\!] \Longrightarrow$
$(\bigcup k \le Max\ A\ div\ m.\ A \cap [k*m\ldots,m - Suc\ 0]) = A$
⟨*proof*⟩

lemma *mod-partition-is-disjoint*:
$[\![\ 0 < (m::nat);\ k \neq k'\ ]\!] \Longrightarrow$
$(A \cap [k * m\ldots,m - Suc\ 0]) \cap (A \cap [k' * m\ldots,m - Suc\ 0]) = \{\}$
⟨*proof*⟩

1.5 Cutting intervals

lemma *iTILL-cut-le*: $[\ldots n] \downarrow\le t = (if\ t \le n\ then\ [\ldots t]\ else\ [\ldots n])$
⟨*proof*⟩

corollary *iTILL-cut-le1*: $t \in [\ldots n] \Longrightarrow [\ldots n] \downarrow\le t = [\ldots t]$
⟨*proof*⟩

corollary *iTILL-cut-le2*: $t \notin [\ldots n] \Longrightarrow [\ldots n] \downarrow\le t = [\ldots n]$
⟨*proof*⟩

lemma *iFROM-cut-le*:
$[n\ldots] \downarrow\le t =$
$(if\ t < n\ then\ \{\}\ else\ [n\ldots,t-n])$
⟨*proof*⟩

corollary *iFROM-cut-le1*: $t \in [n\ldots] \Longrightarrow [n\ldots] \downarrow\le t = [n\ldots,t - n]$
⟨*proof*⟩

lemma *iIN-cut-le*:
$[n\ldots,d] \downarrow\le t = ($
$if\ t < n\ then\ \{\}\ else$
$if\ t \le n+d\ then\ [n\ldots,t-n]$
$else\ [n\ldots,d])$
⟨*proof*⟩

corollary *iIN-cut-le1*:
$t \in [n\ldots,d] \Longrightarrow [n\ldots,d] \downarrow\le t = [n\ldots,t - n]$
⟨*proof*⟩

lemma *iMOD-cut-le*:
$[r,\ mod\ m] \downarrow\le t = ($
$if\ t < r\ then\ \{\}$
$else\ [r,\ mod\ m,\ (t - r)\ div\ m])$
⟨*proof*⟩

lemma *iMOD-cut-le1*:
$t \in [r, mod\ m] \Longrightarrow$
$[r, mod\ m] \downarrow\leq t = [r, mod\ m, (t - r)\ div\ m]$
⟨*proof*⟩

lemma *iMODb-cut-le*:
$[r, mod\ m, c] \downarrow\leq t = ($
$if\ t < r\ then\ \{\}$
$else\ if\ t < r + m * c\ then\ [r, mod\ m, (t - r)\ div\ m]$
$else\ [r, mod\ m, c])$
⟨*proof*⟩

lemma *iMODb-cut-le1*:
$t \in [r, mod\ m, c] \Longrightarrow$
$[r, mod\ m, c] \downarrow\leq t = [r, mod\ m, (t - r)\ div\ m]$
⟨*proof*⟩

lemma *iTILL-cut-less*:
$[\ldots n] \downarrow< t = ($
$if\ n < t\ then\ [\ldots n]\ else$
$if\ t = 0\ then\ \{\}$
$else\ [\ldots t - Suc\ 0])$
⟨*proof*⟩

lemma *iTILL-cut-less1*:
$[\![\ t \in [\ldots n];\ 0 < t\ ]\!] \Longrightarrow [\ldots n] \downarrow< t = [\ldots t - Suc\ 0]$
⟨*proof*⟩

lemma *iFROM-cut-less*:
$[n\ldots] \downarrow< t = ($
$if\ t \leq n\ then\ \{\}$
$else\ [n\ldots,t - Suc\ n])$
⟨*proof*⟩

lemma *iFROM-cut-less1*:
$n < t \Longrightarrow [n\ldots] \downarrow< t = [n\ldots,t - Suc\ n]$
⟨*proof*⟩

lemma *iIN-cut-less*:
$[n\ldots,d] \downarrow< t = ($
$if\ t \leq n\ then\ \{\}\ else$
$if\ t \leq n + d\ then\ [n\ldots, t - Suc\ n]$
$else\ [n\ldots,d])$
⟨*proof*⟩

lemma *iIN-cut-less1*:
$[\![t \in [n\ldots,d];\ n < t]\!] \Longrightarrow [n\ldots,d] \downarrow< t = [n\ldots,\ t - Suc\ n]$
⟨*proof*⟩

lemma *iMOD-cut-less*:
$[r,\ mod\ m] \downarrow< t = ($
$\quad if\ t \le r\ then\ \{\}$
$\quad else\ [r,\ mod\ m,\ (t - Suc\ r)\ div\ m])$
⟨*proof*⟩

lemma *iMOD-cut-less1*:
$[\![t \in [r,\ mod\ m];\ r < t]\!] \Longrightarrow$
$[r,\ mod\ m] \downarrow< t = [r,\ mod\ m,\ (t - r)\ div\ m - Suc\ 0]$
⟨*proof*⟩

lemma *iMODb-cut-less*:
$[r,\ mod\ m,\ c] \downarrow< t = ($
$\quad if\ t \le r\ then\ \{\}\ else$
$\quad if\ r + m * c < t\ then\ [r,\ mod\ m,\ c]$
$\quad else\ [r,\ mod\ m,\ (t - Suc\ r)\ div\ m])$
⟨*proof*⟩

lemma *iMODb-cut-less1*: $[\![t \in [r,\ mod\ m,\ c];\ r < t]\!] \Longrightarrow$
$[r,\ mod\ m,\ c] \downarrow< t = [r,\ mod\ m,\ (t - r)\ div\ m - Suc\ 0]$
⟨*proof*⟩

lemmas *iT-cut-le* =
iTILL-cut-le
iFROM-cut-le
iIN-cut-le
iMOD-cut-le
iMODb-cut-le

lemmas *iT-cut-le1* =
iTILL-cut-le1
iFROM-cut-le1
iIN-cut-le1
iMOD-cut-le1
iMODb-cut-le1

lemmas *iT-cut-less* =
iTILL-cut-less
iFROM-cut-less
iIN-cut-less
iMOD-cut-less
iMODb-cut-less

lemmas *iT-cut-less1* =
iTILL-cut-less1
iFROM-cut-less1
iIN-cut-less1
iMOD-cut-less1
iMODb-cut-less1

lemmas *iT-cut-le-less* =
iTILL-cut-le
iTILL-cut-less
iFROM-cut-le
iFROM-cut-less
iIN-cut-le
iIN-cut-less
iMOD-cut-le
iMOD-cut-less
iMODb-cut-le
iMODb-cut-less

lemmas *iT-cut-le-less1* =
iTILL-cut-le1
iTILL-cut-less1
iFROM-cut-le1
iFROM-cut-less1
iIN-cut-le1
iIN-cut-less1
iMOD-cut-le1
iMOD-cut-less1
iMODb-cut-le1
iMODb-cut-less1

lemma *iTILL-cut-ge*:
$[\ldots n] \downarrow\geq t = (\mathit{if}\ n < t\ \mathit{then}\ \{\}\ \mathit{else}\ [t\ldots,n-t])$
⟨*proof*⟩

corollary *iTILL-cut-ge1*: $t \in [\ldots n] \Longrightarrow [\ldots n] \downarrow\geq t = [t\ldots,n-t]$
⟨*proof*⟩

corollary *iTILL-cut-ge2*: $t \notin [\ldots n] \Longrightarrow [\ldots n] \downarrow\geq t = \{\}$
⟨*proof*⟩

lemma *iTILL-cut-greater*:
$[\ldots n] \downarrow> t = (\mathit{if}\ n \leq t\ \mathit{then}\ \{\}\ \mathit{else}\ [\mathit{Suc}\ t\ldots,n - \mathit{Suc}\ t])$
⟨*proof*⟩

corollary *iTILL-cut-greater1*:
$t \in [\ldots n] \Longrightarrow t < n \Longrightarrow [\ldots n] \downarrow> t = [\mathit{Suc}\ t\ldots,n - \mathit{Suc}\ t]$
⟨*proof*⟩

corollary *iTILL-cut-greater2*: $t \notin [\ldots n] \Longrightarrow [\ldots n] \downarrow> t = \{\}$
⟨*proof*⟩

lemma *iFROM-cut-ge*:
$[n\ldots] \downarrow\geq t = (\mathit{if}\ n \leq t\ \mathit{then}\ [t\ldots]\ \mathit{else}\ [n\ldots])$
⟨*proof*⟩
corollary *iFROM-cut-ge1*: $t \in [n\ldots] \Longrightarrow [n\ldots] \downarrow\geq t = [t\ldots]$
⟨*proof*⟩

lemma *iFROM-cut-greater*:
$[n\ldots] \downarrow> t = (\mathit{if}\ n \leq t\ \mathit{then}\ [\mathit{Suc}\ t\ldots]\ \mathit{else}\ [n\ldots])$
⟨*proof*⟩
corollary *iFROM-cut-greater1*:
$t \in [n\ldots] \Longrightarrow [n\ldots] \downarrow> t = [\mathit{Suc}\ t\ldots]$
⟨*proof*⟩

lemma *iIN-cut-ge*:
$[n\ldots,d] \downarrow\geq t = ($
$\mathit{if}\ t < n\ \mathit{then}\ [n\ldots,d]\ \mathit{else}$
$\mathit{if}\ t \leq n{+}d\ \mathit{then}\ [t\ldots,n{+}d{-}t]$
$\mathit{else}\ \{\})$
⟨*proof*⟩

corollary *iIN-cut-ge1*: $t \in [n\ldots,d] \Longrightarrow$
$[n\ldots,d] \downarrow\geq t = [t\ldots,n{+}d{-}t]$
⟨*proof*⟩

corollary *iIN-cut-ge2*: $t \notin [n\ldots,d] \Longrightarrow$
$[n\ldots,d] \downarrow\geq t = (\mathit{if}\ t < n\ \mathit{then}\ [n\ldots,d]\ \mathit{else}\ \{\})$
⟨*proof*⟩

lemma *iIN-cut-greater*:
$[n\ldots,d] \downarrow> t = ($
$\mathit{if}\ t < n\ \mathit{then}\ [n\ldots,d]\ \mathit{else}$
$\mathit{if}\ t < n{+}d\ \mathit{then}\ [\mathit{Suc}\ t\ldots,n + d - \mathit{Suc}\ t]$
$\mathit{else}\ \{\})$
⟨*proof*⟩

corollary *iIN-cut-greater1*:
$[\![\ t \in [n\ldots,d];\ t < n + d\ ]\!] \Longrightarrow$
$[n\ldots,d] \downarrow> t = [\mathit{Suc}\ t\ldots,n + d - \mathit{Suc}\ t]$
⟨*proof*⟩

lemma *mod-cut-greater-aux-t-less*:

$[\![\ 0 < (m::nat);\ r \le t\ ]\!] \Longrightarrow$
$t < t + m - (t - r)\ mod\ m$
⟨proof⟩

lemma *mod-cut-greater-aux-le-x*:
$[\![\ (r::nat) \le t;\ t < x;\ x\ mod\ m = r\ mod\ m]\!] \Longrightarrow$
$t + m - (t - r)\ mod\ m \le x$
⟨proof⟩

lemma *iMOD-cut-greater*:
$[r,\ mod\ m] \downarrow> t = ($
$\quad if\ t < r\ then\ [r,\ mod\ m]\ else$
$\quad if\ m = 0\ then\ \{\}$
$\quad else\ [t + m - (t - r)\ mod\ m,\ mod\ m])$
⟨proof⟩

lemma *iMOD-cut-greater1*:
$t \in [r,\ mod\ m] \Longrightarrow$
$[r,\ mod\ m] \downarrow> t = ($
$\quad if\ m = 0\ then\ \{\}$
$\quad else\ [t + m,\ mod\ m])$
⟨proof⟩

lemma *iMODb-cut-greater-aux*:
$[\![\ 0 < m;\ t < r + m * c;\ r \le t]\!] \Longrightarrow$
$(r + m * c - (t + m - (t - r)\ mod\ m))\ div\ m =$
$c - Suc\ ((t - r)\ div\ m)$
⟨proof⟩

lemma *iMODb-cut-greater*:
$[r,\ mod\ m,\ c] \downarrow> t = ($
$\quad if\ t < r\ then\ [r,\ mod\ m,\ c]\ else$
$\quad if\ r + m * c \le t\ then\ \{\}$
$\quad else\ [t + m - (t - r)\ mod\ m,\ mod\ m,\ c - Suc\ ((t-r)\ div\ m)])$
⟨proof⟩

lemma *iMODb-cut-greater1*:
$t \in [r,\ mod\ m,\ c] \Longrightarrow$
$\quad [r,\ mod\ m,\ c] \downarrow> t = ($
$\quad if\ r + m * c \le t\ then\ \{\}$
$\quad else\ [t + m,\ mod\ m,\ c - Suc\ ((t-r)\ div\ m)])$
⟨proof⟩

lemma *iMOD-cut-ge*:
$[r,\ mod\ m] \downarrow\ge t = ($
$\quad if\ t \le r\ then\ [r,\ mod\ m]\ else$
$\quad if\ m = 0\ then\ \{\}$

else $[t + m - Suc\ ((t - Suc\ r)\ mod\ m),\ mod\ m])$
⟨*proof*⟩

lemma *iMOD-cut-ge1*:
$t \in [r,\ mod\ m] \Longrightarrow$
$[r,\ mod\ m] \downarrow\geq t = [t,\ mod\ m]$
⟨*proof*⟩

lemma *iMODb-cut-ge*:
$[r,\ mod\ m,\ c] \downarrow\geq t = ($
if $t \leq r$ *then* $[r,\ mod\ m,\ c]$ *else*
if $r + m * c < t$ *then* $\{\}$
else $[t + m - Suc\ ((t - Suc\ r)\ mod\ m),\ mod\ m,\ c - (t + m - Suc\ r)\ div\ m])$
⟨*proof*⟩

lemma *iMODb-cut-ge1*:
$t \in [r,\ mod\ m,\ c] \Longrightarrow$
$[r,\ mod\ m,\ c] \downarrow\geq t = ($
if $r + m * c < t$ *then* $\{\}$
else $[t,\ mod\ m,\ c - (t - r)\ div\ m])$
⟨*proof*⟩

lemma *iMOD-0-cut-greater*:
$t \in [r,\ mod\ 0] \Longrightarrow [r,\ mod\ 0] \downarrow> t = \{\}$
⟨*proof*⟩
lemma *iMODb-0-cut-greater*: $t \in [r,\ mod\ 0,\ c] \Longrightarrow$
$[r,\ mod\ 0,\ c] \downarrow> t = \{\}$
⟨*proof*⟩

lemmas *iT-cut-ge* =
iTILL-cut-ge
iFROM-cut-ge
iIN-cut-ge
iMOD-cut-ge
iMODb-cut-ge

lemmas *iT-cut-ge1* =
iTILL-cut-ge1
iFROM-cut-ge1
iIN-cut-ge1
iMOD-cut-ge1
iMODb-cut-ge1

lemmas *iT-cut-greater* =
iTILL-cut-greater
iFROM-cut-greater
iIN-cut-greater

iMOD-cut-greater
iMODb-cut-greater

lemmas *iT-cut-greater1* =
iTILL-cut-greater1
iFROM-cut-greater1
iIN-cut-greater1
iMOD-cut-greater1
iMODb-cut-greater1

lemmas *iT-cut-ge-greater* =
iTILL-cut-ge
iTILL-cut-greater
iFROM-cut-ge
iFROM-cut-greater
iIN-cut-ge
iIN-cut-greater
iMOD-cut-ge
iMOD-cut-greater
iMODb-cut-ge
iMODb-cut-greater
lemmas *iT-cut-ge-greater1* =
iTILL-cut-ge1
iTILL-cut-greater1
iFROM-cut-ge1
iFROM-cut-greater1
iIN-cut-ge1
iIN-cut-greater1
iMOD-cut-ge1
iMOD-cut-greater1
iMODb-cut-ge1
iMODb-cut-greater1

1.6 Cardinality of intervals

lemma *iFROM-card*: *card* [*n*. . .] = *0*
⟨*proof*⟩

lemma *iTILL-card*: *card* [. . .*n*] = *Suc n*
⟨*proof*⟩

lemma *iIN-card*: *card* [*n*. . .,*d*] = *Suc d*
⟨*proof*⟩

lemma *iMOD-0-card*: *card* [*r*, *mod 0*] = *Suc 0*
⟨*proof*⟩

lemma *iMOD-card*: *0* < *m* ⟹ *card* [*r*, *mod m*] = *0*
⟨*proof*⟩

lemma *iMOD-card-if*: $card\ [r,\ mod\ m] = (if\ m = 0\ then\ Suc\ 0\ else\ 0)$
⟨*proof*⟩

lemma *iMODb-mod-0-card*: $card\ [r,\ mod\ 0,\ c] = Suc\ 0$
⟨*proof*⟩

lemma *iMODb-card*: $0 < m \Longrightarrow card\ [r,\ mod\ m,\ c] = Suc\ c$
⟨*proof*⟩

lemma *iMODb-card-if*:
$card\ [r,\ mod\ m,\ c] = (if\ m = 0\ then\ Suc\ 0\ else\ Suc\ c)$
⟨*proof*⟩

lemmas *iT-card* =
iFROM-card
iTILL-card
iIN-card
iMOD-card-if
iMODb-card-if

Cardinality with *icard*

lemma *iFROM-icard*: $icard\ [n\ldots] = \infty$
⟨*proof*⟩

lemma *iTILL-icard*: $icard\ [\ldots n] = enat\ (Suc\ n)$
⟨*proof*⟩

lemma *iIN-icard*: $icard\ [n\ldots,d] = enat\ (Suc\ d)$
⟨*proof*⟩

lemma *iMOD-0-icard*: $icard\ [r,\ mod\ 0] = eSuc\ 0$
⟨*proof*⟩

lemma *iMOD-icard*: $0 < m \Longrightarrow icard\ [r,\ mod\ m] = \infty$
⟨*proof*⟩

lemma *iMOD-icard-if*: $icard\ [r,\ mod\ m] = (if\ m = 0\ then\ eSuc\ 0\ else\ \infty)$
⟨*proof*⟩

lemma *iMODb-mod-0-icard*: $icard\ [r,\ mod\ 0,\ c] = eSuc\ 0$
⟨*proof*⟩

lemma *iMODb-icard*: $0 < m \Longrightarrow icard\ [r,\ mod\ m,\ c] = enat\ (Suc\ c)$
⟨*proof*⟩

lemma *iMODb-icard-if*: $icard\ [r,\ mod\ m,\ c] = enat\ (if\ m = 0\ then\ Suc\ 0\ else\ Suc\ c)$
⟨*proof*⟩

lemmas *iT-icard* =
iFROM-icard
iTILL-icard
iIN-icard
iMOD-icard-if
iMODb-icard-if

1.7 Functions *inext* and *iprev* with intervals

lemma
iFROM-inext: $t \in [n\ldots] \Longrightarrow \mathit{inext}\ t\ [n\ldots] = \mathit{Suc}\ t$ **and**
iTILL-inext: $t < n \Longrightarrow \mathit{inext}\ t\ [\ldots n] = \mathit{Suc}\ t$ **and**
iIN-inext: $[\![\ n \le t;\ t < n + d\ ]\!] \Longrightarrow \mathit{inext}\ t\ [n\ldots,d] = \mathit{Suc}\ t$
⟨*proof*⟩

lemma
iFROM-iprev': $t \in [n\ldots] \Longrightarrow \mathit{iprev}\ (\mathit{Suc}\ t)\ [n\ldots] = t$ **and**
iFROM-iprev: $n < t \Longrightarrow \mathit{iprev}\ t\ [n\ldots] = t - \mathit{Suc}\ 0$ **and**
iTILL-iprev: $t \in [\ldots n] \Longrightarrow \mathit{iprev}\ t\ [\ldots n] = t - \mathit{Suc}\ 0$ **and**
iIN-iprev: $[\![\ n < t;\ t \le n + d\ ]\!] \Longrightarrow \mathit{iprev}\ t\ [n\ldots,d] = t - \mathit{Suc}\ 0$ **and**
iIN-iprev': $[\![\ n \le t;\ t < n + d\ ]\!] \Longrightarrow \mathit{iprev}\ (\mathit{Suc}\ t)\ [n\ldots,d] = t$
⟨*proof*⟩

lemma *iMOD-inext*: $t \in [r,\ \mathit{mod}\ m] \Longrightarrow \mathit{inext}\ t\ [r,\ \mathit{mod}\ m] = t + m$
⟨*proof*⟩
lemma *iMOD-iprev*: $[\![\ t \in [r,\ \mathit{mod}\ m];\ r < t\ ]\!] \Longrightarrow \mathit{iprev}\ t\ [r,\ \mathit{mod}\ m] = t - m$
⟨*proof*⟩

lemma *iMOD-iprev'*: $t \in [r,\ \mathit{mod}\ m] \Longrightarrow \mathit{iprev}\ (t + m)\ [r,\ \mathit{mod}\ m] = t$
⟨*proof*⟩

lemma *iMODb-inext*:
$[\![\ t \in [r,\ \mathit{mod}\ m,\ c];\ t < r + m * c\ ]\!] \Longrightarrow$
$\mathit{inext}\ t\ [r,\ \mathit{mod}\ m,\ c] = t + m$
⟨*proof*⟩

lemma *iMODb-iprev*:
$[\![\ t \in [r,\ \mathit{mod}\ m,\ c];\ r < t\ ]\!] \Longrightarrow$
$\mathit{iprev}\ t\ [r,\ \mathit{mod}\ m,\ c] = t - m$
⟨*proof*⟩

lemma *iMODb-iprev'*:
$[\![\ t \in [r,\ \mathit{mod}\ m,\ c];\ t < r + m * c\ ]\!] \Longrightarrow$
$\mathit{iprev}\ (t + m)\ [r,\ \mathit{mod}\ m,\ c] = t$
⟨*proof*⟩

lemmas *iT-inext* =
iFROM-inext

iTILL-inext
iIN-inext
iMOD-inext
iMODb-inext
lemmas *iT-iprev* =
iFROM-iprev′
iFROM-iprev
iTILL-iprev
iIN-iprev
iIN-iprev′
iMOD-iprev
iMOD-iprev′
iMODb-iprev
iMODb-iprev′

lemma *iFROM-inext-if*:
inext t [*n*. . .] = (*if t* ∈ [*n*. . .] *then Suc t else t*)
⟨*proof*⟩

lemma *iTILL-inext-if*:
inext t [. . .*n*] = (*if t* < *n then Suc t else t*)
⟨*proof*⟩

lemma *iIN-inext-if*:
inext t [*n*. . .,*d*] = (*if n* ≤ *t* ∧ *t* < *n* + *d then Suc t else t*)
⟨*proof*⟩

lemma *iMOD-inext-if*:
inext t [*r*, *mod m*] = (*if t* ∈ [*r*, *mod m*] *then t* + *m else t*)
⟨*proof*⟩

lemma *iMODb-inext-if*:
inext t [*r*, *mod m*, *c*] =
(*if t* ∈ [*r*, *mod m*, *c*] ∧ *t* < *r* + *m* ∗ *c then t* + *m else t*)
⟨*proof*⟩

lemmas *iT-inext-if* =
iFROM-inext-if
iTILL-inext-if
iIN-inext-if
iMOD-inext-if
iMODb-inext-if

lemma *iFROM-iprev-if*:
iprev t [*n*. . .] = (*if n* < *t then t* − *Suc 0 else t*)
⟨*proof*⟩
lemma *iTILL-iprev-if*:
iprev t [. . .*n*] = (*if t* ∈ [. . .*n*] *then t* − *Suc 0 else t*)
⟨*proof*⟩

lemma *iIN-iprev-if*:
$iprev\ t\ [n\ldots,d] = (if\ n < t \wedge t \leq n + d\ \ then\ t - Suc\ 0\ else\ t)$
⟨*proof*⟩
lemma *iMOD-iprev-if*:
$iprev\ t\ [r,\ mod\ m] =$
$(if\ t \in [r,\ mod\ m] \wedge r < t\ then\ t - m\ else\ t)$
⟨*proof*⟩
lemma *iMODb-iprev-if*:
$iprev\ t\ [r,\ mod\ m,\ c] =$
$(if\ t \in [r,\ mod\ m,\ c] \wedge r < t\ then\ t - m\ else\ t)$
⟨*proof*⟩

lemmas *iT-iprev-if* =
iFROM-iprev-if
iTILL-iprev-if
iIN-iprev-if
iMOD-iprev-if
iMODb-iprev-if

The difference between an element and the next/previous element is constant if the element is different from Min/Max of the interval

lemma *iFROM-inext-diff-const*:
$t \in [n\ldots] \Longrightarrow inext\ t\ [n\ldots] - t = Suc\ 0$
⟨*proof*⟩

lemma *iFROM-iprev-diff-const*:
$n < t \Longrightarrow t - iprev\ t\ [n\ldots] = Suc\ 0$
⟨*proof*⟩

lemma *iFROM-iprev-diff-const'*:
$t \in [n\ldots] \Longrightarrow Suc\ t - iprev\ (Suc\ t)\ [n\ldots] = Suc\ 0$
⟨*proof*⟩

lemma *iTILL-inext-diff-const*:
$t < n \Longrightarrow inext\ t\ [\ldots n] - t = Suc\ 0$
⟨*proof*⟩
lemma *iTILL-iprev-diff-const*:
$[\![\ t \in [\ldots n];\ 0 < t\ ]\!] \Longrightarrow t - iprev\ t\ [\ldots n] = Suc\ 0$
⟨*proof*⟩

lemma *iIN-inext-diff-const*:
$[\![\ n \leq t;\ t < n + d\ ]\!] \Longrightarrow inext\ t\ [n\ldots,d] - t = Suc\ 0$
⟨*proof*⟩

lemma *iIN-iprev-diff-const*:
$[\![\ n < t;\ t \leq n + d\ ]\!] \Longrightarrow t - iprev\ t\ [n\ldots,d] = Suc\ 0$
⟨*proof*⟩
lemma *iIN-iprev-diff-const'*:
$[\![\ n \leq t;\ t < n + d\ ]\!] \Longrightarrow Suc\ t - iprev\ (Suc\ t)\ [n\ldots,d] = Suc\ 0$

⟨*proof*⟩

lemma *iMOD-inext-diff-const*:
$t \in [r,\ mod\ m] \Longrightarrow inext\ t\ [r,\ mod\ m] - t = m$
⟨*proof*⟩

lemma *iMOD-iprev-diff-const′*:
$t \in [r,\ mod\ m] \Longrightarrow (t + m) - iprev\ (t + m)\ [r,\ mod\ m] = m$
⟨*proof*⟩

lemma *iMOD-iprev-diff-const*:
$[\![\ t \in [r,\ mod\ m];\ r < t\ ]\!] \Longrightarrow t - iprev\ t\ [r,\ mod\ m] = m$
⟨*proof*⟩

lemma *iMODb-inext-diff-const*:
$[\![\ t \in [r,\ mod\ m,\ c];\ t < r + m * c\ ]\!] \Longrightarrow inext\ t\ [r,\ mod\ m,\ c] - t = m$
⟨*proof*⟩

lemma *iMODb-iprev-diff-const′*:
$[\![\ t \in [r,\ mod\ m,\ c];\ t < r + m * c\ ]\!] \Longrightarrow (t + m) - iprev\ (t + m)\ [r,\ mod\ m,\ c] = m$
⟨*proof*⟩

lemma *iMODb-iprev-diff-const*:
$[\![\ t \in [r,\ mod\ m,\ c];\ r < t\ ]\!] \Longrightarrow t - iprev\ t\ [r,\ mod\ m,\ c] = m$
⟨*proof*⟩

lemmas *iT-inext-diff-const* =
iFROM-inext-diff-const
iTILL-inext-diff-const
iIN-inext-diff-const
iMOD-inext-diff-const
iMODb-inext-diff-const
lemmas *iT-iprev-diff-const* =
iFROM-iprev-diff-const
iFROM-iprev-diff-const′
iTILL-iprev-diff-const
iIN-iprev-diff-const
iIN-iprev-diff-const′
iMOD-iprev-diff-const′
iMOD-iprev-diff-const
iMODb-iprev-diff-const′
iMODb-iprev-diff-const

1.7.1 Mirroring of intervals

lemma
iIN-mirror-elem: $mirror\text{-}elem\ x\ [n\ldots,d] = n + n + d - x$ **and**
iTILL-mirror-elem: $mirror\text{-}elem\ x\ [\ldots n] = n - x$ **and**

iMODb-mirror-elem: *mirror-elem* x [r, *mod* m, c] = $r + r + m * c - x$
⟨*proof*⟩

lemma *iMODb-imirror-bounds*:
$r' + m' * c' \leq l + r \Longrightarrow$
imirror-bounds [r', *mod* m', c'] l r = [$l + r - r' - m' * c'$, *mod* m', c']
⟨*proof*⟩

lemma *iIN-imirror-bounds*:
$n + d \leq l + r \Longrightarrow$ *imirror-bounds* [$n\ldots$,d] l r = [$l + r - n - d\ldots$,d]
⟨*proof*⟩

lemma *iTILL-imirror-bounds*:
$n \leq l + r \Longrightarrow$ *imirror-bounds* [$\ldots n$] l r = [$l + r - n\ldots$,n]
⟨*proof*⟩

lemmas *iT-imirror-bounds* =
iTILL-imirror-bounds
iIN-imirror-bounds
iMODb-imirror-bounds

lemma *iMODb-imirror-ident*: *imirror* [r, *mod* m, c] = [r, *mod* m, c]
⟨*proof*⟩
lemma *iIN-imirror-ident*: *imirror* [$n\ldots$,d] = [$n\ldots$,d]
⟨*proof*⟩
lemma *iTILL-imirror-ident*: *imirror* [$\ldots n$] = [$\ldots n$]
⟨*proof*⟩

lemmas *iT-imirror-ident* =
iTILL-imirror-ident
iIN-imirror-ident
iMODb-imirror-ident

1.7.2 Functions *inext-nth* and *iprev-nth* on intervals

lemma *iFROM-inext-nth* : [$n\ldots$] → $a = n + a$
⟨*proof*⟩

lemma *iIN-inext-nth* : $a \leq d \Longrightarrow$ [$n\ldots$,d] → $a = n + a$
⟨*proof*⟩

lemma *iIN-iprev-nth*: $a \leq d \Longrightarrow$ [$n\ldots$,d] ← $a = n + d - a$
⟨*proof*⟩

lemma *iIN-inext-nth-if* :
[$n\ldots$,d] → a = (*if* $a \leq d$ *then* $n + a$ *else* $n + d$)
⟨*proof*⟩

lemma *iIN-iprev-nth-if*:
$[n\ldots,d] \leftarrow a = (\mathit{if}\ a \le d\ \mathit{then}\ n + d - a\ \mathit{else}\ n)$
⟨*proof*⟩

lemma *iTILL-inext-nth* : $a \le n \Longrightarrow [\ldots n] \rightarrow a = a$
⟨*proof*⟩

lemma *iTILL-inext-nth-if* :
$[\ldots n] \rightarrow a = (\mathit{if}\ a \le n\ \mathit{then}\ a\ \mathit{else}\ n)$
⟨*proof*⟩

lemma *iTILL-iprev-nth*: $a \le n \Longrightarrow [\ldots n] \leftarrow a = n - a$
⟨*proof*⟩

lemma *iTILL-iprev-nth-if*:
$[\ldots n] \leftarrow a = (\mathit{if}\ a \le n\ \mathit{then}\ n - a\ \mathit{else}\ 0)$
⟨*proof*⟩

lemma *iMOD-inext-nth*: $[r,\ \mathit{mod}\ m] \rightarrow a = r + m * a$
⟨*proof*⟩

lemma *iMODb-inext-nth*: $a \le c \Longrightarrow [r,\ \mathit{mod}\ m,\ c] \rightarrow a = r + m * a$
⟨*proof*⟩

lemma *iMODb-inext-nth-if*:
$[r,\ \mathit{mod}\ m,\ c] \rightarrow a = (\mathit{if}\ a \le c\ \mathit{then}\ r + m * a\ \mathit{else}\ r + m * c)$
⟨*proof*⟩

lemma *iMODb-iprev-nth*:
$a \le c \Longrightarrow [r,\ \mathit{mod}\ m,\ c] \leftarrow a = r + m * (c - a)$
⟨*proof*⟩

lemma *iMODb-iprev-nth-if*:
$[r,\ \mathit{mod}\ m,\ c] \leftarrow a = (\mathit{if}\ a \le c\ \mathit{then}\ r + m * (c - a)\ \mathit{else}\ r)$
⟨*proof*⟩

lemma *iIN-iFROM-inext-nth*:
$a \le d \Longrightarrow [n\ldots,d] \rightarrow a = [n\ldots] \rightarrow a$
⟨*proof*⟩

lemma *iIN-iFROM-inext*:
$a < n + d \Longrightarrow \mathit{inext}\ a\ [n\ldots,d] = \mathit{inext}\ a\ [n\ldots]$
⟨*proof*⟩

lemma *iMOD-iMODb-inext-nth*:
$a \le c \Longrightarrow [r,\ \mathit{mod}\ m,\ c] \rightarrow a = [r,\ \mathit{mod}\ m] \rightarrow a$
⟨*proof*⟩

lemma *iMOD-iMODb-inext*:
$a < r + m * c \Longrightarrow inext\ a\ [r,\ mod\ m,\ c] = inext\ a\ [r,\ mod\ m]$
⟨*proof*⟩

lemma *iMOD-inext-nth-Suc-diff*:
$([r,\ mod\ m] \to (Suc\ n)) - ([r,\ mod\ m] \to n) = m$
⟨*proof*⟩

lemma *iMOD-inext-nth-diff*:
$([r,\ mod\ m] \to a) - ([r,\ mod\ m] \to b) = (a - b) * m$
⟨*proof*⟩

lemma *iMODb-inext-nth-diff*: $[\![\ a \le c;\ b \le c\ ]\!] \Longrightarrow$
$([r,\ mod\ m,\ c] \to a) - ([r,\ mod\ m,\ c] \to b) = (a - b) * m$
⟨*proof*⟩

1.8 Induction with intervals

lemma *iFROM-induct*:
$[\![\ P\ n;\ \bigwedge k.\ [\![\ k \in [n\ldots];\ P\ k\ ]\!] \Longrightarrow P\ (Suc\ k);\ a \in [n\ldots]\ ]\!] \Longrightarrow P\ a$
⟨*proof*⟩

lemma *iIN-induct*:
$[\![\ P\ n;\ \bigwedge k.\ [\![\ k \in [n\ldots,d];\ k \ne n + d;\ P\ k\ ]\!] \Longrightarrow P\ (Suc\ k);\ a \in [n\ldots,d]\ ]\!] \Longrightarrow P\ a$
⟨*proof*⟩

lemma *iTILL-induct*:
$[\![\ P\ 0;\ \bigwedge k.\ [\![\ k \in [\ldots n];\ k \ne n;\ P\ k\ ]\!] \Longrightarrow P\ (Suc\ k);\ a \in [\ldots n]\ ]\!] \Longrightarrow P\ a$
⟨*proof*⟩

lemma *iMOD-induct*:
$[\![\ P\ r;\ \bigwedge k.\ [\![\ k \in [r,\ mod\ m];\ P\ k\ ]\!] \Longrightarrow P\ (k + m);\ a \in [r,\ mod\ m]\ ]\!] \Longrightarrow P\ a$
⟨*proof*⟩

lemma *iMODb-induct*:
$[\![\ P\ r;\ \bigwedge k.\ [\![\ k \in [r,\ mod\ m,\ c];\ k \ne r + m * c;\ P\ k\ ]\!] \Longrightarrow P\ (k + m);\ a \in [r,\ mod\ m,\ c]\ ]\!] \Longrightarrow P\ a$
⟨*proof*⟩

lemma *iIN-rev-induct*:
$[\![\ P\ (n + d);\ \bigwedge k.\ [\![\ k \in [n\ldots,d];\ k \ne n;\ P\ k\ ]\!] \Longrightarrow P\ (k - Suc\ 0);\ a \in [n\ldots,d]\ ]\!] \Longrightarrow P\ a$
⟨*proof*⟩

lemma *iTILL-rev-induct*:
$[\![\ P\ n;\ \bigwedge k.\ [\![\ k \in [\ldots n];\ 0 < k;\ P\ k\ ]\!] \Longrightarrow P\ (k - Suc\ 0);\ a \in [\ldots n]\ ]\!] \Longrightarrow P\ a$
⟨*proof*⟩

lemma *iMODb-rev-induct*:
$[\![\, P\ (r + m * c);\ \bigwedge k.\ [\![\, k \in [r,\ mod\ m,\ c];\ k \neq r;\ P\ k \,]\!] \Longrightarrow P\ (k - m);\ a \in [r,\ mod\ m,\ c] \,]\!] \Longrightarrow P\ a$
⟨*proof*⟩

end

2 Arithmetic operators on natural intervals

theory *IL-IntervalOperators*
imports *IL-Interval*
begin

2.1 Arithmetic operations with intervals

2.1.1 Addition of and multiplication by constants

definition *iT-Plus* :: $iT \Rightarrow Time \Rightarrow iT$ (**infixl** ‹⊕› *55*)
where $I \oplus k \equiv (\lambda n.(n + k))$ ‘ I

definition *iT-Mult* :: $iT \Rightarrow Time \Rightarrow iT$ (**infixl** ‹⊗› *55*)
where *iT-Mult-def* : $I \otimes k \equiv (\lambda n.(n * k))$ ‘ I

lemma *iT-Plus-image-conv*: $I \oplus k = (\lambda n.(n + k))$ ‘ I
⟨*proof*⟩

lemma *iT-Mult-image-conv*: $I \otimes k = (\lambda n.(n * k))$ ‘ I
⟨*proof*⟩

lemma *iT-Plus-empty*: $\{\} \oplus k = \{\}$
⟨*proof*⟩

lemma *iT-Mult-empty*: $\{\} \otimes k = \{\}$
⟨*proof*⟩

lemma *iT-Plus-not-empty*: $I \neq \{\} \Longrightarrow I \oplus k \neq \{\}$
⟨*proof*⟩

lemma *iT-Mult-not-empty*: $I \neq \{\} \Longrightarrow I \otimes k \neq \{\}$
⟨*proof*⟩

lemma *iT-Plus-empty-iff*: $(I \oplus k = \{\}) = (I = \{\})$
⟨*proof*⟩

lemma *iT-Mult-empty-iff*: $(I \otimes k = \{\}) = (I = \{\})$
⟨*proof*⟩

lemma *iT-Plus-mono*: $A \subseteq B \Longrightarrow A \oplus k \subseteq B \oplus k$
⟨*proof*⟩

lemma *iT-Mult-mono*: $A \subseteq B \Longrightarrow A \otimes k \subseteq B \otimes k$
⟨*proof*⟩

lemma *iT-Mult-0*: $I \neq \{\} \Longrightarrow I \otimes 0 = [\ldots 0]$
⟨*proof*⟩

corollary *iT-Mult-0-if*: $I \otimes 0 = (\mathit{if}\ I = \{\}\ \mathit{then}\ \{\}\ \mathit{else}\ [\ldots 0])$
⟨*proof*⟩

lemma *iT-Plus-mem-iff*: $x \in (I \oplus k) = (k \leq x \wedge (x - k) \in I)$
⟨*proof*⟩

lemma *iT-Plus-mem-iff2*: $x + k \in (I \oplus k) = (x \in I)$
⟨*proof*⟩

lemma *iT-Mult-mem-iff-0*: $x \in (I \otimes 0) = (I \neq \{\} \wedge x = 0)$
⟨*proof*⟩

lemma *iT-Mult-mem-iff*:
$0 < k \Longrightarrow x \in (I \otimes k) = (x\ \mathit{mod}\ k = 0 \wedge x\ \mathit{div}\ k \in I)$
⟨*proof*⟩

lemma *iT-Mult-mem-iff2*: $0 < k \Longrightarrow x * k \in (I \otimes k) = (x \in I)$
⟨*proof*⟩

lemma *iT-Plus-singleton*: $\{a\} \oplus k = \{a + k\}$
⟨*proof*⟩

lemma *iT-Mult-singleton*: $\{a\} \otimes k = \{a * k\}$
⟨*proof*⟩

lemma *iT-Plus-Un*: $(A \cup B) \oplus k = (A \oplus k) \cup (B \oplus k)$
⟨*proof*⟩

lemma *iT-Mult-Un*: $(A \cup B) \otimes k = (A \otimes k) \cup (B \otimes k)$
⟨*proof*⟩

lemma *iT-Plus-Int*: $(A \cap B) \oplus k = (A \oplus k) \cap (B \oplus k)$
⟨*proof*⟩

lemma *iT-Mult-Int*: $0 < k \Longrightarrow (A \cap B) \otimes k = (A \otimes k) \cap (B \otimes k)$
⟨*proof*⟩

lemma *iT-Plus-image*: f ‘ $I \oplus n = (\lambda x.\ f\ x + n)$ ‘ I
⟨*proof*⟩

lemma *iT-Mult-image*: f ‘ $I \otimes n = (\lambda x.\ f\ x * n)$ ‘ I
⟨*proof*⟩

lemma *iT-Plus-commute*: $I \oplus a \oplus b = I \oplus b \oplus a$
⟨*proof*⟩

lemma *iT-Mult-commute*: $I \otimes a \otimes b = I \otimes b \otimes a$
⟨*proof*⟩

lemma *iT-Plus-assoc*:$I \oplus a \oplus b = I \oplus (a + b)$
⟨*proof*⟩

lemma *iT-Mult-assoc*:$I \otimes a \otimes b = I \otimes (a * b)$
⟨*proof*⟩

lemma *iT-Plus-Mult-distrib*: $I \oplus n \otimes m = I \otimes m \oplus n * m$
⟨*proof*⟩⟨*proof*⟩⟨*proof*⟩⟨*proof*⟩⟨*proof*⟩⟨*proof*⟩
lemma *iT-Plus-finite-iff*: *finite* $(I \oplus k)$ = *finite* I
⟨*proof*⟩

lemma *iT-Mult-0-finite*: *finite* $(I \otimes 0)$
⟨*proof*⟩

lemma *iT-Mult-finite-iff*: $0 < k \Longrightarrow$ *finite* $(I \otimes k)$ = *finite* I
⟨*proof*⟩

lemma *iT-Plus-Min*: $I \neq \{\} \Longrightarrow iMin\ (I \oplus k) = iMin\ I + k$
⟨*proof*⟩

lemma *iT-Mult-Min*: $I \neq \{\} \Longrightarrow iMin\ (I \otimes k) = iMin\ I * k$
⟨*proof*⟩

lemma *iT-Plus-Max*: $[\![$ *finite* I; $I \neq \{\}$ $]\!] \Longrightarrow Max\ (I \oplus k) = Max\ I + k$
⟨*proof*⟩

lemma *iT-Mult-Max*: $[\![$ *finite* I; $I \neq \{\}$ $]\!] \Longrightarrow Max\ (I \otimes k) = Max\ I * k$
⟨*proof*⟩

corollary
iMOD-mult-0: $[r,\ mod\ m] \otimes 0 = [\ldots 0]$ **and**
iMODb-mult-0: $[r,\ mod\ m,\ c] \otimes 0 = [\ldots 0]$ **and**
iFROM-mult-0: $[n\ldots] \otimes 0 = [\ldots 0]$ **and**
iIN-mult-0: $[n\ldots,d] \otimes 0 = [\ldots 0]$ **and**
iTILL-mult-0: $[\ldots n] \otimes 0 = [\ldots 0]$
⟨*proof*⟩

lemmas *iT-mult-0* =
iTILL-mult-0
iFROM-mult-0
iIN-mult-0
iMOD-mult-0
iMODb-mult-0

lemma *iT-Plus-0*: $I \oplus 0 = I$
⟨*proof*⟩

lemma *iT-Mult-1*: $I \otimes Suc\ 0 = I$
⟨*proof*⟩

corollary
iFROM-add-Min: $iMin\ ([n\ldots] \oplus k) = n + k$ **and**
iIN-add-Min: $iMin\ ([n\ldots,d] \oplus k) = n + k$ **and**
iTILL-add-Min: $iMin\ ([\ldots n] \oplus k) = k$ **and**
iMOD-add-Min: $iMin\ ([r,\ mod\ m] \oplus k) = r + k$ **and**
iMODb-add-Min: $iMin\ ([r,\ mod\ m,\ c] \oplus k) = r + k$
⟨*proof*⟩

corollary
iFROM-mult-Min: $iMin\ ([n\ldots] \otimes k) = n * k$ **and**
iIN-mult-Min: $iMin\ ([n\ldots,d] \otimes k) = n * k$ **and**
iTILL-mult-Min: $iMin\ ([\ldots n] \otimes k) = 0$ **and**
iMOD-mult-Min: $iMin\ ([r,\ mod\ m] \otimes k) = r * k$ **and**
iMODb-mult-Min: $iMin\ ([r,\ mod\ m,\ c] \otimes k) = r * k$
⟨*proof*⟩

lemmas *iT-add-Min* =
iIN-add-Min
iTILL-add-Min
iFROM-add-Min
iMOD-add-Min
iMODb-add-Min

lemmas *iT-mult-Min* =
iIN-mult-Min
iTILL-mult-Min
iFROM-mult-Min
iMOD-mult-Min
iMODb-mult-Min

lemma *iFROM-add*: $[n\ldots] \oplus k = [n+k\ldots]$
⟨*proof*⟩

lemma *iIN-add*: $[n\ldots,d] \oplus k = [n+k\ldots,d]$

⟨*proof*⟩

lemma *iTILL-add*: $[\ldots i] \oplus k = [k\ldots,i]$
⟨*proof*⟩

lemma *iMOD-add*: $[r, \; mod \; m] \oplus k = [r + k, \; mod \; m]$
⟨*proof*⟩

lemma *iMODb-add*: $[r, \; mod \; m, \; c] \oplus k = [r + k, \; mod \; m, \; c]$
⟨*proof*⟩

lemmas *iT-add* =
iMOD-add
iMODb-add
iFROM-add
iIN-add
iTILL-add
iT-Plus-singleton

lemma *iFROM-mult*: $[n\ldots] \otimes k = [\; n * k, \; mod \; k \;]$
⟨*proof*⟩

lemma *iIN-mult*: $[n\ldots,d] \otimes k = [\; n * k, \; mod \; k, \; d \;]$
⟨*proof*⟩

lemma *iTILL-mult*: $[\ldots n] \otimes k = [\; 0, \; mod \; k, \; n \;]$
⟨*proof*⟩

lemma *iMOD-mult*: $[r, \; mod \; m \;] \otimes k = [\; r * k, \; mod \; m * k \;]$
⟨*proof*⟩

lemma *iMODb-mult*:
$[r, \; mod \; m, \; c \;] \otimes k = [\; r * k, \; mod \; m * k, \; c \;]$
⟨*proof*⟩

lemmas *iT-mult* =
iTILL-mult
iFROM-mult
iIN-mult
iMOD-mult
iMODb-mult
iT-Mult-singleton

2.1.2 Some conversions between intervals using constant addition and multiplication

lemma *iFROM-conv*: $[n\ldots] = UNIV \oplus n$
⟨*proof*⟩

lemma *iIN-conv*: $[n\ldots,d] = [\ldots d] \oplus n$
⟨*proof*⟩

lemma *iMOD-conv*: $[r,\ mod\ m] = [0\ldots] \otimes m \oplus r$
⟨*proof*⟩

lemma *iMODb-conv*: $[r,\ mod\ m,\ c] = [\ldots c] \otimes m \oplus r$
⟨*proof*⟩

Some examples showing the utility of iMODb_conv

lemma $[12,\ mod\ 10,\ 4] = \{12,\ 22,\ 32,\ 42,\ 52\}$
⟨*proof*⟩

lemma $[12,\ mod\ 10,\ 4] = \{12,\ 22,\ 32,\ 42,\ 52\}$
⟨*proof*⟩

lemma $[12,\ mod\ 10,\ 4] = \{12,\ 22,\ 32,\ 42,\ 52\}$
⟨*proof*⟩

lemma $[r,\ mod\ m,\ 4] = \{r,\ r+m,\ r+2*m,\ r+3*m,\ r+4*m\}$
⟨*proof*⟩

lemma $[2,\ mod\ 10,\ 4] = \{2,\ 12,\ 22,\ 32,\ 42\}$
⟨*proof*⟩

2.1.3 Subtraction of constants

definition *iT-Plus-neg* :: $iT \Rightarrow Time \Rightarrow iT$ (**infixl** ‹⊕−› *55*) **where**
  $I \oplus- k \equiv \{x.\ x + k \in I\}$

lemma *iT-Plus-neg-mem-iff*: $(x \in I \oplus- k) = (x + k \in I)$
⟨*proof*⟩

lemma *iT-Plus-neg-mem-iff2*: $k \leq x \Longrightarrow (x - k \in I \oplus- k) = (x \in I)$
⟨*proof*⟩

lemma *iT-Plus-neg-image-conv*: $I \oplus- k = (\lambda n.(n - k))$ ‘ $(I \downarrow\geq k)$
⟨*proof*⟩

lemma *iT-Plus-neg-cut-eq*: $t \leq k \Longrightarrow (I \downarrow\geq t) \oplus- k = I \oplus- k$
⟨*proof*⟩

lemma *iT-Plus-neg-mono*: $A \subseteq B \Longrightarrow A \oplus- k \subseteq B \oplus- k$
⟨*proof*⟩

lemma *iT-Plus-neg-empty*: $\{\} \oplus- k = \{\}$
⟨*proof*⟩
lemma *iT-Plus-neg-Max-less-empty*:

$[\![$ *finite I*; *Max I* $< k$ $]\!] \Longrightarrow I \oplus- k = \{\}$
⟨*proof*⟩

lemma *iT-Plus-neg-not-empty-iff*: $(I \oplus- k \neq \{\}) = (\exists x \in I.\ k \leq x)$
⟨*proof*⟩

lemma *iT-Plus-neg-empty-iff*:
$(I \oplus- k = \{\}) = (I = \{\} \vee (\textit{finite } I \wedge \textit{Max } I < k))$
⟨*proof*⟩

lemma *iT-Plus-neg-assoc*: $(I \oplus- a) \oplus- b = I \oplus- (a + b)$
⟨*proof*⟩

lemma *iT-Plus-neg-commute*: $I \oplus- a \oplus- b = I \oplus- b \oplus- a$
⟨*proof*⟩

lemma *iT-Plus-neg-0*: $I \oplus- 0 = I$
⟨*proof*⟩

lemma *iT-Plus-Plus-neg-assoc*: $b \leq a \Longrightarrow I \oplus a \oplus- b = I \oplus (a - b)$
⟨*proof*⟩

lemma *iT-Plus-Plus-neg-assoc2*: $a \leq b \Longrightarrow I \oplus a \oplus- b = I \oplus- (b - a)$
⟨*proof*⟩

lemma *iT-Plus-neg-Plus-le-cut-eq*:
$a \leq b \Longrightarrow (I \oplus- a) \oplus b = (I \downarrow\geq a) \oplus (b - a)$
⟨*proof*⟩

corollary *iT-Plus-neg-Plus-le-Min-eq*:
$[\![a \leq b; a \leq \textit{iMin } I]\!] \Longrightarrow (I \oplus- a) \oplus b = I \oplus (b - a)$
⟨*proof*⟩

lemma *iT-Plus-neg-Plus-ge-cut-eq*:
$b \leq a \Longrightarrow (I \oplus- a) \oplus b = (I \downarrow\geq a) \oplus- (a - b)$
⟨*proof*⟩

corollary *iT-Plus-neg-Plus-ge-Min-eq*:
$[\![b \leq a; a \leq \textit{iMin } I]\!] \Longrightarrow (I \oplus- a) \oplus b = I \oplus- (a - b)$
⟨*proof*⟩

lemma *iT-Plus-neg-Mult-distrib*:
$0 < m \Longrightarrow I \oplus- n \otimes m = I \otimes m \oplus- n * m$
⟨*proof*⟩

lemma *iT-Plus-neg-Plus-le-inverse*: $k \leq \textit{iMin } I \Longrightarrow I \oplus- k \oplus k = I$
⟨*proof*⟩

lemma *iT-Plus-neg-Plus-inverse*: $I \oplus- k \oplus k = I \downarrow\geq k$

⟨*proof*⟩

lemma *iT-Plus-Plus-neg-inverse*: $I \oplus k \oplus- k = I$
⟨*proof*⟩

lemma *iT-Plus-neg-Un*: $(A \cup B) \oplus- k = (A \oplus- k) \cup (B \oplus- k)$
⟨*proof*⟩

lemma *iT-Plus-neg-Int*: $(A \cap B) \oplus- k = (A \oplus- k) \cap (B \oplus- k)$
⟨*proof*⟩

lemma *iT-Plus-neg-Max-singleton*: $\llbracket$ *finite* I; $I \neq \{\}$ $\rrbracket \Longrightarrow I \oplus- Max\ I= \{0\}$
⟨*proof*⟩

lemma *iT-Plus-neg-singleton*: $\{a\} \oplus- k = ($*if* $k \le a$ *then* $\{a - k\}$ *else* $\{\})$
⟨*proof*⟩

corollary *iT-Plus-neg-singleton1*: $k \le a \Longrightarrow \{a\} \oplus- k = \{a-k\}$
⟨*proof*⟩

corollary *iT-Plus-neg-singleton2*: $a < k \Longrightarrow \{a\} \oplus- k= \{\}$
⟨*proof*⟩

lemma *iT-Plus-neg-finite-iff*: *finite* $(I \oplus- k)$ = *finite* I
⟨*proof*⟩

lemma *iT-Plus-neg-Min*:
$I \oplus- k \neq \{\} \Longrightarrow iMin\ (I \oplus- k) = iMin\ (I \downarrow\geq k) - k$
⟨*proof*⟩

lemma *iT-Plus-neg-Max*:
$\llbracket$ *finite* I; $I \oplus- k \neq \{\}$ $\rrbracket \Longrightarrow Max\ (I \oplus- k) = Max\ I - k$
⟨*proof*⟩

Subtractions of constants from intervals

lemma *iFROM-add-neg*: $[n\ldots] \oplus- k = [n - k\ldots]$
⟨*proof*⟩

lemma *iTILL-add-neg*: $[\ldots n] \oplus- k = ($*if* $k \le n$ *then* $[\ldots n - k]$ *else* $\{\})$
⟨*proof*⟩
lemma *iTILL-add-neg1*: $k \le n \Longrightarrow [\ldots n] \oplus- k = [\ldots n-k]$
⟨*proof*⟩
lemma *iTILL-add-neg2*: $n < k \Longrightarrow [\ldots n] \oplus- k = \{\}$
⟨*proof*⟩

lemma *iIN-add-neg*:
$[n\ldots,d] \oplus- k = ($
if $k \le n$ *then* $[n - k\ldots,d]$

else if $k \le n + d$ *then* $[\ldots n + d - k]$ *else* $\{\})$
⟨*proof*⟩

lemma *iIN-add-neg1*: $k \le n \Longrightarrow [n\ldots,d] \oplus- k = [n - k\ldots,d]$
⟨*proof*⟩

lemma *iIN-add-neg2*: $[\![n \le k; k \le n + d]\!] \Longrightarrow [n\ldots,d] \oplus- k = [\ldots n + d - k]$
⟨*proof*⟩

lemma *iIN-add-neg3*: $n + d < k \Longrightarrow [n\ldots,d] \oplus- k = \{\}$
⟨*proof*⟩

lemma *iMOD-0-add-neg*: $[r, \mathit{mod}\ 0] \oplus- k = \{r\} \oplus- k$
⟨*proof*⟩

lemma *iMOD-gr0-add-neg*:
$0 < m \Longrightarrow$
$[r, \mathit{mod}\ m] \oplus- k = ($
if $k \le r$ *then* $[r - k, \mathit{mod}\ m]$
else $[(m + r \mathit{\ mod\ } m - k \mathit{\ mod\ } m) \mathit{\ mod\ } m, \mathit{mod}\ m])$
⟨*proof*⟩

lemma *iMOD-add-neg*:
$[r, \mathit{mod}\ m] \oplus- k = ($
if $k \le r$ *then* $[r - k, \mathit{mod}\ m]$
else if $0 < m$ *then* $[(m + r \mathit{\ mod\ } m - k \mathit{\ mod\ } m) \mathit{\ mod\ } m, \mathit{mod}\ m]$ *else* $\{\})$
⟨*proof*⟩

corollary *iMOD-add-neg1*:
$k \le r \Longrightarrow [r, \mathit{mod}\ m] \oplus- k = [r - k, \mathit{mod}\ m]$
⟨*proof*⟩

lemma *iMOD-add-neg2*:
$[\![0 < m; r < k]\!] \Longrightarrow [r, \mathit{mod}\ m] \oplus- k = [(m + r \mathit{\ mod\ } m - k \mathit{\ mod\ } m) \mathit{\ mod\ } m,$
$\mathit{mod}\ m]$
⟨*proof*⟩

lemma *iMODb-mod-0-add-neg*: $[r, \mathit{mod}\ 0, c] \oplus- k = \{r\} \oplus- k$
⟨*proof*⟩

lemma *iMODb-add-neg*:
$[r, \mathit{mod}\ m, c] \oplus- k = ($
if $k \le r$ *then* $[r - k, \mathit{mod}\ m, c]$
else

$\mathit{if}\ k \le r + m * c\ \mathit{then}$
$[(m + r\ \mathit{mod}\ m - k\ \mathit{mod}\ m)\ \mathit{mod}\ m,\ \mathit{mod}\ m,\ (r + m * c - k)\ \mathit{div}\ m]$
$\mathit{else}\ \{\})$
⟨*proof*⟩

lemma *iMODb-add-neg′*:
$[r,\ \mathit{mod}\ m,\ c] \oplus\!- k = ($
$\mathit{if}\ k \le r\ \mathit{then}\ [r - k,\ \mathit{mod}\ m,\ c]$
$\mathit{else}\ \mathit{if}\ k \le r + m * c\ \mathit{then}$
$\mathit{if}\ k\ \mathit{mod}\ m \le r\ \mathit{mod}\ m$
$\mathit{then}\ [\ r\ \mathit{mod}\ m - k\ \mathit{mod}\ m,\ \mathit{mod}\ m,\ c + r\ \mathit{div}\ m - k\ \mathit{div}\ m]$
$\mathit{else}\ [\ m + r\ \mathit{mod}\ m - k\ \mathit{mod}\ m,\ \mathit{mod}\ m,\ c + r\ \mathit{div}\ m - \mathit{Suc}\ (k\ \mathit{div}\ m)\ ]$
$\mathit{else}\ \{\})$
⟨*proof*⟩

corollary *iMODb-add-neg1*:
$k \le r \Longrightarrow [r,\ \mathit{mod}\ m,\ c] \oplus\!- k = [r - k,\ \mathit{mod}\ m,\ c]$
⟨*proof*⟩

corollary *iMODb-add-neg2*:
$[\![\ r < k;\ k \le r + m * c\ ]\!] \Longrightarrow$
$[r,\ \mathit{mod}\ m,\ c] \oplus\!- k =$
$[(m + r\ \mathit{mod}\ m - k\ \mathit{mod}\ m)\ \mathit{mod}\ m,\ \mathit{mod}\ m,\ (r + m * c - k)\ \mathit{div}\ m]$
⟨*proof*⟩

corollary *iMODb-add-neg2-mod-le*:
$[\![\ r < k;\ k \le r + m * c;\ k\ \mathit{mod}\ m \le r\ \mathit{mod}\ m\ ]\!] \Longrightarrow$
$[r,\ \mathit{mod}\ m,\ c] \oplus\!- k =$
$[\ r\ \mathit{mod}\ m - k\ \mathit{mod}\ m,\ \mathit{mod}\ m,\ c + r\ \mathit{div}\ m - k\ \mathit{div}\ m]$
⟨*proof*⟩

corollary *iMODb-add-neg2-mod-less*:
$[\![\ r < k;\ k \le r + m * c;\ r\ \mathit{mod}\ m < k\ \mathit{mod}\ m]\!] \Longrightarrow$
$[r,\ \mathit{mod}\ m,\ c] \oplus\!- k =$
$[\ m + r\ \mathit{mod}\ m - k\ \mathit{mod}\ m,\ \mathit{mod}\ m,\ c + r\ \mathit{div}\ m - \mathit{Suc}\ (k\ \mathit{div}\ m)\ ]$
⟨*proof*⟩

lemma *iMODb-add-neg3*: $r + m * c < k \Longrightarrow [r,\ \mathit{mod}\ m,\ c] \oplus\!- k = \{\}$
⟨*proof*⟩

lemmas *iT-add-neg* =
iFROM-add-neg
iIN-add-neg
iTILL-add-neg
iMOD-add-neg
iMODb-add-neg
iT-Plus-neg-singleton

2.1.4 Subtraction of intervals from constants

definition *iT-Minus* :: *Time* ⇒ *iT* ⇒ *iT* (**infixl** ‹⊖› *55*)
 where $k \ominus I \equiv \{x.\ x \le k \wedge (k - x) \in I\}$

lemma *iT-Minus-mem-iff*: $(x \in k \ominus I) = (x \le k \wedge k - x \in I)$
⟨*proof*⟩

lemma *iT-Minus-mono*: $A \subseteq B \Longrightarrow k \ominus A \subseteq k \ominus B$
⟨*proof*⟩

lemma *iT-Minus-image-conv*: $k \ominus I = (\lambda x.\ k - x)$ ‘ $(I \downarrow\le k)$
⟨*proof*⟩

lemma *iT-Minus-cut-eq*: $k \le t \Longrightarrow k \ominus (I \downarrow\le t) = k \ominus I$
⟨*proof*⟩

lemma *iT-Minus-Minus-cut-eq*: $k \ominus (k \ominus (I \downarrow\le k)) = I \downarrow\le k$
⟨*proof*⟩

lemma $10 \ominus [\ldots 3] = [7\ldots,3]$
⟨*proof*⟩

lemma *iT-Minus-empty*: $k \ominus \{\} = \{\}$
⟨*proof*⟩

lemma *iT-Minus-0*: $k \ominus \{0\} = \{k\}$
⟨*proof*⟩

lemma *iT-Minus-bound*: $x \in k \ominus I \Longrightarrow x \le k$
⟨*proof*⟩

lemma *iT-Minus-finite*: *finite* $(k \ominus I)$
⟨*proof*⟩

lemma *iT-Minus-less-Min-empty*: $k <$ *iMin* $I \Longrightarrow k \ominus I = \{\}$
⟨*proof*⟩

lemma *iT-Minus-Min-singleton*: $I \ne \{\} \Longrightarrow$ (*iMin* $I) \ominus I = \{0\}$
⟨*proof*⟩

lemma *iT-Minus-empty-iff*: $(k \ominus I = \{\}) = (I = \{\} \vee k <$ *iMin* $I)$
⟨*proof*⟩

lemma *iT-Minus-imirror-conv*:
 $k \ominus I =$ *imirror* $(I \downarrow\le k) \oplus k \oplus-$ (*iMin* I + *Max* $(I \downarrow\le k))$
⟨*proof*⟩

lemma *iT-Minus-imirror-conv'*:
$k \ominus I = imirror\ (I \downarrow\leq k) \oplus k \oplus-\ (iMin\ (I \downarrow\leq k) + Max\ (I \downarrow\leq k))$
⟨*proof*⟩

lemma *iT-Minus-Max*:
$[\![\ I \neq \{\};\ iMin\ I \leq k\ ]\!] \Longrightarrow Max\ (k \ominus I) = k - (iMin\ I)$
⟨*proof*⟩

lemma *iT-Minus-Min*:
$[\![\ I \neq \{\};\ iMin\ I \leq k\ ]\!] \Longrightarrow iMin\ (k \ominus I) = k - (Max\ (I \downarrow\leq k))$
⟨*proof*⟩

lemma *iT-Minus-Minus-eq*: $[\![\ finite\ I;\ Max\ I \leq k\ ]\!] \Longrightarrow k \ominus (k \ominus I) = I$
⟨*proof*⟩

lemma *iT-Minus-Minus-eq2*: $I \subseteq [\ldots k] \Longrightarrow k \ominus (k \ominus I) = I$
⟨*proof*⟩

lemma *iT-Minus-Minus*: $a \ominus (b \ominus I) = (I \downarrow\leq b) \oplus a \oplus-\ b$
⟨*proof*⟩

lemma *iT-Minus-Plus-empty*: $k < n \Longrightarrow k \ominus (I \oplus n) = \{\}$
⟨*proof*⟩
lemma *iT-Minus-Plus-commute*: $n \leq k \Longrightarrow k \ominus (I \oplus n) = (k - n) \ominus I$
⟨*proof*⟩

lemma *iT-Minus-Plus-cut-assoc*: $(k \ominus I) \oplus n = (k + n) \ominus (I \downarrow\leq k)$
⟨*proof*⟩

lemma *iT-Minus-Plus-assoc*:
$[\![\ finite\ I;\ Max\ I \leq k\ ]\!] \Longrightarrow (k \ominus I) \oplus n = (k + n) \ominus I$
⟨*proof*⟩
lemma *iT-Minus-Plus-assoc2*:
$I \subseteq [\ldots k] \Longrightarrow (k \ominus I) \oplus n = (k + n) \ominus I$
⟨*proof*⟩

lemma *iT-Minus-Un*: $k \ominus (A \cup B) = (k \ominus A) \cup (k \ominus B)$
⟨*proof*⟩

lemma *iT-Minus-Int*: $k \ominus (A \cap B) = (k \ominus A) \cap (k \ominus B)$
⟨*proof*⟩

lemma *iT-Minus-singleton*: $k \ominus \{a\} = (if\ a \leq k\ then\ \{k - a\}\ else\ \{\})$
⟨*proof*⟩
corollary *iT-Minus-singleton1*: $a \leq k \Longrightarrow k \ominus \{a\} = \{k-a\}$
⟨*proof*⟩
corollary *iT-Minus-singleton2*: $k < a \Longrightarrow k \ominus \{a\} = \{\}$

⟨*proof*⟩

lemma *iMOD-sub*:
$k \ominus [r,\ mod\ m] =$
$(if\ r \le k\ then\ [(k - r)\ mod\ m,\ mod\ m,\ (k - r)\ div\ m]\ else\ \{\})$
⟨*proof*⟩

corollary *iMOD-sub1*:
$r \le k \Longrightarrow k \ominus [r,\ mod\ m] = [(k - r)\ mod\ m,\ mod\ m,\ (k - r)\ div\ m]$
⟨*proof*⟩

corollary *iMOD-sub2*: $k < r \Longrightarrow k \ominus [r,\ mod\ m] = \{\}$
⟨*proof*⟩

lemma *iTILL-sub*: $k \ominus [\ldots n] = (if\ n \le k\ then\ [k - n\ldots,n]\ else\ [\ldots k])$
⟨*proof*⟩

corollary *iTILL-sub1*: $n \le k \Longrightarrow k \ominus [\ldots n] = [k - n\ldots,n]$
⟨*proof*⟩

corollary *iTILL-sub2*: $k \le n \Longrightarrow k \ominus [\ldots n] = [\ldots k]$
⟨*proof*⟩

lemma *iMODb-sub*:
$k \ominus [r,\ mod\ m,\ c] = ($
$if\ r + m * c \le k\ then\ [\ k - r - m * c,\ mod\ m,\ c]\ else$
$if\ r \le k\ then\ [\ (k - r)\ mod\ m,\ mod\ m,\ (k - r)\ div\ m]\ else\ \{\})$
⟨*proof*⟩

corollary *iMODb-sub1*:
$[\![\ r \le k;\ k \le r + m * c\ ]\!] \Longrightarrow$
$k \ominus [r,\ mod\ m,\ c] = [(k - r)\ mod\ m,\ mod\ m,\ (k - r)\ div\ m]$
⟨*proof*⟩

corollary *iMODb-sub2*: $k < r \Longrightarrow k \ominus [r,\ mod\ m,\ c] = \{\}$
⟨*proof*⟩

corollary *iMODb-sub3*:
$r + m * c \le k \Longrightarrow k \ominus [r,\ mod\ m,\ c] = [\ k - r - m * c,\ mod\ m,\ c]$
⟨*proof*⟩

lemma *iFROM-sub*: $k \ominus [n\ldots] = (if\ n \le k\ then\ [\ldots k - n]\ else\ \{\})$
⟨*proof*⟩

corollary *iFROM-sub1*: $n \le k \Longrightarrow k \ominus [n\ldots] = [\ldots k{-}n]$
⟨*proof*⟩

corollary *iFROM-sub-empty*: $k < n \Longrightarrow k \ominus [n\ldots] = \{\}$
⟨*proof*⟩

lemma *iIN-sub*:
$k \ominus [n\ldots,d] = ($
if $n + d \le k$ *then* $[k - (n + d)\ldots,d]$
else if $n \le k$ *then* $[\ldots k - n]$ *else* $\{\})$
⟨*proof*⟩

lemma *iIN-sub1*: $n + d \le k \Longrightarrow k \ominus [n\ldots,d] = [k - (n + d)\ldots,d]$
⟨*proof*⟩

lemma *iIN-sub2*: $[\![\ n \le k;\ k \le n + d\ ]\!] \Longrightarrow k \ominus [n\ldots,d] = [\ldots k - n]$
⟨*proof*⟩

lemma *iIN-sub3*: $k < n \Longrightarrow k \ominus [n\ldots,d] = \{\}$
⟨*proof*⟩

lemmas *iT-sub* =
iFROM-sub
iIN-sub
iTILL-sub
iMOD-sub
iMODb-sub
iT-Minus-singleton

2.1.5 Division of intervals by constants

Monotonicity and injectivity of artithmetic operators

lemma *iMOD-div-right-strict-mono-on*:
$[\![\ 0 < k;\ k \le m\ ]\!] \Longrightarrow$ *strict-mono-on* $(\lambda x.\ x\ div\ k)\ [r,\ mod\ m]$
⟨*proof*⟩

corollary *iMOD-div-right-inj-on*:
$[\![\ 0 < k;\ k \le m\ ]\!] \Longrightarrow$ *inj-on* $(\lambda x.\ x\ div\ k)\ [r,\ mod\ m]$
⟨*proof*⟩

lemma *iMOD-mult-div-right-inj-on*:
inj-on $(\lambda x.\ x\ div\ (k{::}nat))\ [r,\ mod\ (k * m)]$
⟨*proof*⟩

lemma *iMOD-mult-div-right-inj-on2*:
$m\ mod\ k = 0 \Longrightarrow$ *inj-on* $(\lambda x.\ x\ div\ k)\ [r,\ mod\ m]$
⟨*proof*⟩

lemma *iMODb-div-right-strict-mono-on*:

$[\![0 < k; k \leq m]\!] \Longrightarrow$ *strict-mono-on* $(\lambda x.\ x\ div\ k)$ $[r,\ mod\ m,\ c]$
⟨*proof*⟩

corollary *iMODb-div-right-inj-on*:
$[\![0 < k; k \leq m]\!] \Longrightarrow$ *inj-on* $(\lambda x.\ x\ div\ k)$ $[r,\ mod\ m,\ c]$
⟨*proof*⟩

lemma *iMODb-mult-div-right-inj-on*:
inj-on $(\lambda x.\ x\ div\ (k{::}nat))$ $[r,\ mod\ (k * m),\ c]$
⟨*proof*⟩

corollary *iMODb-mult-div-right-inj-on2*:
$m\ mod\ k = 0 \Longrightarrow$ *inj-on* $(\lambda x.\ x\ div\ k)$ $[r,\ mod\ m,\ c]$
⟨*proof*⟩

definition *iT-Div* :: $iT \Rightarrow Time \Rightarrow iT$ (**infixl** ‹⊘› *55*)
where $I \oslash k \equiv (\lambda n.(n\ div\ k))$ ‘ I

lemma *iT-Div-image-conv*: $I \oslash k = (\lambda n.(n\ div\ k))$ ‘ I
⟨*proof*⟩

lemma *iT-Div-mono*: $A \subseteq B \Longrightarrow A \oslash k \subseteq B \oslash k$
⟨*proof*⟩

lemma *iT-Div-empty*: $\{\} \oslash k = \{\}$
⟨*proof*⟩
lemma *iT-Div-not-empty*: $I \neq \{\} \Longrightarrow I \oslash k \neq \{\}$
⟨*proof*⟩

lemma *iT-Div-empty-iff*: $(I \oslash k = \{\}) = (I = \{\})$
⟨*proof*⟩

lemma *iT-Div-0*: $I \neq \{\} \Longrightarrow I \oslash 0 = [\ldots 0]$
⟨*proof*⟩
corollary *iT-Div-0-if*: $I \oslash 0 = ($*if* $I = \{\}$ *then* $\{\}$ *else* $[\ldots 0])$
⟨*proof*⟩
corollary
iFROM-div-0: $[n\ldots] \oslash 0 = [\ldots 0]$ **and**
iTILL-div-0: $[\ldots n] \oslash 0 = [\ldots 0]$ **and**
iIN-div-0: $[n\ldots,d] \oslash 0 = [\ldots 0]$ **and**
iMOD-div-0: $[r,\ mod\ m] \oslash 0 = [\ldots 0]$ **and**
iMODb-div-0: $[r,\ mod\ m,\ c] \oslash 0 = [\ldots 0]$
⟨*proof*⟩

lemmas *iT-div-0* =
iTILL-div-0
iFROM-div-0

iIN-div-0
iMOD-div-0
iMODb-div-0

lemma *iT-Div-1*: $I \oslash Suc\ 0 = I$
⟨*proof*⟩

lemma *iT-Div-mem-iff-0*: $x \in (I \oslash 0) = (I \neq \{\} \wedge x = 0)$
⟨*proof*⟩

lemma *iT-Div-mem-iff*:
$0 < k \Longrightarrow x \in (I \oslash k) = (\exists y \in I.\ y\ div\ k = x)$
⟨*proof*⟩

lemma *iT-Div-mem-iff2*:
$0 < k \Longrightarrow x\ div\ k \in (I \oslash k) = (\exists y \in I.\ y\ div\ k = x\ div\ k)$
⟨*proof*⟩

lemma *iT-Div-mem-iff-Int*:
$0 < k \Longrightarrow x \in (I \oslash k) = (I \cap [x * k\ldots,k - Suc\ 0] \neq \{\})$
⟨*proof*⟩

lemma *iT-Div-imp-mem*:
$0 < k \Longrightarrow x \in I \Longrightarrow x\ div\ k \in (I \oslash k)$
⟨*proof*⟩

lemma *iT-Div-singleton*: $\{a\} \oslash k = \{a\ div\ k\}$
⟨*proof*⟩

lemma *iT-Div-Un*: $(A \cup B) \oslash k = (A \oslash k) \cup (B \oslash k)$
⟨*proof*⟩

lemma *iT-Div-insert*: $(insert\ n\ I) \oslash k = insert\ (n\ div\ k)\ (I \oslash k)$
⟨*proof*⟩

lemma *not-iT-Div-Int*: $\neg\ (\forall\ k\ A\ B.\ (A \cap B) \oslash k = (A \oslash k) \cap (B \oslash k))$
⟨*proof*⟩

lemma *subset-iT-Div-Int*: $A \subseteq B \Longrightarrow (A \cap B) \oslash k = (A \oslash k) \cap (B \oslash k)$
⟨*proof*⟩

lemma *iFROM-iT-Div-Int*:
$[\![\ 0 < k;\ n \le iMin\ A\ ]\!] \Longrightarrow (A \cap [n\ldots]) \oslash k = (A \oslash k) \cap ([n\ldots] \oslash k)$
⟨*proof*⟩

lemma *iIN-iT-Div-Int*:
$[\![\ 0 < k;\ n \le iMin\ A;\ \forall x{\in}A.\ x\ div\ k \le (n + d)\ div\ k \longrightarrow x \le n + d\ ]\!] \Longrightarrow$

$(A \cap [n\ldots,d]) \oslash k = (A \oslash k) \cap ([n\ldots,d] \oslash k)$
⟨*proof*⟩
corollary *iTILL-iT-Div-Int*:
$[\![\ 0 < k;\ \forall x{\in}A.\ x\ div\ k \leq n\ div\ k \longrightarrow x \leq n\ ]\!] \Longrightarrow$
$(A \cap [\ldots n]) \oslash k = (A \oslash k) \cap ([\ldots n] \oslash k)$
⟨*proof*⟩
lemma *iIN-iT-Div-Int-mod-0*:
$[\![\ 0 < k;\ n\ mod\ k = 0;\ \forall x{\in}A.\ x\ div\ k \leq (n + d)\ div\ k \longrightarrow x \leq n + d\ ]\!] \Longrightarrow$
$(A \cap [n\ldots,d]) \oslash k = (A \oslash k) \cap ([n\ldots,d] \oslash k)$
⟨*proof*⟩

lemma *mod-partition-iT-Div-Int*:
$[\![\ 0 < k;\ 0 < d\ ]\!] \Longrightarrow$
$(A \cap [n * k\ldots,d * k - Suc\ 0]) \oslash k =$
$(A \oslash k) \cap ([n * k\ldots,d * k - Suc\ 0] \oslash k)$
⟨*proof*⟩⟨*proof*⟩
corollary *mod-partition-iT-Div-Int2*:
$[\![\ 0 < k;\ 0 < d;\ n\ mod\ k = 0;\ d\ mod\ k = 0\ ]\!] \Longrightarrow$
$(A \cap [n\ldots,d - Suc\ 0]) \oslash k =$
$(A \oslash k) \cap ([n\ldots,d - Suc\ 0] \oslash k)$
⟨*proof*⟩

corollary *mod-partition-iT-Div-Int-one-segment*:
$0 < k \Longrightarrow$
$(A \cap [n * k\ldots,k - Suc\ 0]) \oslash k = (A \oslash k) \cap ([n * k\ldots,k - Suc\ 0] \oslash k)$
⟨*proof*⟩

corollary *mod-partition-iT-Div-Int-one-segment2*:
$[\![\ 0 < k;\ n\ mod\ k = 0\ ]\!] \Longrightarrow$
$(A \cap [n\ldots,k - Suc\ 0]) \oslash k = (A \oslash k) \cap ([n\ldots,k - Suc\ 0] \oslash k)$
⟨*proof*⟩

lemma *iT-Div-assoc*:$I \oslash a \oslash b = I \oslash (a * b)$
⟨*proof*⟩

lemma *iT-Div-commute*: $I \oslash a \oslash b = I \oslash b \oslash a$
⟨*proof*⟩

lemma *iT-Mult-Div-self*: $0 < k \Longrightarrow I \otimes k \oslash k = I$
⟨*proof*⟩
lemma *iT-Mult-Div*:
$[\![\ 0 < d;\ \ k\ mod\ d = 0\ ]\!] \Longrightarrow I \otimes k \oslash d = I \otimes (k\ div\ d)$
⟨*proof*⟩

lemma *iT-Div-Mult-self*:
$0 < k \Longrightarrow I \oslash k \otimes k = \{y.\ \exists x \in I.\ y = x - x\ mod\ k\}$
⟨*proof*⟩

lemma *iT-Plus-Div-distrib-mod-less*:
$\forall x \in I.\ x\ mod\ m + n\ mod\ m < m \Longrightarrow I \oplus n \oslash m = I \oslash m \oplus n\ div\ m$
⟨*proof*⟩
corollary *iT-Plus-Div-distrib-mod-0*:
$n\ mod\ m = 0 \Longrightarrow I \oplus n \oslash m = I \oslash m \oplus n\ div\ m$
⟨*proof*⟩

lemma *iT-Div-Min*: $I \neq \{\} \Longrightarrow iMin\ (I \oslash k) = iMin\ I\ div\ k$
⟨*proof*⟩

corollary
iFROM-div-Min: $iMin\ ([n\ldots] \oslash k) = n\ div\ k$ **and**
iIN-div-Min: $iMin\ ([n\ldots,d] \oslash k) = n\ div\ k$ **and**
iTILL-div-Min: $iMin\ ([\ldots n] \oslash k) = 0$ **and**
iMOD-div-Min: $iMin\ ([r,\ mod\ m] \oslash k) = r\ div\ k$ **and**
iMODb-div-Min: $iMin\ ([r,\ mod\ m,\ c] \oslash k) = r\ div\ k$
⟨*proof*⟩

lemmas *iT-div-Min* =
iFROM-div-Min
iIN-div-Min
iTILL-div-Min
iMOD-div-Min
iMODb-div-Min

lemma *iT-Div-Max*: $\llbracket\ finite\ I;\ I \neq \{\}\ \rrbracket \Longrightarrow Max\ (I \oslash k) = Max\ I\ div\ k$
⟨*proof*⟩

corollary
iIN-div-Max: $Max\ ([n\ldots,d] \oslash k) = (n + d)\ div\ k$ **and**
iTILL-div-Max: $Max\ ([\ldots n] \oslash k) = n\ div\ k$ **and**
iMODb-div-Max: $Max\ ([r,\ mod\ m,\ c] \oslash k) = (r + m * c)\ div\ k$
⟨*proof*⟩

lemma *iT-Div-0-finite*: $finite\ (I \oslash 0)$
⟨*proof*⟩

lemma *iT-Div-infinite-iff*: $0 < k \Longrightarrow infinite\ (I \oslash k) = infinite\ I$
⟨*proof*⟩
lemma *iT-Div-finite-iff*: $0 < k \Longrightarrow finite\ (I \oslash k) = finite\ I$
⟨*proof*⟩

lemma *iFROM-div*: $0 < k \Longrightarrow [n\ldots] \oslash k = [n\ div\ k\ldots]$
⟨*proof*⟩

lemma *iIN-div*:
$0 < k \Longrightarrow$
$[n\ldots,d] \oslash k = [n\ div\ k\ldots,\ d\ div\ k + (n\ mod\ k + d\ mod\ k)\ div\ k\ ]$

⟨*proof*⟩

corollary *iIN-div-if*:
$0 < k \Longrightarrow [n\ldots,d] \oslash k =$
$[n \; div \; k\ldots, \; d \; div \; k + (if \; n \; mod \; k + d \; mod \; k < k \; then \; 0 \; else \; Suc \; 0)]$
⟨*proof*⟩

corollary *iIN-div-eq1*:
$[\![\; 0 < k; \; n \; mod \; k + d \; mod \; k < k \;]\!] \Longrightarrow$
$[n\ldots,d] \oslash k = [n \; div \; k\ldots,d \; div \; k]$
⟨*proof*⟩

corollary *iIN-div-eq2*:
$[\![\; 0 < k; \; k \le n \; mod \; k + d \; mod \; k \;]\!] \Longrightarrow$
$[n\ldots,d] \oslash k = [n \; div \; k\ldots, \; Suc \; (d \; div \; k)]$
⟨*proof*⟩

corollary *iIN-div-mod-eq-0*:
$[\![\; 0 < k; \; n \; mod \; k = 0 \;]\!] \Longrightarrow [n\ldots,d] \oslash k = [n \; div \; k\ldots,d \; div \; k]$
⟨*proof*⟩

lemma *iTILL-div*:
$0 < k \Longrightarrow [\ldots n] \oslash k = [\ldots n \; div \; k]$
⟨*proof*⟩

lemma *iMOD-div-ge*:
$[\![\; 0 < m; \; m \le k \;]\!] \Longrightarrow [r, \; mod \; m] \oslash k = [r \; div \; k\ldots]$
⟨*proof*⟩
corollary *iMOD-div-self*:
$0 < m \Longrightarrow [r, \; mod \; m] \oslash m = [r \; div \; m\ldots]$
⟨*proof*⟩

lemma *iMOD-div*:
$[\![\; 0 < k; \; m \; mod \; k = 0 \;]\!] \Longrightarrow$
$[r, \; mod \; m] \oslash k = [r \; div \; k, \; mod \; (m \; div \; k) \;]$
⟨*proof*⟩

lemma *iMODb-div-self*:
$0 < m \Longrightarrow [r, \; mod \; m, \; c] \oslash m = [r \; div \; m\ldots,c]$
⟨*proof*⟩

lemma *iMODb-div-ge*:
$[\![\; 0 < m; \; m \le k \;]\!] \Longrightarrow$
$[r, \; mod \; m, \; c] \oslash k = [r \; div \; k\ldots,(r + m * c) \; div \; k - r \; div \; k]$
⟨*proof*⟩

corollary *iMODb-div-ge-if*:
$[\![\; 0 < m; \; m \le k \;]\!] \Longrightarrow$

$[r, mod\ m, c] \oslash k =$
$[r\ div\ k\ldots, m * c\ div\ k + (if\ r\ mod\ k + m * c\ mod\ k < k\ then\ 0\ else\ Suc\ 0)]$
⟨*proof*⟩

lemma *iMODb-div*:
$[\![\ 0 < k; m\ mod\ k = 0\ ]\!] \Longrightarrow$
$[r, mod\ m, c] \oslash k = [r\ div\ k, mod\ (m\ div\ k), c\ ]$
⟨*proof*⟩

lemmas *iT-div* =
iTILL-div
iFROM-div
iIN-div
iMOD-div
iMODb-div
iT-Div-singleton

This lemma is valid for all $k \le m$,i. e., also for k with $m\ mod\ k \ne 0$.

lemma *iMODb-div-unique*:
$[\![\ 0 < k; k \le m; k \le c; [r', mod\ m', c'] = [r, mod\ m, c] \oslash k\ ]\!] \Longrightarrow$
$r' = r\ div\ k \wedge m' = m\ div\ k \wedge c' = c$
⟨*proof*⟩

lemma *iMODb-div-mod-gr0-is-0-not-ex0*:
$[\![\ 0 < k; k < m; 0 < m\ mod\ k; k \le c; r\ mod\ k = 0\ ]\!] \Longrightarrow$
$\neg(\exists r'\ m'\ c'. [r', mod\ m', c'] = [r, mod\ m, c] \oslash k)$
⟨*proof*⟩

lemma *iMODb-div-mod-gr0-not-ex--arith-aux1*:
$[\![\ (0::nat) < k; k < m; 0 < x1\ ]\!] \Longrightarrow$
$x1 * m + x2 - x\ mod\ k + x3 + x\ mod\ k = x1 * m + x2 + x3$
⟨*proof*⟩

lemma *iMODb-div-mod-gr0-not-ex*:
$[\![\ 0 < k; k < m; 0 < m\ mod\ k; k \le c\ ]\!] \Longrightarrow$
$\neg(\exists r'\ m'\ c'. [r', mod\ m', c'] = [r, mod\ m, c] \oslash k)$
⟨*proof*⟩

lemma *iMOD-div-eq-imp-iMODb-div-eq*:
$[\![\ 0 < k; k \le m; [r', mod\ m'] = [r, mod\ m] \oslash k\ ]\!] \Longrightarrow$
$[r', mod\ m', c] = [r, mod\ m, c] \oslash k$
⟨*proof*⟩

lemma *iMOD-div-unique*:
$[\![\ 0 < k; k \le m; [r', mod\ m'] = [r, mod\ m] \oslash k\ ]\!] \Longrightarrow$
$r' = r\ div\ k \wedge m' = m\ div\ k$

$\langle proof \rangle$

lemma *iMOD-div-mod-gr0-not-ex*:
$[\![0 < k; k < m; 0 < m \ mod \ k]\!] \Longrightarrow$
$\neg (\exists r' \ m'. [r', mod \ m'] = [r, mod \ m] \oslash k)$
$\langle proof \rangle$

2.2 Interval cut operators with arithmetic interval operators

lemma
iT-Plus-cut-le2: $(I \oplus k) \downarrow\leq (t + k) = (I \downarrow\leq t) \oplus k$ **and**
iT-Plus-cut-less2: $(I \oplus k) \downarrow< (t + k) = (I \downarrow< t) \oplus k$ **and**
iT-Plus-cut-ge2: $(I \oplus k) \downarrow\geq (t + k) = (I \downarrow\geq t) \oplus k$ **and**
iT-Plus-cut-greater2: $(I \oplus k) \downarrow> (t + k) = (I \downarrow> t) \oplus k$
$\langle proof \rangle$

lemma *iT-Plus-cut-le*:
$(I \oplus k) \downarrow\leq t = (if \ t < k \ then \ \{\} \ else \ I \downarrow\leq (t - k) \oplus k)$
$\langle proof \rangle$

lemma *iT-Plus-cut-less*: $(I \oplus k) \downarrow< t = I \downarrow< (t - k) \oplus k$
$\langle proof \rangle$

lemma *iT-Plus-cut-ge*: $(I \oplus k) \downarrow\geq t = I \downarrow\geq (t - k) \oplus k$
$\langle proof \rangle$

lemma *iT-Plus-cut-greater*:
$(I \oplus k) \downarrow> t = (if \ t < k \ then \ I \oplus k \ else \ I \downarrow> (t - k) \oplus k)$
$\langle proof \rangle$

lemma
iT-Mult-cut-le2: $0 < k \Longrightarrow (I \otimes k) \downarrow\leq (t * k) = (I \downarrow\leq t) \otimes k$ **and**
iT-Mult-cut-less2: $0 < k \Longrightarrow (I \otimes k) \downarrow< (t * k) = (I \downarrow< t) \otimes k$ **and**
iT-Mult-cut-ge2: $0 < k \Longrightarrow (I \otimes k) \downarrow\geq (t * k) = (I \downarrow\geq t) \otimes k$ **and**
iT-Mult-cut-greater2: $0 < k \Longrightarrow (I \otimes k) \downarrow> (t * k) = (I \downarrow> t) \otimes k$
$\langle proof \rangle$

lemma *iT-Mult-cut-le*:
$0 < k \Longrightarrow (I \otimes k) \downarrow\leq t = (I \downarrow\leq (t \ div \ k)) \otimes k$
$\langle proof \rangle$

lemma *iT-Mult-cut-less*:
$0 < k \Longrightarrow (I \otimes k) \downarrow< t =$
$(if \ t \ mod \ k = 0 \ then \ (I \downarrow< (t \ div \ k)) \ else \ I \downarrow< Suc \ (t \ div \ k)) \otimes k$
$\langle proof \rangle$

lemma *iT-Mult-cut-greater*:
$0 < k \Longrightarrow (I \otimes k) \downarrow> t = (I \downarrow> (t \ div \ k)) \otimes k$

⟨*proof*⟩

lemma *iT-Mult-cut-ge*:
$0 < k \Longrightarrow (I \otimes k) \downarrow\geq t =$
$(\textit{if } t \textit{ mod } k = 0 \textit{ then } (I \downarrow\geq (t \textit{ div } k)) \textit{ else } I \downarrow\geq \textit{Suc } (t \textit{ div } k)) \otimes k$
⟨*proof*⟩

lemma *iT-Plus-neg-cut-le2*: $k \leq t \Longrightarrow (I \oplus- k) \downarrow\leq (t - k) = (I \downarrow\leq t) \oplus- k$
⟨*proof*⟩

lemma *iT-Plus-neg-cut-less2*: $(I \oplus- k) \downarrow< (t - k) = (I \downarrow< t) \oplus- k$
⟨*proof*⟩

lemma *iT-Plus-neg-cut-ge2*: $(I \oplus- k) \downarrow\geq (t - k) = (I \downarrow\geq t) \oplus- k$
⟨*proof*⟩

lemma *iT-Plus-neg-cut-greater2*: $k \leq t \Longrightarrow (I \oplus- k) \downarrow> (t - k) = (I \downarrow> t) \oplus- k$
⟨*proof*⟩

lemma *iT-Plus-neg-cut-le*: $(I \oplus- k) \downarrow\leq t = I \downarrow\leq (t + k) \oplus- k$
⟨*proof*⟩

lemma *iT-Plus-neg-cut-less*: $(I \oplus- k) \downarrow< t = I \downarrow< (t + k) \oplus- k$
⟨*proof*⟩

lemma *iT-Plus-neg-cut-ge*: $(I \oplus- k) \downarrow\geq t = I \downarrow\geq (t + k) \oplus- k$
⟨*proof*⟩

lemma *iT-Plus-neg-cut-greater*: $(I \oplus- k) \downarrow> t = I \downarrow> (t + k) \oplus- k$
⟨*proof*⟩

lemma *iT-Minus-cut-le2*: $t \leq k \Longrightarrow (k \ominus I) \downarrow\leq (k - t) = k \ominus (I \downarrow\geq t)$
⟨*proof*⟩

lemma *iT-Minus-cut-less2*: $(k \ominus I) \downarrow< (k - t) = k \ominus (I \downarrow> t)$
⟨*proof*⟩

lemma *iT-Minus-cut-ge2*: $(k \ominus I) \downarrow\geq (k - t) = k \ominus (I \downarrow\leq t)$
⟨*proof*⟩

lemma *iT-Minus-cut-greater2*: $t \leq k \Longrightarrow (k \ominus I) \downarrow> (k - t) = k \ominus (I \downarrow< t)$
⟨*proof*⟩

lemma *iT-Minus-cut-le*: $(k \ominus I) \downarrow\leq t = k \ominus (I \downarrow\geq (k - t))$
⟨*proof*⟩

lemma *iT-Minus-cut-less*:

$(k \ominus I) \downarrow< t = (\mathit{if}\ t \le k\ \mathit{then}\ k \ominus (I \downarrow> (k - t))\ \mathit{else}\ k \ominus I)$
⟨*proof*⟩

lemma *iT-Minus-cut-ge*:
$(k \ominus I) \downarrow\ge t = (\mathit{if}\ t \le k\ \mathit{then}\ k \ominus (I \downarrow\le (k - t))\ \mathit{else}\ \{\})$
⟨*proof*⟩

lemma *iT-Minus-cut-greater*: $(k \ominus I) \downarrow> t = k \ominus (I \downarrow< (k - t))$
⟨*proof*⟩

lemma *iT-Div-cut-le*:
$0 < k \Longrightarrow (I \oslash k) \downarrow\le t = I \downarrow\le (t * k + (k - \mathit{Suc}\ 0)) \oslash k$
⟨*proof*⟩

lemma *iT-Div-cut-less*:
$0 < k \Longrightarrow (I \oslash k) \downarrow< t = I \downarrow< (t * k) \oslash k$
⟨*proof*⟩

lemma *iT-Div-cut-ge*:
$0 < k \Longrightarrow (I \oslash k) \downarrow\ge t = I \downarrow\ge (t * k) \oslash k$
⟨*proof*⟩

lemma *iT-Div-cut-greater*:
$0 < k \Longrightarrow (I \oslash k) \downarrow> t = I \downarrow> (t * k + (k - \mathit{Suc}\ 0)) \oslash k$
⟨*proof*⟩

lemma *iT-Div-cut-le2*:
$0 < k \Longrightarrow (I \oslash k) \downarrow\le (t\ \mathit{div}\ k) = I \downarrow\le (t - t\ \mathit{mod}\ k + (k - \mathit{Suc}\ 0)) \oslash k$
⟨*proof*⟩

lemma *iT-Div-cut-less2*:
$0 < k \Longrightarrow (I \oslash k) \downarrow< (t\ \mathit{div}\ k) = I \downarrow< (t - t\ \mathit{mod}\ k) \oslash k$
⟨*proof*⟩

lemma *iT-Div-cut-ge2*:
$0 < k \Longrightarrow (I \oslash k) \downarrow\ge (t\ \mathit{div}\ k) = I \downarrow\ge (t - t\ \mathit{mod}\ k) \oslash k$
⟨*proof*⟩

lemma *iT-Div-cut-greater2*:
$0 < k \Longrightarrow (I \oslash k) \downarrow> (t\ \mathit{div}\ k) = I \downarrow> (t - t\ \mathit{mod}\ k + (k - \mathit{Suc}\ 0)) \oslash k$
⟨*proof*⟩

2.3 *inext* and *iprev* with interval operators

lemma *iT-Plus-inext*: $\mathit{inext}\ (n + k)\ (I \oplus k) = (\mathit{inext}\ n\ I) + k$
⟨*proof*⟩

lemma *iT-Plus-iprev*: $iprev\ (n + k)\ (I \oplus k) = (iprev\ n\ I) + k$
⟨*proof*⟩

lemma *iT-Plus-inext2*: $k \le n \Longrightarrow inext\ n\ (I \oplus k) = (inext\ (n - k)\ I) + k$
⟨*proof*⟩
lemma *iT-Plus-prev2*: $k \le n \Longrightarrow iprev\ n\ (I \oplus k) = (iprev\ (n - k)\ I) + k$
⟨*proof*⟩

lemma *iT-Mult-inext*: $inext\ (n * k)\ (I \otimes k) = (inext\ n\ I) * k$
⟨*proof*⟩

lemma *iT-Mult-iprev*: $iprev\ (n * k)\ (I \otimes k) = (iprev\ n\ I) * k$
⟨*proof*⟩

lemma *iT-Mult-inext2-if*:
$inext\ n\ (I \otimes k) = (if\ n\ mod\ k = 0\ then\ (inext\ (n\ div\ k)\ I) * k\ else\ n)$
⟨*proof*⟩

lemma *iT-Mult-iprev2-if*:
$iprev\ n\ (I \otimes k) = (if\ n\ mod\ k = 0\ then\ (iprev\ (n\ div\ k)\ I) * k\ else\ n)$
⟨*proof*⟩

corollary *iT-Mult-inext2*:
$n\ mod\ k = 0 \Longrightarrow inext\ n\ (I \otimes k) = (inext\ (n\ div\ k)\ I) * k$
⟨*proof*⟩

corollary *iT-Mult-iprev2*:
$n\ mod\ k = 0 \Longrightarrow iprev\ n\ (I \otimes k) = (iprev\ (n\ div\ k)\ I) * k$
⟨*proof*⟩

lemma *iT-Plus-neg-inext*:
$k \le n \Longrightarrow inext\ (n - k)\ (I \oplus- k) = inext\ n\ I - k$
⟨*proof*⟩

lemma *iT-Plus-neg-iprev*:
$iprev\ (n - k)\ (I \oplus- k) = iprev\ n\ (I \downarrow\ge k) - k$
⟨*proof*⟩

corollary *iT-Plus-neg-inext2*: $inext\ n\ (I \oplus- k) = inext\ (n + k)\ I - k$
⟨*proof*⟩

corollary *iT-Plus-neg-iprev2*: $iprev\ n\ (I \oplus- k) = iprev\ (n + k)\ (I \downarrow\ge k) - k$
⟨*proof*⟩

lemma *iT-Minus-inext*:
$[\![\ k \ominus I \ne \{\};\ n \le k\ ]\!] \Longrightarrow inext\ (k - n)\ (k \ominus I) = k - iprev\ n\ I$
⟨*proof*⟩

corollary *iT-Minus-inext2*:
$[\![k \ominus I \neq \{\}; n \leq k]\!] \Longrightarrow inext\ n\ (k \ominus I) = k - iprev\ (k - n)\ I$
⟨*proof*⟩

lemma *iT-Minus-iprev*:
$[\![k \ominus I \neq \{\}; n \leq k]\!] \Longrightarrow iprev\ (k - n)\ (k \ominus I) = k - inext\ n\ (I \downarrow\leq k)$
⟨*proof*⟩

lemma *iT-Minus-iprev2*:
$[\![k \ominus I \neq \{\}; n \leq k]\!] \Longrightarrow iprev\ n\ (k \ominus I) = k - inext\ (k - n)\ (I \downarrow\leq k)$
⟨*proof*⟩

lemma *iT-Plus-inext-nth*: $I \neq \{\} \Longrightarrow (I \oplus k) \rightarrow n = (I \rightarrow n) + k$
⟨*proof*⟩

lemma *iT-Plus-iprev-nth*: $[\![finite\ I; I \neq \{\}]\!] \Longrightarrow (I \oplus k) \leftarrow n = (I \leftarrow n) + k$
⟨*proof*⟩

lemma *iT-Mult-inext-nth*: $I \neq \{\} \Longrightarrow (I \otimes k) \rightarrow n = (I \rightarrow n) * k$
⟨*proof*⟩

lemma *iT-Mult-iprev-nth*: $[\![finite\ I; I \neq \{\}]\!] \Longrightarrow (I \otimes k) \leftarrow n = (I \leftarrow n) * k$
⟨*proof*⟩

lemma *iT-Plus-neg-inext-nth*:
$I \oplus- k \neq \{\} \Longrightarrow (I \oplus- k) \rightarrow n = (I \downarrow\geq k \rightarrow n) - k$
⟨*proof*⟩

lemma *iT-Plus-neg-iprev-nth*:
$[\![finite\ I; I \oplus- k \neq \{\}]\!] \Longrightarrow (I \oplus- k) \leftarrow n = (I \downarrow\geq k \leftarrow n) - k$
⟨*proof*⟩

lemma *iT-Minus-inext-nth*:
$k \ominus I \neq \{\} \Longrightarrow (k \ominus I) \rightarrow n = k - ((I \downarrow\leq k) \leftarrow n)$
⟨*proof*⟩

lemma *iT-Minus-iprev-nth*:
$k \ominus I \neq \{\} \Longrightarrow (k \ominus I) \leftarrow n = k - ((I \downarrow\leq k) \rightarrow n)$
⟨*proof*⟩

lemma *iT-Div-ge-inext-nth*:
$[\![I \neq \{\}; \forall x{\in}I.\ \forall y{\in}I.\ x < y \longrightarrow x + k \leq y]\!] \Longrightarrow$
$(I \oslash k) \rightarrow n = (I \rightarrow n)\ div\ k$
⟨*proof*⟩

lemma *iT-Div-mod-inext-nth*:
$[\![I \neq \{\}; \forall x{\in}I.\ \forall y{\in}I.\ x\ mod\ k = y\ mod\ k]\!] \Longrightarrow$

$(I \oslash k) \rightarrow n = (I \rightarrow n)$ *div* k
⟨*proof*⟩

lemma *iT-Div-ge-iprev-nth*:
$[\![$ *finite* I; $I \neq \{\}$; $\forall x \in I.\ \forall y \in I.\ x < y \longrightarrow x + k \leq y\ ]\!] \Longrightarrow$
$(I \oslash k) \leftarrow n = (I \leftarrow n)$ *div* k
⟨*proof*⟩

lemma *iT-Div-mod-iprev-nth*:
$[\![$ *finite* I; $I \neq \{\}$; $\forall x \in I.\ \forall y \in I.\ x$ *mod* $k = y$ *mod* $k\ ]\!] \Longrightarrow$
$(I \oslash k) \leftarrow n = (I \leftarrow n)$ *div* k
⟨*proof*⟩

2.4 Cardinality of intervals with interval operators

lemma *iT-Plus-card*: *card* $(I \oplus k) =$ *card* I
⟨*proof*⟩
lemma *iT-Mult-card*: $0 < k \Longrightarrow$ *card* $(I \otimes k) =$ *card* I
⟨*proof*⟩

lemma *iT-Plus-neg-card*: *card* $(I \oplus- k) =$ *card* $(I \downarrow\geq k)$
⟨*proof*⟩

lemma *iT-Plus-neg-card-le*: *card* $(I \oplus- k) \leq$ *card* I
⟨*proof*⟩

lemma *iT-Minus-card*: *card* $(k \ominus I) =$ *card* $(I \downarrow\leq k)$
⟨*proof*⟩

lemma *iT-Minus-card-le*: *finite* $I \Longrightarrow$ *card* $(k \ominus I) \leq$ *card* I
⟨*proof*⟩

lemma *iT-Div-0-card-if*:
card $(I \oslash 0) =$ (*if* $I = \{\}$ *then* 0 *else* *Suc* 0)
⟨*proof*⟩

lemma *Int-empty-sum*:
$(\sum k \leq (n::nat).$ *if* $\{\} \cap (I\ k) = \{\}$ *then* 0 *else* *Suc* $0) = 0$
⟨*proof*⟩

lemma *iT-Div-mod-partition-card*:
card $(I \cap [n * d\ldots,d -$ *Suc* $0] \oslash d) =$
(*if* $I \cap [n * d\ldots,d -$ *Suc* $0] = \{\}$ *then* 0 *else* *Suc* 0)
⟨*proof*⟩

lemma *iT-Div-conv-count*:
$0 < d \Longrightarrow I \oslash d = \{k.\ I \cap [k * d\ldots,d -$ *Suc* $0] \neq \{\}\}$
⟨*proof*⟩

lemma *iT-Div-conv-count2*:
$[\![0 < d; \mathit{finite}\ I; \mathit{Max}\ I\ \mathit{div}\ d \le n]\!] \Longrightarrow$
$I \oslash d = \{k.\ k \le n \wedge I \cap [k * d\ldots,d - \mathit{Suc}\ 0] \ne \{\}\}$
⟨*proof*⟩

lemma *mod-partition-count-Suc*:
$\{k.\ k \le \mathit{Suc}\ n \wedge I \cap [k * d\ldots,d - \mathit{Suc}\ 0] \ne \{\}\} =$
$\{k.\ k \le n \wedge I \cap [k * d\ldots,d - \mathit{Suc}\ 0] \ne \{\}\} \cup$
$(\mathit{if}\ I \cap [\mathit{Suc}\ n * d\ldots,d - \mathit{Suc}\ 0] \ne \{\}\ \mathit{then}\ \{\mathit{Suc}\ n\}\ \mathit{else}\ \{\})$
⟨*proof*⟩

lemma *iT-Div-card*:
$\bigwedge I.[\![0 < d; \mathit{finite}\ I; \mathit{Max}\ I\ \mathit{div}\ d \le n]\!] \Longrightarrow$
$\mathit{card}\ (I \oslash d) = (\sum k{\le}n.$
$\mathit{if}\ I \cap [k * d\ldots,d - \mathit{Suc}\ 0] = \{\}\ \mathit{then}\ 0\ \mathit{else}\ \mathit{Suc}\ 0)$
⟨*proof*⟩

corollary *iT-Div-card-Suc*:
$\bigwedge I.[\![0 < d; \mathit{finite}\ I; \mathit{Max}\ I\ \mathit{div}\ d \le n]\!] \Longrightarrow$
$\mathit{card}\ (I \oslash d) = (\sum k{<}\mathit{Suc}\ n.$
$\mathit{if}\ I \cap [k * d\ldots,d - \mathit{Suc}\ 0] = \{\}\ \mathit{then}\ 0\ \mathit{else}\ \mathit{Suc}\ 0)$
⟨*proof*⟩
corollary *iT-Div-Max-card*: $[\![0 < d; \mathit{finite}\ I]\!] \Longrightarrow$
$\mathit{card}\ (I \oslash d) = (\sum k{\le}\mathit{Max}\ I\ \mathit{div}\ d.$
$\mathit{if}\ I \cap [k * d\ldots,d - \mathit{Suc}\ 0] = \{\}\ \mathit{then}\ 0\ \mathit{else}\ \mathit{Suc}\ 0)$
⟨*proof*⟩

lemma *iT-Div-card-le*: $0 < k \Longrightarrow \mathit{card}\ (I \oslash k) \le \mathit{card}\ I$
⟨*proof*⟩

lemma *iT-Div-card-inj-on*:
$\mathit{inj\text{-}on}\ (\lambda n.\ n\ \mathit{div}\ k)\ I \Longrightarrow \mathit{card}\ (I \oslash k) = \mathit{card}\ I$
⟨*proof*⟩

lemma *mod-Suc′*:
$0 < n \Longrightarrow \mathit{Suc}\ m\ \mathit{mod}\ n = (\mathit{if}\ m\ \mathit{mod}\ n < n - \mathit{Suc}\ 0\ \mathit{then}\ \mathit{Suc}\ (m\ \mathit{mod}\ n)\ \mathit{else}\ 0)$
⟨*proof*⟩

lemma *div-Suc*:
$0 < n \Longrightarrow \mathit{Suc}\ m\ \mathit{div}\ n = (\mathit{if}\ \mathit{Suc}\ (m\ \mathit{mod}\ n) = n\ \mathit{then}\ \mathit{Suc}\ (m\ \mathit{div}\ n)\ \mathit{else}\ m\ \mathit{div}\ n)$
⟨*proof*⟩

lemma *div-Suc′*:
$0 < n \Longrightarrow Suc\ m\ div\ n = (if\ m\ mod\ n < n - Suc\ 0\ then\ m\ div\ n\ else\ Suc\ (m\ div\ n))$
⟨proof⟩

lemma *iT-Div-card-ge-aux*:
$\bigwedge I.\ [\![\ 0 < d;\ finite\ I;\ Max\ I\ div\ d \le n\ ]\!] \Longrightarrow$
$card\ I\ div\ d + (if\ card\ I\ mod\ d = 0\ then\ 0\ else\ Suc\ 0) \le card\ (I \oslash d)$
⟨proof⟩

lemma *iT-Div-card-ge*:
$card\ I\ div\ d + (if\ card\ I\ mod\ d = 0\ then\ 0\ else\ Suc\ 0) \le card\ (I \oslash d)$
⟨proof⟩

corollary *iT-Div-card-ge-div*: $card\ I\ div\ d \le card\ (I \oslash d)$
⟨proof⟩

There is no better lower bound function f for $i \oslash d$ with $card\ i$ and d as arguments.

lemma *iT-Div-card-ge--is-maximal-lower-bound*:
$\forall I\ d.\ card\ I\ div\ d + (if\ card\ I\ mod\ d = 0\ then\ 0\ else\ Suc\ 0) \le f\ (card\ I)\ d\ \wedge$
$f\ (card\ I)\ d \le card\ (I \oslash d) \Longrightarrow$
$f\ (card\ (I{::}nat\ set))\ d = card\ I\ div\ d + (if\ card\ I\ mod\ d = 0\ then\ 0\ else\ Suc\ 0)$
⟨proof⟩

lemma *iT-Plus-icard*: $icard\ (I \oplus k) = icard\ I$
⟨proof⟩

lemma *iT-Mult-icard*: $0 < k \Longrightarrow icard\ (I \otimes k) = icard\ I$
⟨proof⟩

lemma *iT-Plus-neg-icard*: $icard\ (I \oplus- k) = icard\ (I \downarrow\ge k)$
⟨proof⟩

lemma *iT-Plus-neg-icard-le*: $icard\ (I \oplus- k) \le icard\ I$
⟨proof⟩

lemma *iT-Minus-icard*: $icard\ (k \ominus I) = icard\ (I \downarrow\le k)$
⟨proof⟩

lemma *iT-Minus-icard-le*: $icard\ (k \ominus I) \le icard\ I$
⟨proof⟩

lemma *iT-Div-0-icard-if*: $icard\ (I \oslash 0) = enat\ (if\ I = \{\}\ then\ 0\ else\ Suc\ 0)$
⟨proof⟩

lemma *iT-Div-mod-partition-icard*:
$icard\ (I \cap [n * d\ldots,d - Suc\ 0] \oslash d) =$

$enat\ (if\ I \cap [n * d\ldots,d - Suc\ 0] = \{\}\ then\ 0\ else\ Suc\ 0)$
$\langle proof \rangle$

lemma *iT-Div-icard*:
$[\![\ 0 < d;\ finite\ I \Longrightarrow Max\ I\ div\ d \le n]\!] \Longrightarrow$
$icard\ (I \oslash d) =$
$(if\ finite\ I\ then\ enat\ (\sum k{\le}n.\ if\ I \cap [k * d\ldots,d - Suc\ 0] = \{\}\ then\ 0\ else\ Suc\ 0)\ else\ \infty)$
$\langle proof \rangle$

corollary *iT-Div-Max-icard*: $0 < d \Longrightarrow$
$icard\ (I \oslash d) = (if\ finite\ I$
$then\ enat\ (\sum k{\le}Max\ I\ div\ d.\ if\ I \cap [k * d\ldots,d - Suc\ 0] = \{\}\ then\ 0\ else\ Suc\ 0)\ else\ \infty)$
$\langle proof \rangle$

lemma *iT-Div-icard-le*: $0 < k \Longrightarrow icard\ (I \oslash k) \le icard\ I$
$\langle proof \rangle$

lemma *iT-Div-icard-inj-on*: $inj\text{-}on\ (\lambda n.\ n\ div\ k)\ I \Longrightarrow icard\ (I \oslash k) = icard\ I$
$\langle proof \rangle$

lemma *iT-Div-icard-ge*: $icard\ I\ div\ (enat\ d) + enat\ (if\ icard\ I\ mod\ (enat\ d) = 0\ then\ 0\ else\ Suc\ 0) \le icard\ (I \oslash d)$
$\langle proof \rangle$

corollary *iT-Div-icard-ge-div*: $icard\ I\ div\ (enat\ d) \le icard\ (I \oslash d)$
$\langle proof \rangle$

lemma *iT-Div-icard-ge--is-maximal-lower-bound*:
$\forall I\ d.\ icard\ I\ div\ (enat\ d) + enat\ (if\ icard\ I\ mod\ (enat\ d) = 0\ then\ 0\ else\ Suc\ 0)$
$\le f\ (icard\ I)\ d\ \wedge$
$f\ (icard\ I)\ d \le icard\ (I \oslash d) \Longrightarrow$
$f\ (icard\ (I{::}nat\ set))\ d =$
$icard\ I\ div\ (enat\ d) + enat\ (if\ icard\ I\ mod\ (enat\ d) = 0\ then\ 0\ else\ Suc\ 0)$
$\langle proof \rangle$

2.5 Results about sets of intervals

2.5.1 Set of intervals without and with empty interval

definition *iFROM-UN-set* :: (*nat set*) *set*
where $iFROM\text{-}UN\text{-}set \equiv \bigcup n.\ \{[n\ldots]\}$

definition *iTILL-UN-set* :: (*nat set*) *set*
where $iTILL\text{-}UN\text{-}set \equiv \bigcup n.\ \{[\ldots n]\}$

definition *iIN-UN-set* :: (*nat set*) *set*
where $iIN\text{-}UN\text{-}set \equiv \bigcup n\ d.\ \{[n\ldots,d]\}$

definition *iMOD-UN-set* :: (*nat set*) *set*
where *iMOD-UN-set* $\equiv \bigcup r\ m.\ \{[r,\ mod\ m]\}$

definition *iMODb-UN-set* :: (*nat set*) *set*
where *iMODb-UN-set* $\equiv \bigcup r\ m\ c.\ \{[r,\ mod\ m,\ c]\}$

definition *iFROM-set* :: (*nat set*) *set*
where *iFROM-set* $\equiv \{[n\ldots]\ |n.\ True\}$

definition *iTILL-set* :: (*nat set*) *set*
where *iTILL-set* $\equiv \{[\ldots n]\ |n.\ True\}$

definition *iIN-set* :: (*nat set*) *set*
where *iIN-set* $\equiv \{[n\ldots,d]\ |n\ d.\ True\}$

definition *iMOD-set* :: (*nat set*) *set*
where *iMOD-set* $\equiv \{[r,\ mod\ m]\ |r\ m.\ True\}$

definition *iMODb-set* :: (*nat set*) *set*
where *iMODb-set* $\equiv \{[r,\ mod\ m,\ c]\ |r\ m\ c.\ True\}$

definition *iMOD2-set* :: (*nat set*) *set*
where *iMOD2-set* $\equiv \{[r,\ mod\ m]\ |r\ m.\ 2 \le m\}$

definition *iMODb2-set* :: (*nat set*) *set*
where *iMODb2-set* $\equiv \{[r,\ mod\ m,\ c]\ |r\ m\ c.\ 2 \le m \land 1 \le c\}$

definition *iMOD2-UN-set* :: (*nat set*) *set*
where *iMOD2-UN-set* $\equiv \bigcup r.\ \bigcup m{\in}\{2..\}.\ \{[r,\ mod\ m]\}$

definition *iMODb2-UN-set* :: (*nat set*) *set*
where *iMODb2-UN-set* $\equiv \bigcup r.\ \bigcup m{\in}\{2..\}.\ \bigcup c{\in}\{1..\}.\ \{[r,\ mod\ m,\ c]\}$

lemmas *i-set-defs* =
iFROM-set-def iTILL-set-def iIN-set-def
iMOD-set-def iMODb-set-def
iMOD2-set-def iMODb2-set-def
lemmas *i-UN-set-defs* =
iFROM-UN-set-def iTILL-UN-set-def iIN-UN-set-def
iMOD-UN-set-def iMODb-UN-set-def
iMOD2-UN-set-def iMODb2-UN-set-def

lemma *iFROM-set-UN-set-eq*: *iFROM-set* = *iFROM-UN-set*
⟨*proof*⟩

lemma
iTILL-set-UN-set-eq: *iTILL-set* = *iTILL-UN-set* **and**
iIN-set-UN-set-eq: *iIN-set* = *iIN-UN-set* **and**
iMOD-set-UN-set-eq: *iMOD-set* = *iMOD-UN-set* **and**
iMODb-set-UN-set-eq: *iMODb-set* = *iMODb-UN-set*
⟨*proof*⟩

lemma *iMOD2-set-UN-set-eq*: *iMOD2-set* = *iMOD2-UN-set*
⟨*proof*⟩

lemma *iMODb2-set-UN-set-eq*: *iMODb2-set* = *iMODb2-UN-set*
⟨*proof*⟩

lemmas *i-set-i-UN-set-sets-eq* =
iFROM-set-UN-set-eq
iTILL-set-UN-set-eq
iIN-set-UN-set-eq
iMOD-set-UN-set-eq
iMODb-set-UN-set-eq
iMOD2-set-UN-set-eq
iMODb2-set-UN-set-eq

lemma *iMOD2-set-iMOD-set-subset*: *iMOD2-set* ⊆ *iMOD-set*
⟨*proof*⟩

lemma *iMODb2-set-iMODb-set-subset*: *iMODb2-set* ⊆ *iMODb-set*
⟨*proof*⟩

definition *i-set* :: (*nat set*) *set*
where *i-set* ≡ *iFROM-set* ∪ *iTILL-set* ∪ *iIN-set* ∪ *iMOD-set* ∪ *iMODb-set*

definition *i-UN-set* :: (*nat set*) *set*
where *i-UN-set* ≡ *iFROM-UN-set* ∪ *iTILL-UN-set* ∪ *iIN-UN-set* ∪ *iMOD-UN-set* ∪ *iMODb-UN-set*

Minimal definitions for *i-set* and *i-set*

definition *i-set-min* :: (*nat set*) *set*
where *i-set-min* ≡ *iFROM-set* ∪ *iIN-set* ∪ *iMOD2-set* ∪ *iMODb2-set*

definition *i-UN-set-min* :: (*nat set*) *set*
where *i-UN-set-min* ≡ *iFROM-UN-set* ∪ *iIN-UN-set* ∪ *iMOD2-UN-set* ∪ *iMODb2-UN-set*

definition *i-set0* :: (*nat set*) *set*
where *i-set0* ≡ *insert* {} *i-set*

lemma *i-set-i-UN-set-eq*: *i-set* = *i-UN-set*
⟨*proof*⟩

lemma *i-set-min-i-UN-set-min-eq*: *i-set-min* = *i-UN-set-min*
⟨*proof*⟩

lemma *i-set-min-eq*: *i-set* = *i-set-min*
⟨*proof*⟩

corollary *i-UN-set-i-UN-min-set-eq*: *i-UN-set* = *i-UN-set-min*
⟨*proof*⟩

lemma *i-set-min-is-minimal-let*:
  let s1 = *iFROM-set*; *s2*= *iIN-set*; *s3*= *iMOD2-set*; *s4*= *iMODb2-set in*
  s1 ∩ *s2* = {} ∧ *s1* ∩ *s3* = {} ∧ *s1* ∩ *s4* = {} ∧
  s2 ∩ *s3* = {} ∧ *s2* ∩ *s4* = {} ∧ *s3* ∩ *s4* = {}
⟨*proof*⟩

lemmas *i-set-min-is-minimal* = *i-set-min-is-minimal-let*[*simplified*]

inductive-set *i-set-ind*:: (*nat set*) *set*
where
  i-set-ind-FROM[*intro*!]: [*n*. . .] ∈ *i-set-ind*
| *i-set-ind-TILL*[*intro*!]: [. . .*n*] ∈ *i-set-ind*
| *i-set-ind-IN*[*intro*!]: [*n*. . .,*d*] ∈ *i-set-ind*
| *i-set-ind-MOD*[*intro*!]: [*r*, *mod m*] ∈ *i-set-ind*
| *i-set-ind-MODb*[*intro*!]: [*r*, *mod m*, *c*] ∈ *i-set-ind*

inductive-set *i-set0-ind* :: (*nat set*) *set*
where
  i-set0-ind-empty[*intro*!] : {} ∈ *i-set0-ind*
| *i-set0-ind-i-set*[*intro*]: *I* ∈ *i-set-ind* ⟹ *I* ∈ *i-set0-ind*

The introduction rule *i-set0-ind-i-set* is not declared a safe introduction rule, because it would disturb the correct usage of the *safe* method.

lemma *i-set-ind-subset-i-set0-ind*: *i-set-ind* ⊆ *i-set0-ind*
⟨*proof*⟩

lemma
  i-set0-ind-FROM[*intro*!] : [*n*. . .] ∈ *i-set0-ind* **and**
  i-set0-ind-TILL[*intro*!] : [. . .*n*] ∈ *i-set0-ind* **and**
  i-set0-ind-IN[*intro*!] : [*n*. . .,*d*] ∈ *i-set0-ind* **and**
  i-set0-ind-MOD[*intro*!] : [*r*, *mod m*] ∈ *i-set0-ind* **and**
  i-set0-ind-MODb[*intro*!] : [*r*, *mod m*, *c*] ∈ *i-set0-ind*
⟨*proof*⟩

lemmas *i-set0-ind-intros2* =
  i-set0-ind-empty
  i-set0-ind-FROM
  i-set0-ind-TILL

i-set0-ind-IN
i-set0-ind-MOD
i-set0-ind-MODb

lemma *i-set-i-set-ind-eq*: *i-set* = *i-set-ind*
⟨*proof*⟩

lemma *i-set0-i-set0-ind-eq*: *i-set0* = *i-set0-ind*
⟨*proof*⟩

lemma *i-set-imp-not-empty*: $I \in$ *i-set* $\Longrightarrow I \neq \{\}$
⟨*proof*⟩

lemma *i-set0-i-set-mem-conv*: $(I \in$ *i-set0*$) = (I \in$ *i-set* $\vee I = \{\})$
⟨*proof*⟩

lemma *i-set-i-set0-mem-conv*: $(I \in$ *i-set*$) = (I \in$ *i-set0* $\wedge I \neq \{\})$
⟨*proof*⟩

lemma *i-set0-i-set-conv*: *i-set0* $- \{\{\}\} =$ *i-set*
⟨*proof*⟩

corollary *i-set-subset-i-set0*: *i-set* $\subseteq$ *i-set0*
⟨*proof*⟩

lemma *i-set-singleton*: $\{a\} \in$ *i-set*
⟨*proof*⟩

lemma *i-set0-singleton*: $\{a\} \in$ *i-set0*
⟨*proof*⟩

corollary
i-set-FROM[*intro*!] : [*n*...] ∈ *i-set* **and**
i-set-TILL[*intro*!] : [...*n*] ∈ *i-set* **and**
i-set-IN[*intro*!] : [*n*...,*d*] ∈ *i-set* **and**
i-set-MOD[*intro*!] : [*r*, *mod m*] ∈ *i-set* **and**
i-set-MODb[*intro*!] : [*r*, *mod m*, *c*] ∈ *i-set*
⟨*proof*⟩

lemmas *i-set-intros* =
i-set-FROM
i-set-TILL
i-set-IN
i-set-MOD
i-set-MODb

lemma
i-set0-empty[*intro*!]: {} ∈ *i-set0* **and**
i-set0-FROM[*intro*!] : [*n*...] ∈ *i-set0* **and**

$i\text{-}set0\text{-}TILL[intro!]$: $[\ldots n] \in i\text{-}set0$ **and**
$i\text{-}set0\text{-}IN[intro!]$: $[n\ldots,d] \in i\text{-}set0$ **and**
$i\text{-}set0\text{-}MOD[intro!]$: $[r,\ mod\ m] \in i\text{-}set0$ **and**
$i\text{-}set0\text{-}MODb[intro!]$: $[r,\ mod\ m,\ c] \in i\text{-}set0$
⟨*proof*⟩

lemmas *i-set0-intros* =
i-set0-empty
i-set0-FROM
i-set0-TILL
i-set0-IN
i-set0-MOD
i-set0-MODb

lemma *i-set-infinite-as-iMOD*:
$[\![\ I \in i\text{-}set;\ infinite\ I\ ]\!] \Longrightarrow \exists\, r\ m.\ I = [r,\ mod\ m]$
⟨*proof*⟩

lemma *i-set-finite-as-iMODb*:
$[\![\ I \in i\text{-}set;\ finite\ I\ ]\!] \Longrightarrow \exists\, r\ m\ c.\ I = [r,\ mod\ m,\ c]$
⟨*proof*⟩

lemma *i-set-as-iMOD-iMODb*:
$I \in i\text{-}set \Longrightarrow (\exists\, r\ m.\ I = [r,\ mod\ m]) \vee (\exists\, r\ m\ c.\ I = [r,\ mod\ m,\ c])$
⟨*proof*⟩

2.5.2 Interval sets are closed under cutting

lemma *i-set-cut-le-ge-closed-disj*:
$[\![\ I \in i\text{-}set;\ t \in I;\ cut\text{-}op = (\downarrow\le) \vee cut\text{-}op = (\downarrow\ge)\ ]\!] \Longrightarrow$
$cut\text{-}op\ I\ t \in i\text{-}set$
⟨*proof*⟩

corollary
i-set-cut-le-closed: $[\![\ I \in i\text{-}set;\ t \in I\ ]\!] \Longrightarrow I \downarrow\le t \in i\text{-}set$ **and**
i-set-cut-ge-closed: $[\![\ I \in i\text{-}set;\ t \in I\ ]\!] \Longrightarrow I \downarrow\ge t \in i\text{-}set$
⟨*proof*⟩

lemmas *i-set-cut-le-ge-closed* = *i-set-cut-le-closed i-set-cut-ge-closed*

lemma *i-set0-cut-closed-disj*:
$[\![\ I \in i\text{-}set0;$
$cut\text{-}op = (\downarrow\le) \vee cut\text{-}op = (\downarrow\ge) \vee$
$cut\text{-}op = (\downarrow<) \vee cut\text{-}op = (\downarrow>)\ ]\!] \Longrightarrow$
$cut\text{-}op\ I\ t \in i\text{-}set0$
⟨*proof*⟩

corollary

i-set0-cut-le-closed: $I \in$ *i-set0* $\Longrightarrow I \downarrow\leq t \in$ *i-set0* **and**
i-set0-cut-less-closed: $I \in$ *i-set0* $\Longrightarrow I \downarrow< t \in$ *i-set0* **and**
i-set0-cut-ge-closed: $I \in$ *i-set0* $\Longrightarrow I \downarrow\geq t \in$ *i-set0* **and**
i-set0-cut-greater-closed: $I \in$ *i-set0* $\Longrightarrow I \downarrow> t \in$ *i-set0*
⟨*proof*⟩

lemmas *i-set0-cut-closed* =
i-set0-cut-le-closed
i-set0-cut-less-closed
i-set0-cut-ge-closed
i-set0-cut-greater-closed

2.5.3 Interval sets are closed under addition and multiplication

lemma *i-set-Plus-closed*: $I \in$ *i-set* $\Longrightarrow I \oplus k \in$ *i-set*
⟨*proof*⟩

lemma *i-set-Mult-closed*: $I \in$ *i-set* $\Longrightarrow I \otimes k \in$ *i-set*
⟨*proof*⟩

lemma *i-set0-Plus-closed*: $I \in$ *i-set0* $\Longrightarrow I \oplus k \in$ *i-set0*
⟨*proof*⟩

lemma *i-set0-Mult-closed*: $I \in$ *i-set0* $\Longrightarrow I \otimes k \in$ *i-set0*
⟨*proof*⟩

2.5.4 Interval sets are closed with certain conditions under subtraction

lemma *i-set-Plus-neg-closed*:
$[\![I \in$ *i-set*; $\exists x \in I.\ k \leq x]\!] \Longrightarrow I \oplus- k \in$ *i-set*
⟨*proof*⟩

lemma *i-set-Minus-closed*:
$[\![I \in$ *i-set*; *iMin* $I \leq k]\!] \Longrightarrow k \ominus I \in$ *i-set*
⟨*proof*⟩

lemma *i-set0-Plus-neg-closed*: $I \in$ *i-set0* $\Longrightarrow I \oplus- k \in$ *i-set0*
⟨*proof*⟩

lemma *i-set0-Minus-closed*: $I \in$ *i-set0* $\Longrightarrow k \ominus I \in$ *i-set0*
⟨*proof*⟩

lemmas *i-set-IntOp-closed* =
i-set-Plus-closed
i-set-Mult-closed
i-set-Plus-neg-closed
i-set-Minus-closed

lemmas *i-set0-IntOp-closed* =
i-set0-Plus-closed
i-set0-Mult-closed
i-set0-Plus-neg-closed
i-set0-Minus-closed

2.5.5 Interval sets are not closed under division

lemma *iMOD-div-mod-gr0-not-in-i-set*:
$[\![\ 0 < k;\ k < m;\ 0 < m \ mod \ k \ ]\!] \Longrightarrow [r,\ mod\ m] \oslash k \notin i\text{-}set$
⟨*proof*⟩

lemma *iMODb-div-mod-gr0-not-in-i-set*:
$[\![\ 0 < k;\ k < m;\ 0 < m \ mod \ k;\ k \le c \ ]\!] \Longrightarrow [r,\ mod\ m,\ c] \oslash k \notin i\text{-}set$
⟨*proof*⟩

lemma $[0,\ mod\ 3] \oslash 2 \notin i\text{-}set$
⟨*proof*⟩

lemma *i-set-Div-not-closed*: $Suc\ 0 < k \Longrightarrow \exists I{\in}i\text{-}set.\ I \oslash k \notin i\text{-}set$
⟨*proof*⟩
lemma *i-set0-Div-not-closed*: $Suc\ 0 < k \Longrightarrow \exists I{\in}i\text{-}set0.\ I \oslash k \notin i\text{-}set0$
⟨*proof*⟩

2.5.6 Sets of intervals closed under division

inductive-set *NatMultiples* :: *nat set* ⇒ *nat set*
for *F* :: *nat set*
where
NatMultiples-Factor: $k \in F \Longrightarrow k \in NatMultiples\ F$
| *NatMultiples-Product*: $[\![\ k \in F;\ m \in NatMultiples\ F \ ]\!] \Longrightarrow k * m \in NatMultiples\ F$

lemma *NatMultiples-ex-divisor*: $m \in NatMultiples\ F \Longrightarrow \exists k{\in}F.\ m\ mod\ k = 0$
⟨*proof*⟩

lemma *NatMultiples-product-factor*: $[\![\ a \in F;\ b \in F \ ]\!] \Longrightarrow a * b \in NatMultiples\ F$
⟨*proof*⟩

lemma *NatMultiples-product-factor-multiple*:
$[\![\ a \in F;\ b \in NatMultiples\ F \ ]\!] \Longrightarrow a * b \in NatMultiples\ F$
⟨*proof*⟩

lemma *NatMultiples-product-multiple-factor*:
$[\![\ a \in NatMultiples\ F;\ b \in F \ ]\!] \Longrightarrow a * b \in NatMultiples\ F$
⟨*proof*⟩

lemma *NatMultiples-product-multiple*:
$[\![\ a \in NatMultiples\ F;\ b \in NatMultiples\ F \ ]\!] \Longrightarrow a * b \in NatMultiples\ F$
⟨*proof*⟩

Subset of *i-set* containing only continuous intervals, i. e., without *iMOD* and *iMODb*.

inductive-set *i-set-cont* :: (*nat set*) *set*
where
  i-set-cont-FROM[*intro*]: $[n\ldots] \in$ *i-set-cont*
| *i-set-cont-TILL*[*intro*]: $[\ldots n] \in$ *i-set-cont*
| *i-set-cont-IN*[*intro*]: $[n\ldots,d] \in$ *i-set-cont*

definition *i-set0-cont* :: (*nat set*) *set*
where *i-set0-cont* $\equiv$ *insert* {} *i-set-cont*

lemma *i-set-cont-subset-i-set*: *i-set-cont* $\subseteq$ *i-set*
⟨*proof*⟩

lemma *i-set0-cont-subset-i-set0*: *i-set0-cont* $\subseteq$ *i-set0*
⟨*proof*⟩

Minimal definition of *i-set-cont*

inductive-set *i-set-cont-min* :: (*nat set*) *set*
where
  i-set-cont-FROM[*intro*]: $[n\ldots] \in$ *i-set-cont-min*
| *i-set-cont-IN*[*intro*]: $[n\ldots,d] \in$ *i-set-cont-min*

definition *i-set0-cont-min* :: (*nat set*) *set*
where *i-set0-cont-min* $\equiv$ *insert* {} *i-set-cont-min*

lemma *i-set-cont-min-eq*: *i-set-cont* = *i-set-cont-min*
⟨*proof*⟩

inext and *iprev* with continuous intervals

lemma *i-set-cont-inext*:
$[\![\ I \in \textit{i-set-cont};\ n \in I;\ \textit{finite}\ I \Longrightarrow n < \textit{Max}\ I\ ]\!] \Longrightarrow \textit{inext}\ n\ I = \textit{Suc}\ n$
⟨*proof*⟩

lemma *i-set-cont-iprev*:
$[\![\ I \in \textit{i-set-cont};\ n \in I;\ \textit{iMin}\ I < n\ ]\!] \Longrightarrow \textit{iprev}\ n\ I = n - \textit{Suc}\ 0$
⟨*proof*⟩

lemma *i-set-cont-inext--less*:
$[\![\ I \in \textit{i-set-cont};\ n \in I;\ n < n0;\ n0 \in I\ ]\!] \Longrightarrow \textit{inext}\ n\ I = \textit{Suc}\ n$
⟨*proof*⟩

lemma *i-set-cont-iprev--greater*:
$[\![\ I \in \textit{i-set-cont};\ n \in I;\ n0 < n;\ n0 \in I\ ]\!] \Longrightarrow \textit{iprev}\ n\ I = n - \textit{Suc}\ 0$
⟨*proof*⟩

Sets of modulo intervals

inductive-set *i-set-mult* :: *nat* $\Rightarrow$ (*nat set*) *set*

for *k* :: *nat*
where
i-set-mult-FROM[*intro*!]: [*n*. . .] ∈ *i-set-mult k*
| *i-set-mult-TILL*[*intro*!]: [. . .*n*] ∈ *i-set-mult k*
| *i-set-mult-IN*[*intro*!]: [*n*. . .,*d*] ∈ *i-set-mult k*
| *i-set-mult-MOD*[*intro*!]: [*r*, *mod m* ∗ *k*] ∈ *i-set-mult k*
| *i-set-mult-MODb*[*intro*!]: [*r*, *mod m* ∗ *k*, *c*] ∈ *i-set-mult k*

definition *i-set0-mult* :: *nat* ⇒ (*nat set*) *set*
where *i-set0-mult k* ≡ *insert* {} (*i-set-mult k*)

lemma
i-set0-mult-empty[*intro*!]: {} ∈ *i-set0-mult k* **and**
i-set0-mult-FROM[*intro*!] : [*n*. . .] ∈ *i-set0-mult k* **and**
i-set0-mult-TILL[*intro*!] : [. . .*n*] ∈ *i-set0-mult k* **and**
i-set0-mult-IN[*intro*!] : [*n*. . .,*d*] ∈ *i-set0-mult k* **and**
i-set0-mult-MOD[*intro*!] : [*r*, *mod m* ∗ *k*] ∈ *i-set0-mult k* **and**
i-set0-mult-MODb[*intro*!] : [*r*, *mod m* ∗ *k*, *c*] ∈ *i-set0-mult k*
⟨*proof*⟩

lemmas *i-set0-mult-intros* =
i-set0-mult-empty
i-set0-mult-FROM
i-set0-mult-TILL
i-set0-mult-IN
i-set0-mult-MOD
i-set0-mult-MODb

lemma *i-set-mult-subset-i-set0-mult*: *i-set-mult k* ⊆ *i-set0-mult k*
⟨*proof*⟩

lemma *i-set-mult-subset-i-set*: *i-set-mult k* ⊆ *i-set*
⟨*proof*⟩

lemma *i-set0-mult-subset-i-set0*: *i-set0-mult k* ⊆ *i-set0*
⟨*proof*⟩

lemma *i-set-mult-0-eq-i-set-cont*: *i-set-mult 0* = *i-set-cont*
⟨*proof*⟩

lemma *i-set0-mult-0-eq-i-set0-cont*: *i-set0-mult 0* = *i-set0-cont*
⟨*proof*⟩

lemma *i-set-mult-1-eq-i-set*: *i-set-mult* (*Suc 0*) = *i-set*
⟨*proof*⟩

lemma *i-set0-mult-1-eq-i-set0*: *i-set0-mult* (*Suc 0*) = *i-set0*
⟨*proof*⟩

lemma *i-set-mult-imp-not-empty*: $I \in \textit{i-set-mult } k \Longrightarrow I \neq \{\}$
⟨*proof*⟩

lemma *iMOD-in-i-set-mult-imp-divisor-mod-0*:
$[\![m \neq \textit{Suc } 0;\ [r, \textit{mod } m] \in \textit{i-set-mult } k]\!] \Longrightarrow m \textit{ mod } k = 0$
⟨*proof*⟩

lemma
divisor-mod-0-imp-iMOD-in-i-set-mult: $m \textit{ mod } k = 0 \Longrightarrow [r, \textit{mod } m] \in \textit{i-set-mult } k$ **and**
divisor-mod-0-imp-iMODb-in-i-set-mult: $m \textit{ mod } k = 0 \Longrightarrow [r, \textit{mod } m, c] \in \textit{i-set-mult } k$
⟨*proof*⟩

lemma *iMOD-in-i-set-mult--divisor-mod-0-conv*:
$m \neq \textit{Suc } 0 \Longrightarrow ([r, \textit{mod } m] \in \textit{i-set-mult } k) = (m \textit{ mod } k = 0)$
⟨*proof*⟩

lemma *i-set-mult-neq1-subset-i-set*: $k \neq \textit{Suc } 0 \Longrightarrow \textit{i-set-mult } k \subset \textit{i-set}$
⟨*proof*⟩

lemma *mod-0-imp-i-set-mult-subset*:
$a \textit{ mod } b = 0 \Longrightarrow \textit{i-set-mult } a \subseteq \textit{i-set-mult } b$
⟨*proof*⟩

lemma *i-set-mult-subset-imp-mod-0*:
$[\![a \neq \textit{Suc } 0;\ (\textit{i-set-mult } a \subseteq \textit{i-set-mult } b)]\!] \Longrightarrow a \textit{ mod } b = 0$
⟨*proof*⟩

lemma *i-set-mult-subset-conv*:
$a \neq \textit{Suc } 0 \Longrightarrow (\textit{i-set-mult } a \subseteq \textit{i-set-mult } b) = (a \textit{ mod } b = 0)$
⟨*proof*⟩

lemma *i-set-mult-mod-0-div*:
$[\![I \in \textit{i-set-mult } k;\ k \textit{ mod } d = 0]\!] \Longrightarrow I \oslash d \in \textit{i-set-mult } (k \textit{ div } d)$
⟨*proof*⟩

Intervals from *i-set-mult k* remain in *i-set* after division by d a divisor of k.

corollary *i-set-mult-mod-0-div-i-set*:
$[\![I \in \textit{i-set-mult } k;\ k \textit{ mod } d = 0]\!] \Longrightarrow I \oslash d \in \textit{i-set}$
⟨*proof*⟩
corollary *i-set-mult-div-self-i-set*:
$I \in \textit{i-set-mult } k \Longrightarrow I \oslash k \in \textit{i-set}$
⟨*proof*⟩

lemma *i-set-mod-0-mult-in-i-set-mult*:
$[\![I \in \textit{i-set};\ m \textit{ mod } k = 0]\!] \Longrightarrow I \otimes m \in \textit{i-set-mult } k$

⟨*proof*⟩

lemma *i-set-self-mult-in-i-set-mult*:
$I \in \textit{i-set} \Longrightarrow I \otimes k \in \textit{i-set-mult}\ k$
⟨*proof*⟩

lemma *i-set-mult-mod-gr0-div-not-in-i-set*:
$[\![\ 0 < k;\ 0 < d;\ 0 < k\ \textit{mod}\ d\ ]\!] \Longrightarrow \exists I{\in}\textit{i-set-mult}\ k.\ I \oslash d \notin \textit{i-set}$
⟨*proof*⟩

lemma *i-set0-mult-mod-0-div*:
$[\![\ I \in \textit{i-set0-mult}\ k;\ k\ \textit{mod}\ d = 0\ ]\!] \Longrightarrow I \oslash d \in \textit{i-set0-mult}\ (k\ \textit{div}\ d)$
⟨*proof*⟩

corollary *i-set0-mult-mod-0-div-i-set0*:
$[\![\ I \in \textit{i-set0-mult}\ k;\ k\ \textit{mod}\ d = 0\ ]\!] \Longrightarrow I \oslash d \in \textit{i-set0}$
⟨*proof*⟩

corollary *i-set0-mult-div-self-i-set0*:
$I \in \textit{i-set0-mult}\ k \Longrightarrow I \oslash k \in \textit{i-set0}$
⟨*proof*⟩

lemma *i-set0-mod-0-mult-in-i-set0-mult*:
$[\![\ I \in \textit{i-set0};\ m\ \textit{mod}\ k = 0\ ]\!] \Longrightarrow I \otimes m \in \textit{i-set0-mult}\ k$
⟨*proof*⟩

lemma *i-set0-self-mult-in-i-set0-mult*:
$I \in \textit{i-set0} \Longrightarrow I \otimes k \in \textit{i-set0-mult}\ k$
⟨*proof*⟩

lemma *i-set0-mult-mod-gr0-div-not-in-i-set0*:
$[\![\ 0 < k;\ 0 < d;\ 0 < k\ \textit{mod}\ d\ ]\!] \Longrightarrow \exists I{\in}\textit{i-set0-mult}\ k.\ I \oslash d \notin \textit{i-set0}$
⟨*proof*⟩

end

3 Temporal logic operators on natural intervals

theory *IL-TemporalOperators*
imports *IL-IntervalOperators*
begin

Bool : some additional properties

instantiation *bool* :: {*ord*, *zero*, *one*, *plus*, *times*, *order*}
begin

definition *Zero-bool-def* [*simp*]: $0 \equiv \textit{False}$
definition *One-bool-def* [*simp*]: $1 \equiv \textit{True}$
definition *add-bool-def*: $a + b \equiv a \vee b$

definition *mult-bool-def*: $a * b \equiv a \wedge b$

instance ⟨*proof*⟩

end

value *False* < *True*
value *True* < *True*
value *True* ≤ *True*

lemmas *bool-op-rel-defs* =
add-bool-def
mult-bool-def
less-bool-def
le-bool-def

instance *bool* :: *wellorder*
⟨*proof*⟩

instance *bool* :: *comm-semiring-1*
⟨*proof*⟩

3.1 Basic definitions

lemma *UNIV-nat*: $\mathbb{N}$ = (*UNIV*::*nat set*)
⟨*proof*⟩

Universal temporal operator: Always/Globally

definition *iAll* :: *iT* ⇒ (*Time* ⇒ *bool*) ⇒ *bool* — Always
where *iAll I P* ≡ ∀ *t*∈*I*. *P t*

Existential temporal operator: Eventually/Finally

definition *iEx* :: *iT* ⇒ (*Time* ⇒ *bool*) ⇒ *bool* — Eventually
where *iEx I P* ≡ ∃ *t*∈*I*. *P t*

syntax
-iAll :: *Time* ⇒ *iT* ⇒ (*Time* ⇒ *bool*) ⇒ *bool* (‹(*3*□ - -./ -)› [*0*, *0*, *10*] *10*)
-iEx :: *Time* ⇒ *iT* ⇒ (*Time* ⇒ *bool*) ⇒ *bool* (‹(*3*◇ - -./ -)› [*0*, *0*, *10*] *10*)
syntax-consts
-iAll ⇌ *iAll* **and**
-iEx ⇌ *iEx*
translations
□ *t I*. *P* ⇌ *CONST iAll I* (λ*t*. *P*)
◇ *t I*. *P* ⇌ *CONST iEx I* (λ*t*. *P*)

Future temporal operator: Next

definition *iNext* :: *Time* ⇒ *iT* ⇒ (*Time* ⇒ *bool*) ⇒ *bool* — Next
where *iNext t0 I P* ≡ *P* (*inext t0 I*)

Past temporal operator: Last/Previous

definition *iLast* :: *Time* ⇒ *iT* ⇒ (*Time* ⇒ *bool*) ⇒ *bool* — Last
where *iLast t0 I P* ≡ *P* (*iprev t0 I*)

syntax
-*iNext* :: *Time* ⇒ *Time* ⇒ *iT* ⇒ (*Time* ⇒ *bool*) ⇒ *bool* (‹(3○ - - -./ -)› [*0*, *0*, *10*] *10*)
-*iLast* :: *Time* ⇒ *Time* ⇒ *iT* ⇒ (*Time* ⇒ *bool*) ⇒ *bool* (‹(3⊖ - - -./ -)› [*0*, *0*, *10*] *10*)
syntax-consts
-*iNext* ⇌ *iNext* **and**
-*iLast* ⇌ *iLast*
translations
○ *t t0 I*. *P* ⇌ *CONST iNext t0 I* (λ*t*. *P*)
⊖ *t t0 I*. *P* ⇌ *CONST iLast t0 I* (λ*t*. *P*)

lemma ○ *t 10* [*0*...]. (*t* + *10* > *10*)
⟨*proof*⟩

The following versions of Next and Last operator differ in the cases where no next/previous element exists or specified time point is not in interval: the weak versions return *True* and the strong versions return *False*.

definition *iNextWeak* :: *Time* ⇒ *iT* ⇒ (*Time* ⇒ *bool*) ⇒ *bool* — Weak Next
where *iNextWeak t0 I P* ≡ (□ *t* {*inext t0 I*} ↓> *t0*. *P t*)

definition *iNextStrong* :: *Time* ⇒ *iT* ⇒ (*Time* ⇒ *bool*) ⇒ *bool* — Strong Next
where *iNextStrong t0 I P* ≡ (◇ *t* {*inext t0 I*} ↓> *t0*. *P t*)

definition *iLastWeak* :: *Time* ⇒ *iT* ⇒ (*Time* ⇒ *bool*) ⇒ *bool* — Weak Last
where *iLastWeak t0 I P* ≡ (□ *t* {*iprev t0 I*} ↓< *t0*. *P t*)

definition *iLastStrong* :: *Time* ⇒ *iT* ⇒ (*Time* ⇒ *bool*) ⇒ *bool* — Strong Last
where *iLastStrong t0 I P* ≡ (◇ *t* {*iprev t0 I*} ↓< *t0*. *P t*)

syntax
-*iNextWeak* :: *Time* ⇒ *Time* ⇒ *iT* ⇒ (*Time* ⇒ *bool*) ⇒ *bool* (‹(3$○_W$ - - -./ -)› [*0*, *0*, *10*] *10*)
-*iNextStrong* :: *Time* ⇒ *Time* ⇒ *iT* ⇒ (*Time* ⇒ *bool*) ⇒ *bool* (‹(3$○_S$ - - -./ -)› [*0*, *0*, *10*] *10*)
-*iLastWeak* :: *Time* ⇒ *Time* ⇒ *iT* ⇒ (*Time* ⇒ *bool*) ⇒ *bool* (‹(3$⊖_W$ - - -./ -)› [*0*, *0*, *10*] *10*)
-*iLastStrong* :: *Time* ⇒ *Time* ⇒ *iT* ⇒ (*Time* ⇒ *bool*) ⇒ *bool* (‹(3$⊖_S$ - - -./ -)› [*0*, *0*, *10*] *10*)
syntax-consts
-*iNextWeak* ⇌ *iNextWeak* **and**
-*iNextStrong* ⇌ *iNextStrong* **and**
-*iLastWeak* ⇌ *iLastWeak* **and**
-*iLastStrong* ⇌ *iLastStrong*

translations
$\bigcirc_W$ *t t0 I. P* ⇌ *CONST iNextWeak t0 I* (*λt. P*)
$\bigcirc_S$ *t t0 I. P* ⇌ *CONST iNextStrong t0 I* (*λt. P*)
$\ominus_W$ *t t0 I. P* ⇌ *CONST iLastWeak t0 I* (*λt. P*)
$\ominus_S$ *t t0 I. P* ⇌ *CONST iLastStrong t0 I* (*λt. P*)

Some examples for Next and Last operator

lemma ○ *t 5* [*0. . .,10*]. ([*0::int,10,20,30,40,50,60,70,80,90*] ! *t* < *80*)
⟨*proof*⟩

lemma ⊖ *t 5* [*0. . .,10*]. ([*0::int,10,20,30,40,50,60,70,80,90*] ! *t* < *80*)
⟨*proof*⟩

Temporal Until operator

definition *iUntil* :: *iT* ⇒ (*Time* ⇒ *bool*) ⇒ (*Time* ⇒ *bool*) ⇒ *bool* — Until
where *iUntil I P Q* ≡ ◇ *t I. Q t* ∧ (□ *t′* (*I* ↓< *t*). *P t′*)

Temporal Since operator (past operator corresponding to Until)

definition *iSince* :: *iT* ⇒ (*Time* ⇒ *bool*) ⇒ (*Time* ⇒ *bool*) ⇒ *bool* — Since
where *iSince I P Q* ≡ ◇ *t I. Q t* ∧ (□ *t′* (*I* ↓> *t*). *P t′*)

syntax
-iUntil :: *Time* ⇒ *Time* ⇒ *iT* ⇒ (*Time* ⇒ *bool*) ⇒ (*Time* ⇒ *bool*) ⇒ *bool*
(‹(-./ - (*3*$\mathcal{U}$ - -)./ -)› [*10, 0, 0, 0, 10*] *10*)
-iSince :: *Time* ⇒ *Time* ⇒ *iT* ⇒ (*Time* ⇒ *bool*) ⇒ (*Time* ⇒ *bool*) ⇒ *bool*
(‹(-./ - (*3*$\mathcal{S}$ - -)./ -)› [*10, 0, 0, 0, 10*] *10*)
syntax-consts
-iUntil ⇌ *iUntil* **and**
-iSince ⇌ *iSince*
translations
P. t $\mathcal{U}$ *t′ I. Q* ⇌ *CONST iUntil I* (*λt. P*) (*λt′. Q*)
P. t $\mathcal{S}$ *t′ I. Q* ⇌ *CONST iSince I* (*λt. P*) (*λt′. Q*)

definition *iWeakUntil* :: *iT* ⇒ (*Time* ⇒ *bool*) ⇒ (*Time* ⇒ *bool*) ⇒ *bool* — Weak Until/Wating-for/Unless
where *iWeakUntil I P Q* ≡ (□ *t I. P t*) ∨ (◇ *t I. Q t* ∧ (□ *t′* (*I* ↓< *t*). *P t′*))

definition *iWeakSince* :: *iT* ⇒ (*Time* ⇒ *bool*) ⇒ (*Time* ⇒ *bool*) ⇒ *bool* — Weak Since/Back-to
where *iWeakSince I P Q* ≡ (□ *t I. P t*) ∨ (◇ *t I. Q t* ∧ (□ *t′* (*I* ↓> *t*). *P t′*))

syntax
-iWeakUntil :: *Time* ⇒ *Time* ⇒ *iT* ⇒ (*Time* ⇒ *bool*) ⇒ (*Time* ⇒ *bool*) ⇒ *bool*

(‹(-./ - (*3*$\mathcal{W}$ - -)./ -)› [*10, 0, 0, 0, 10*] *10*)
-iWeakSince :: *Time* ⇒ *Time* ⇒ *iT* ⇒ (*Time* ⇒ *bool*) ⇒ (*Time* ⇒ *bool*) ⇒ *bool*

(‹(-./ - (*3*$\mathcal{B}$ - -)./ -)› [*10, 0, 0, 0, 10*] *10*)
syntax-consts

$-iWeakUntil \rightleftharpoons iWeakUntil$ **and**
$-iWeakSince \rightleftharpoons iWeakSince$
translations
$P.\ t\ \mathcal{W}\ t'\ I.\ Q \rightleftharpoons CONST\ iWeakUntil\ I\ (\lambda t.\ P)\ (\lambda t'.\ Q)$
$P.\ t\ \mathcal{B}\ t'\ I.\ Q \rightleftharpoons CONST\ iWeakSince\ I\ (\lambda t.\ P)\ (\lambda t'.\ Q)$

definition $iRelease :: iT \Rightarrow (Time \Rightarrow bool) \Rightarrow (Time \Rightarrow bool) \Rightarrow bool$ — Release
where $iRelease\ I\ P\ Q \equiv (\Box\ t\ I.\ Q\ t) \vee (\Diamond\ t\ I.\ P\ t \wedge (\Box\ t'\ (I \downarrow\leq t).\ Q\ t'))$

definition $iTrigger :: iT \Rightarrow (Time \Rightarrow bool) \Rightarrow (Time \Rightarrow bool) \Rightarrow bool$ — Trigger
where $iTrigger\ I\ P\ Q \equiv (\Box\ t\ I.\ Q\ t) \vee (\Diamond\ t\ I.\ P\ t \wedge (\Box\ t'\ (I \downarrow\geq t).\ Q\ t'))$

syntax
$-iRelease :: Time \Rightarrow Time \Rightarrow iT \Rightarrow (Time \Rightarrow bool) \Rightarrow (Time \Rightarrow bool) \Rightarrow bool$
(‹(-./ - (3$\mathcal{R}$ - -)./ -)› [10, 0, 0, 0, 10] 10)
$-iTrigger :: Time \Rightarrow Time \Rightarrow iT \Rightarrow (Time \Rightarrow bool) \Rightarrow (Time \Rightarrow bool) \Rightarrow bool$
(‹(-./ - (3$\mathcal{T}$ - -)./ -)› [10, 0, 0, 0, 10] 10)
syntax-consts
$-iRelease \rightleftharpoons iRelease$ **and**
$-iTrigger \rightleftharpoons iTrigger$
translations
$P.\ t\ \mathcal{R}\ t'\ I.\ Q \rightleftharpoons CONST\ iRelease\ I\ (\lambda t.\ P)\ (\lambda t'.\ Q)$
$P.\ t\ \mathcal{T}\ t'\ I.\ Q \rightleftharpoons CONST\ iTrigger\ I\ (\lambda t.\ P)\ (\lambda t'.\ Q)$

lemmas *iTL-Next-defs* =
iNext-def iLast-def
iNextWeak-def iLastWeak-def
iNextStrong-def iLastStrong-def
lemmas *iTL-defs* =
iAll-def iEx-def
iUntil-def iSince-def
iWeakUntil-def iWeakSince-def
iRelease-def iTrigger-def
iTL-Next-defs

⟨*ML*⟩

3.2 Basic lemmata for temporal operators

3.2.1 Intro/elim rules

lemma
iexI[*intro*]: $[\![\ P\ t;\ t \in I\ ]\!] \Longrightarrow \Diamond\ t\ I.\ P\ t$ **and**
rev-iexI[*intro?*]: $[\![\ t \in I;\ P\ t\ ]\!] \Longrightarrow \Diamond\ t\ I.\ P\ t$ **and**
iexE[*elim!*]: $[\![\ \Diamond\ t\ I.\ P\ t;\ \bigwedge t.\ [\![\ t \in I;\ P\ t\ ]\!] \Longrightarrow Q\ ]\!] \Longrightarrow Q$

⟨*proof*⟩

lemma
iallI[*intro*!]: $(\bigwedge t.\ t \in I \Longrightarrow P\ t) \Longrightarrow \Box\ t\ I.\ P\ t$ **and**
ispec[*dest?*]: $[\![\Box\ t\ I.\ P\ t;\ t \in I]\!] \Longrightarrow P\ t$ **and**
iallE[*elim*]: $[\![\Box\ t\ I.\ P\ t;\ P\ t \Longrightarrow Q;\ t \notin I \Longrightarrow Q]\!] \Longrightarrow Q$
⟨*proof*⟩

lemma
iuntilI[*intro*]:
$[\![Q\ t;\ (\bigwedge t'.\ t' \in I \downarrow< t \Longrightarrow P\ t');\ t \in I]\!] \Longrightarrow P\ t'.\ t'\ \mathcal{U}\ t\ I.\ Q\ t$ **and**
rev-iuntilI[*intro?*]:
$[\![t \in I;\ Q\ t;\ (\bigwedge t'.\ t' \in I \downarrow< t \Longrightarrow P\ t')]\!] \Longrightarrow P\ t'.\ t'\ \mathcal{U}\ t\ I.\ Q\ t$
⟨*proof*⟩

lemma
iuntilE[*elim*]:
$[\![P'\ t'.\ t'\ \mathcal{U}\ t\ I.\ P\ t;\ \bigwedge t.\ [\![t \in I;\ P\ t]\!] \Longrightarrow Q]\!] \Longrightarrow Q$
⟨*proof*⟩

lemma
isinceI[*intro*]:
$[\![Q\ t;\ (\bigwedge t'.\ t' \in I \downarrow> t \Longrightarrow P\ t');\ t \in I]\!] \Longrightarrow P\ t'.\ t'\ \mathcal{S}\ t\ I.\ Q\ t$ **and**
rev-isinceI[*intro?*]:
$[\![t \in I;\ Q\ t;\ (\bigwedge t'.\ t' \in I \downarrow> t \Longrightarrow P\ t')]\!] \Longrightarrow P\ t'.\ t'\ \mathcal{S}\ t\ I.\ Q\ t$
⟨*proof*⟩
lemma
isinceE[*elim*]:
$[\![P'\ t'.\ t'\ \mathcal{S}\ t\ I.\ P\ t;\ \bigwedge t.\ [\![t \in I;\ P\ t]\!] \Longrightarrow Q]\!] \Longrightarrow Q$
⟨*proof*⟩

3.2.2 Rewrite rules for trivial simplification

lemma *iall-triv*[*simp*]: $(\Box\ t\ I.\ P) = ((\exists\, t.\ t \in I) \longrightarrow P)$
⟨*proof*⟩

lemma *iex-triv*[*simp*]: $(\Diamond\ t\ I.\ P) = ((\exists\, t.\ t \in I) \wedge P)$
⟨*proof*⟩

lemma *iex-conjL1*:
$(\Diamond\ t1\ I1.\ (P1\ t1 \wedge (\Diamond\ t2\ I2.\ P2\ t1\ t2))) =$
$(\Diamond\ t1\ I1.\ \Diamond\ t2\ I2.\ P1\ t1 \wedge P2\ t1\ t2)$
⟨*proof*⟩

lemma *iex-conjR1*:
$(\Diamond\ t1\ I1.\ ((\Diamond\ t2\ I2.\ P2\ t1\ t2) \wedge P1\ t1)) =$
$(\Diamond\ t1\ I1.\ \Diamond\ t2\ I2.\ P2\ t1\ t2 \wedge P1\ t1)$
⟨*proof*⟩

lemma *iex-conjL2*:
$(\Diamond\ t1\ I1.\ (P1\ t1 \wedge (\Diamond\ t2\ (I2\ t1).\ P2\ t1\ t2))) =$
$(\Diamond\ t1\ I1.\ \Diamond\ t2\ (I2\ t1).\ P1\ t1 \wedge P2\ t1\ t2)$
⟨*proof*⟩

lemma *iex-conjR2*:
$(\Diamond\ t1\ I1.\ ((\Diamond\ t2\ (I2\ t1).\ P2\ t1\ t2) \wedge P1\ t1)) =$
$(\Diamond\ t1\ I1.\ \Diamond\ t2\ (I2\ t1).\ P2\ t1\ t2 \wedge P1\ t1)$
⟨*proof*⟩

lemma *iex-commute*:
$(\Diamond\ t1\ I1.\ \Diamond\ t2\ I2.\ P\ t1\ t2) =$
$(\Diamond\ t2\ I2.\ \Diamond\ t1\ I1.\ P\ t1\ t2)$
⟨*proof*⟩

lemma *iall-conjL1*:
$I2 \neq \{\} \Longrightarrow$
$(\Box\ t1\ I1.\ (P1\ t1 \wedge (\Box\ t2\ I2.\ P2\ t1\ t2))) =$
$(\Box\ t1\ I1.\ \Box\ t2\ I2.\ P1\ t1 \wedge P2\ t1\ t2)$
⟨*proof*⟩

lemma *iall-conjR1*:
$I2 \neq \{\} \Longrightarrow$
$(\Box\ t1\ I1.\ ((\Box\ t2\ I2.\ P2\ t1\ t2) \wedge P1\ t1)) =$
$(\Box\ t1\ I1.\ \Box\ t2\ I2.\ P2\ t1\ t2 \wedge P1\ t1)$
⟨*proof*⟩

lemma *iall-conjL2*:
$\Box\ t1\ I1.\ I2\ t1 \neq \{\} \Longrightarrow$
$(\Box\ t1\ I1.\ (P1\ t1 \wedge (\Box\ t2\ (I2\ t1).\ P2\ t1\ t2))) =$
$(\Box\ t1\ I1.\ \Box\ t2\ (I2\ t1).\ P1\ t1 \wedge P2\ t1\ t2)$
⟨*proof*⟩

lemma *iall-conjR2*:
$\Box\ t1\ I1.\ I2\ t1 \neq \{\} \Longrightarrow$
$(\Box\ t1\ I1.\ ((\Box\ t2\ (I2\ t1).\ P2\ t1\ t2) \wedge P1\ t1)) =$
$(\Box\ t1\ I1.\ \Box\ t2\ (I2\ t1).\ P2\ t1\ t2 \wedge P1\ t1)$
⟨*proof*⟩

lemma *iall-commute*:
$(\Box\ t1\ I1.\ \Box\ t2\ I2.\ P\ t1\ t2) =$
$(\Box\ t2\ I2.\ \Box\ t1\ I1.\ P\ t1\ t2)$
⟨*proof*⟩

lemma *iall-conj-distrib*:
$(\Box\ t\ I.\ P\ t \wedge Q\ t) = ((\Box\ t\ I.\ P\ t) \wedge (\Box\ t\ I.\ Q\ t))$
⟨*proof*⟩

lemma *iex-disj-distrib*:

$(\Diamond\ t\ I.\ P\ t \vee Q\ t) = ((\Diamond\ t\ I.\ P\ t) \vee (\Diamond\ t\ I.\ Q\ t))$
⟨*proof*⟩

lemma *iall-conj-distrib2*:
$(\Box\ t1\ I1.\ \Box\ t2\ (I2\ t1).\ P\ t1\ t2 \wedge Q\ t1\ t2) =$
$((\Box\ t1\ I1.\ \Box\ t2\ (I2\ t1).\ P\ t1\ t2) \wedge (\Box\ t1\ I1.\ \Box\ t2\ (I2\ t1).\ Q\ t1\ t2))$
⟨*proof*⟩

lemma *iex-disj-distrib2*:
$(\Diamond\ t1\ I1.\ \Diamond\ t2\ (I2\ t1).\ P\ t1\ t2 \vee Q\ t1\ t2) =$
$((\Diamond\ t1\ I1.\ \Diamond\ t2\ (I2\ t1).\ P\ t1\ t2) \vee (\Diamond\ t1\ I1.\ \Diamond\ t2\ (I2\ t1).\ Q\ t1\ t2))$
⟨*proof*⟩

lemma *iUntil-disj-distrib*:
$(P\ t1.\ t1\ \mathcal{U}\ t2\ I.\ (Q1\ t2 \vee Q2\ t2)) = ((P\ t1.\ t1\ \mathcal{U}\ t2\ I.\ Q1\ t2) \vee (P\ t1.\ t1\ \mathcal{U}\ t2\ I.\ Q2\ t2))$
⟨*proof*⟩

lemma *iSince-disj-distrib*:
$(P\ t1.\ t1\ \mathcal{S}\ t2\ I.\ (Q1\ t2 \vee Q2\ t2)) = ((P\ t1.\ t1\ \mathcal{S}\ t2\ I.\ Q1\ t2) \vee (P\ t1.\ t1\ \mathcal{S}\ t2\ I.\ Q2\ t2))$
⟨*proof*⟩

lemma
iNext-iff: $(\bigcirc\ t\ t0\ I.\ P\ t) = (\Box\ t\ [\ldots 0] \oplus (inext\ t0\ I).\ P\ t)$ **and**
iLast-iff: $(\ominus\ t\ t0\ I.\ P\ t) = (\Box\ t\ [\ldots 0] \oplus (iprev\ t0\ I).\ P\ t)$
⟨*proof*⟩

lemma
iNext-iEx-iff: $(\bigcirc\ t\ t0\ I.\ P\ t) = (\Diamond\ t\ [\ldots 0] \oplus (inext\ t0\ I).\ P\ t)$ **and**
iLast-iEx-iff: $(\ominus\ t\ t0\ I.\ P\ t) = (\Diamond\ t\ [\ldots 0] \oplus (iprev\ t0\ I).\ P\ t)$
⟨*proof*⟩

lemma *inext-singleton-cut-greater-not-empty-iff*:
$(\{inext\ t0\ I\} \downarrow> t0 \neq \{\}) = (I \downarrow> t0 \neq \{\} \wedge t0 \in I)$
⟨*proof*⟩

lemma *iprev-singleton-cut-less-not-empty-iff*:
$(\{iprev\ t0\ I\} \downarrow< t0 \neq \{\}) = (I \downarrow< t0 \neq \{\} \wedge t0 \in I)$
⟨*proof*⟩

lemma *inext-singleton-cut-greater-empty-iff*:
$(\{inext\ t0\ I\} \downarrow> t0 = \{\}) = (I \downarrow> t0 = \{\} \vee t0 \notin I)$
⟨*proof*⟩

lemma *iprev-singleton-cut-less-empty-iff*:
$(\{iprev\ t0\ I\} \downarrow< t0 = \{\}) = (I \downarrow< t0 = \{\} \vee t0 \notin I)$
⟨*proof*⟩

lemma *iNextWeak-iff* : $(\bigcirc_W\ t\ t0\ I.\ P\ t) = ((\bigcirc\ t\ t0\ I.\ P\ t) \vee (I \downarrow> t0 = \{\}) \vee t0 \notin I)$
⟨*proof*⟩

lemma *iNextStrong-iff* : $(\bigcirc_S\ t\ t0\ I.\ P\ t) = ((\bigcirc\ t\ t0\ I.\ P\ t) \wedge (I \downarrow> t0 \neq \{\}) \wedge t0 \in I)$
⟨*proof*⟩

lemma *iLastWeak-iff* : $(\ominus_W\ t\ t0\ I.\ P\ t) = ((\ominus\ t\ t0\ I.\ P\ t) \vee (I \downarrow< t0 = \{\}) \vee t0 \notin I)$
⟨*proof*⟩

lemma *iLastStrong-iff* : $(\ominus_S\ t\ t0\ I.\ P\ t) = ((\ominus\ t\ t0\ I.\ P\ t) \wedge (I \downarrow< t0 \neq \{\}) \wedge t0 \in I)$
⟨*proof*⟩

lemmas *iTL-Next-iff* =
iNext-iff iLast-iff
iNextWeak-iff iNextStrong-iff
iLastWeak-iff iLastStrong-iff

lemma
iNext-iff-singleton : $(\bigcirc\ t\ t0\ I.\ P\ t) = (\Box\ t\ \{inext\ t0\ I\}.\ P\ t)$ **and**
iLast-iff-singleton : $(\ominus\ t\ t0\ I.\ P\ t) = (\Box\ t\ \{iprev\ t0\ I\}.\ P\ t)$
⟨*proof*⟩
lemmas *iNextLast-iff-singleton* = *iNext-iff-singleton iLast-iff-singleton*

lemma
iNext-iEx-iff-singleton : $(\bigcirc\ t\ t0\ I.\ P\ t) = (\Diamond\ t\ \{inext\ t0\ I\}.\ P\ t)$ **and**
iLast-iEx-iff-singleton : $(\ominus\ t\ t0\ I.\ P\ t) = (\Diamond\ t\ \{iprev\ t0\ I\}.\ P\ t)$
⟨*proof*⟩

lemma
iNextWeak-iAll-conv: $(\bigcirc_W\ t\ t0\ I.\ P\ t) = (\Box\ t\ (\{inext\ t0\ I\} \downarrow> t0).\ P\ t)$ **and**
iNextStrong-iEx-conv: $(\bigcirc_S\ t\ t0\ I.\ P\ t) = (\Diamond\ t\ (\{inext\ t0\ I\} \downarrow> t0).\ P\ t)$ **and**
iLastWeak-iAll-conv: $(\ominus_W\ t\ t0\ I.\ P\ t) = (\Box\ t\ (\{iprev\ t0\ I\} \downarrow< t0).\ P\ t)$ **and**
iLastStrong-iEx-conv: $(\ominus_S\ t\ t0\ I.\ P\ t) = (\Diamond\ t\ (\{iprev\ t0\ I\} \downarrow< t0).\ P\ t)$
⟨*proof*⟩

lemma
iAll-True[*simp*]: $\Box\ t\ I.\ True$ **and**
iAll-False[*simp*]: $(\Box\ t\ I.\ False) = (I = \{\})$ **and**
iEx-True[*simp*]: $(\Diamond\ t\ I.\ True) = (I \neq \{\})$ **and**
iEx-False[*simp*]: $\neg\ (\Diamond\ t\ I.\ False)$
⟨*proof*⟩

lemma *empty-iff-iAll-False*: $(I = \{\}) = (\Box\ t\ I.\ \mathit{False})$ ⟨*proof*⟩
lemma *not-empty-iff-iEx-True*: $(I \neq \{\}) = (\Diamond\ t\ I.\ \mathit{True})$ ⟨*proof*⟩

lemma
iNext-True: $\bigcirc\ t\ t0\ I.\ \mathit{True}$ **and**
iNextWeak-True: $(\bigcirc_W\ t\ t0\ I.\ \mathit{True})$ **and**
iNext-False: $\neg\ (\bigcirc\ t\ t0\ I.\ \mathit{False})$ **and**
iNextStrong-False: $\neg\ (\bigcirc_S\ t\ t0\ I.\ \mathit{False})$
⟨*proof*⟩

lemma
iNextStrong-True: $(\bigcirc_S\ t\ t0\ I.\ \mathit{True}) = (I \downarrow> t0 \neq \{\} \wedge t0 \in I)$ **and**
iNextWeak-False: $(\neg\ (\bigcirc_W\ t\ t0\ I.\ \mathit{False})) = (I \downarrow> t0 \neq \{\} \wedge t0 \in I)$
⟨*proof*⟩

lemma
iLast-True: $\ominus\ t\ t0\ I.\ \mathit{True}$ **and**
iLastWeak-True: $(\ominus_W\ t\ t0\ I.\ \mathit{True})$ **and**
iLast-False: $\neg\ (\ominus\ t\ t0\ I.\ \mathit{False})$ **and**
iLastStrong-False: $\neg\ (\ominus_S\ t\ t0\ I.\ \mathit{False})$
⟨*proof*⟩

lemma
iLastStrong-True: $(\ominus_S\ t\ t0\ I.\ \mathit{True}) = (I \downarrow< t0 \neq \{\} \wedge t0 \in I)$ **and**
iLastWeak-False: $(\neg\ (\ominus_W\ t\ t0\ I.\ \mathit{False})) = (I \downarrow< t0 \neq \{\} \wedge t0 \in I)$
⟨*proof*⟩

lemma *iUntil-True-left*[*simp*]: $(\mathit{True}.\ t'\ \mathcal{U}\ t\ I.\ Q\ t) = (\Diamond\ t\ I.\ Q\ t)$
⟨*proof*⟩

lemma *iUntil-True*[*simp*]: $(P\ t'.\ t'\ \mathcal{U}\ t\ I.\ \mathit{True}) = (I \neq \{\})$
⟨*proof*⟩

lemma *iUntil-False-left*[*simp*]: $(\mathit{False}.\ t'\ \mathcal{U}\ t\ I.\ Q\ t) = (I \neq \{\} \wedge Q\ (\mathit{iMin}\ I))$
⟨*proof*⟩

lemma *iUntil-False*[*simp*]: $\neg\ (P\ t'.\ t'\ \mathcal{U}\ t\ I.\ \mathit{False})$
⟨*proof*⟩

lemma *iSince-True-left*[*simp*]: $(\mathit{True}.\ t'\ \mathcal{S}\ t\ I.\ Q\ t) = (\Diamond\ t\ I.\ Q\ t)$
⟨*proof*⟩

lemma *iSince-True-if*:
$(P\ t'.\ t'\ \mathcal{S}\ t\ I.\ \mathit{True}) =$
$(\mathit{if\ finite}\ I\ \mathit{then}\ I \neq \{\}\ \mathit{else}\ \Diamond\ t1\ I.\ \Box\ t2\ (I \downarrow> t1).\ P\ t2)$
⟨*proof*⟩

corollary *iSince-True-finite*[*simp*]: *finite* $I \Longrightarrow (P\ t'.\ t'\ \mathcal{S}\ t\ I.\ True) = (I \neq \{\})$
⟨*proof*⟩

lemma *iSince-False-left*[*simp*]: $(False.\ t'\ \mathcal{S}\ t\ I.\ Q\ t) = (finite\ I \wedge I \neq \{\} \wedge Q\ (Max\ I))$
⟨*proof*⟩

lemma *iSince-False*[*simp*]: $\neg\ (P\ t'.\ t'\ \mathcal{S}\ t\ I.\ False)$
⟨*proof*⟩

lemma *iWeakUntil-True-left*[*simp*]: $True.\ t'\ \mathcal{W}\ t\ I.\ Q\ t$
⟨*proof*⟩

lemma *iWeakUntil-True*[*simp*]: $P\ t'.\ t'\ \mathcal{W}\ t\ I.\ True$
⟨*proof*⟩

lemma *iWeakUntil-False-left*[*simp*]: $(False.\ t'\ \mathcal{W}\ t\ I.\ Q\ t) = (I = \{\} \vee Q\ (iMin\ I))$
⟨*proof*⟩

lemma *iWeakUntil-False*[*simp*]: $(P\ t'.\ t'\ \mathcal{W}\ t\ I.\ False) = (\Box\ t\ I.\ P\ t)$
⟨*proof*⟩

lemma *iWeakSince-True-left*[*simp*]: $True.\ t'\ \mathcal{B}\ t\ I.\ Q\ t$
⟨*proof*⟩

lemma *iWeakSince-True-disj*:
$(P\ t'.\ t'\ \mathcal{B}\ t\ I.\ True) =$
$(I = \{\} \vee (\Diamond\ t1\ I.\ \Box\ t2\ (I \downarrow> t1).\ P\ t2))$
⟨*proof*⟩

lemma *iWeakSince-True-finite*[*simp*]: *finite* $I \Longrightarrow (P\ t'.\ t'\ \mathcal{B}\ t\ I.\ True)$
⟨*proof*⟩

lemma *iWeakSince-False-left*[*simp*]: $(False.\ t'\ \mathcal{B}\ t\ I.\ Q\ t) = (I = \{\} \vee (finite\ I \wedge Q\ (Max\ I)))$
⟨*proof*⟩

lemma *iWeakSince-False*[*simp*]: $(P\ t'.\ t'\ \mathcal{B}\ t\ I.\ False) = (\Box\ t\ I.\ P\ t)$
⟨*proof*⟩

lemma *iRelease-True-left*[*simp*]: $(True.\ t'\ \mathcal{R}\ t\ I.\ Q\ t) = (I = \{\} \vee Q\ (iMin\ I))$
⟨*proof*⟩

lemma *iRelease-True*[*simp*]: $P\ t'.\ t'\ \mathcal{R}\ t\ I.\ True$
⟨*proof*⟩

lemma *iRelease-False-left*[*simp*]: $(False.\ t'\ \mathcal{R}\ t\ I.\ Q\ t) = (\Box\ t\ I.\ Q\ t)$
⟨*proof*⟩

lemma *iRelease-False*[*simp*]: $(P\ t'.\ t'\ \mathcal{R}\ t\ I.\ False) = (I = \{\})$
⟨*proof*⟩

lemma *iTrigger-True-left*[*simp*]: $(True.\ t'\ \mathcal{T}\ t\ I.\ Q\ t) = (I = \{\} \vee (\Diamond\ t1\ I.\ \Box\ t2\ (I \downarrow\geq t1).\ Q\ t2))$
⟨*proof*⟩

lemma *iTrigger-True*[*simp*]: $P\ t'.\ t'\ \mathcal{T}\ t\ I.\ True$
⟨*proof*⟩

lemma *iTrigger-False-left*[*simp*]: $(False.\ t'\ \mathcal{T}\ t\ I.\ Q\ t) = (\Box\ t\ I.\ Q\ t)$
⟨*proof*⟩

lemma *iTrigger-False*[*simp*]: $(P\ t'.\ t'\ \mathcal{T}\ t\ I.\ False) = (I = \{\})$
⟨*proof*⟩

lemma
iUntil-TrueTrue[*simp*]: $(True.\ t'\ \mathcal{U}\ t\ I.\ True) = (I \neq \{\})$ **and**
iSince-TrueTrue[*simp*]: $(True.\ t'\ \mathcal{S}\ t\ I.\ True) = (I \neq \{\})$ **and**
iWeakUntil-TrueTrue[*simp*]: $True.\ t'\ \mathcal{W}\ t\ I.\ True$ **and**
iWeakSince-TrueTrue[*simp*]: $True.\ t'\ \mathcal{B}\ t\ I.\ True$ **and**
iRelease-TrueTrue[*simp*]: $True.\ t'\ \mathcal{R}\ t\ I.\ True$ **and**
iTrigger-TrueTrue[*simp*]: $True.\ t'\ \mathcal{T}\ t\ I.\ True$
⟨*proof*⟩

3.2.3 Empty sets and singletons

lemma *iAll-empty*[*simp*]: $\Box\ t\ \{\}.\ P\ t$ ⟨*proof*⟩
lemma *iEx-empty*[*simp*]: $\neg\ (\Diamond\ t\ \{\}.\ P\ t)$ ⟨*proof*⟩

lemma *iUntil-empty*[*simp*]: $\neg\ (P\ t0.\ t0\ \mathcal{U}\ t1\ \{\}.\ Q\ t1)$ ⟨*proof*⟩
lemma *iSince-empty*[*simp*]: $\neg\ (P\ t0.\ t0\ \mathcal{S}\ t1\ \{\}.\ Q\ t1)$ ⟨*proof*⟩
lemma *iWeakUntil-empty*[*simp*]: $P\ t0.\ t0\ \mathcal{W}\ t1\ \{\}.\ Q\ t1$ ⟨*proof*⟩
lemma *iWeakSince-empty*[*simp*]: $P\ t0.\ t0\ \mathcal{B}\ t1\ \{\}.\ Q\ t1$ ⟨*proof*⟩

lemma *iRelease-empty*[*simp*]: $P\ t0.\ t0\ \mathcal{R}\ t1\ \{\}.\ Q\ t1$ ⟨*proof*⟩
lemma *iTrigger-empty*[*simp*]: $P\ t0.\ t0\ \mathcal{T}\ t1\ \{\}.\ Q\ t1$ ⟨*proof*⟩

lemmas *iTL-empty* =
iAll-empty iEx-empty
iUntil-empty iSince-empty
iWeakUntil-empty iWeakSince-empty
iRelease-empty iTrigger-empty

lemma *iAll-singleton*[*simp*]: $(\Box\ t'\ \{t\}.\ P\ t') = P\ t$ ⟨*proof*⟩
lemma *iEx-singleton*[*simp*]: $(\Diamond\ t'\ \{t\}.\ P\ t') = P\ t$ ⟨*proof*⟩

lemma *iUntil-singleton*[*simp*]: $(P\ t0.\ t0\ \mathcal{U}\ t1\ \{t\}.\ Q\ t1) = Q\ t$

⟨*proof*⟩

lemma *iSince-singleton*[*simp*]: $(P\ t0.\ t0\ \mathcal{S}\ t1\ \{t\}.\ Q\ t1) = Q\ t$
⟨*proof*⟩

lemma *iWeakUntil-singleton*[*simp*]: $(P\ t0.\ t0\ \mathcal{W}\ t1\ \{t\}.\ Q\ t1) = (P\ t \lor Q\ t)$
⟨*proof*⟩

lemma *iWeakSince-singleton*[*simp*]: $(P\ t0.\ t0\ \mathcal{B}\ t1\ \{t\}.\ Q\ t1) = (P\ t \lor Q\ t)$
⟨*proof*⟩

lemma *iRelease-singleton*[*simp*]: $(P\ t0.\ t0\ \mathcal{R}\ t1\ \{t\}.\ Q\ t1) = Q\ t$
⟨*proof*⟩

lemma *iTrigger-singleton*[*simp*]: $(P\ t0.\ t0\ \mathcal{T}\ t1\ \{t\}.\ Q\ t1) = Q\ t$
⟨*proof*⟩

lemmas *iTL-singleton* =
iAll-singleton iEx-singleton
iUntil-singleton iSince-singleton
iWeakUntil-singleton iWeakSince-singleton
iRelease-singleton iTrigger-singleton

3.2.4 Conversions between temporal operators

lemma *iAll-iEx-conv*: $(\Box\ t\ I.\ P\ t) = (\neg\ (\Diamond\ t\ I.\ \neg\ P\ t))$ ⟨*proof*⟩
lemma *iEx-iAll-conv*: $(\Diamond\ t\ I.\ P\ t) = (\neg\ (\Box\ t\ I.\ \neg\ P\ t))$ ⟨*proof*⟩

lemma *not-iAll*[*simp*]: $(\neg\ (\Box\ t\ I.\ P\ t)) = (\Diamond\ t\ I.\ \neg\ P\ t)$ ⟨*proof*⟩
lemma *not-iEx*[*simp*]: $(\neg\ (\Diamond\ t\ I.\ P\ t)) = (\Box\ t\ I.\ \neg\ P\ t)$ ⟨*proof*⟩

lemma *iUntil-iEx-conv*: $(True.\ t'\ \mathcal{U}\ t\ I.\ P\ t) = (\Diamond\ t\ I.\ P\ t)$ ⟨*proof*⟩
lemma *iSince-iEx-conv*: $(True.\ t'\ \mathcal{S}\ t\ I.\ P\ t) = (\Diamond\ t\ I.\ P\ t)$ ⟨*proof*⟩

lemma *iRelease-iAll-conv*: $(False.\ t'\ \mathcal{R}\ t\ I.\ P\ t) = (\Box\ t\ I.\ P\ t)$
⟨*proof*⟩

lemma *iTrigger-iAll-conv*: $(False.\ t'\ \mathcal{T}\ t\ I.\ P\ t) = (\Box\ t\ I.\ P\ t)$
⟨*proof*⟩

lemma *iWeakUntil-iUntil-conv*: $(P\ t'.\ t'\ \mathcal{W}\ t\ I.\ Q\ t) = ((P\ t'.\ t'\ \mathcal{U}\ t\ I.\ Q\ t) \lor (\Box\ t\ I.\ P\ t))$
⟨*proof*⟩

lemma *iWeakSince-iSince-conv*: $(P\ t'.\ t'\ \mathcal{B}\ t\ I.\ Q\ t) = ((P\ t'.\ t'\ \mathcal{S}\ t\ I.\ Q\ t) \lor (\Box\ t\ I.\ P\ t))$
⟨*proof*⟩

lemma *iUntil-iWeakUntil-conv*: $(P\ t'.\ t'\ \mathcal{U}\ t\ I.\ Q\ t) = ((P\ t'.\ t'\ \mathcal{W}\ t\ I.\ Q\ t) \land (\Diamond$

$t\ I.\ Q\ t))$
$\langle proof \rangle$

lemma *iSince-iWeakSince-conv*: $(P\ t'.\ t'\ \mathcal{S}\ t\ I.\ Q\ t) = ((P\ t'.\ t'\ \mathcal{B}\ t\ I.\ Q\ t) \wedge (\Diamond\ t\ I.\ Q\ t))$
$\langle proof \rangle$

lemma *iRelease-iWeakUntil-conv*: $(P\ t'.\ t'\ \mathcal{R}\ t\ I.\ Q\ t) = (Q\ t'.\ t'\ \mathcal{W}\ t\ I.\ (Q\ t \wedge P\ t))$
$\langle proof \rangle$

lemma *iRelease-iUntil-conv*: $(P\ t'.\ t'\ \mathcal{R}\ t\ I.\ Q\ t) = ((\Box\ t\ I.\ Q\ t) \vee (Q\ t'.\ t'\ \mathcal{U}\ t\ I.\ (Q\ t \wedge P\ t)))$
$\langle proof \rangle$

lemma *iTrigger-iWeakSince-conv*: $(P\ t'.\ t'\ \mathcal{T}\ t\ I.\ Q\ t) = (Q\ t'.\ t'\ \mathcal{B}\ t\ I.\ (Q\ t \wedge P\ t))$
$\langle proof \rangle$

lemma *iTrigger-iSince-conv*: $(P\ t'.\ t'\ \mathcal{T}\ t\ I.\ Q\ t) = ((\Box\ t\ I.\ Q\ t) \vee (Q\ t'.\ t'\ \mathcal{S}\ t\ I.\ (Q\ t \wedge P\ t)))$
$\langle proof \rangle$

lemma *iRelease-not-iUntil-conv*: $(P\ t'.\ t'\ \mathcal{R}\ t\ I.\ Q\ t) = (\neg\ (\neg\ P\ t'.\ t'\ \mathcal{U}\ t\ I.\ \neg\ Q\ t))$
$\langle proof \rangle$
lemma *iUntil-not-iRelease-conv*: $(P\ t'.\ t'\ \mathcal{U}\ t\ I.\ Q\ t) = (\neg\ (\neg\ P\ t'.\ t'\ \mathcal{R}\ t\ I.\ \neg\ Q\ t))$
$\langle proof \rangle$

The Trigger operator $\mathcal{T}$is a past operator, so that it is used for time intervals, that are bounded by a current time point, and thus are finite. For an infinite interval the stated relation to the Since operator $\mathcal{S}$would not be fulfilled.

lemma *iTrigger-not-iSince-conv*: *finite* $I \Longrightarrow (P\ t'.\ t'\ \mathcal{T}\ t\ I.\ Q\ t) = (\neg\ (\neg\ P\ t'.\ t'\ \mathcal{S}\ t\ I.\ \neg\ Q\ t))$
$\langle proof \rangle$

lemma *iSince-not-iTrigger-conv*: *finite* $I \Longrightarrow (P\ t'.\ t'\ \mathcal{S}\ t\ I.\ Q\ t) = (\neg\ (\neg\ P\ t'.\ t'\ \mathcal{T}\ t\ I.\ \neg\ Q\ t))$
$\langle proof \rangle$

lemma *not-iUntil*:
$(\neg\ (P\ t1.\ t1\ \mathcal{U}\ t2\ I.\ Q\ t2)) =$
$(\Box\ t\ I.\ (Q\ t \longrightarrow (\Diamond\ t'\ (I \downarrow< t).\ \neg\ P\ t')))$
$\langle proof \rangle$

lemma *not-iSince*:

$(\neg\ (P\ t1.\ t1\ \mathcal{S}\ t2\ I.\ Q\ t2)) =$
$(\Box\ t\ I.\ (Q\ t \longrightarrow (\Diamond\ t'\ (I \downarrow> t).\ \neg\ P\ t')))$
⟨*proof*⟩

lemma *iWeakUntil-conj-iUntil-conv*:
$(P\ t1.\ t1\ \mathcal{W}\ t2\ I.\ (P\ t2 \wedge Q\ t2)) = (\neg\ (\neg\ Q\ t1.\ t1\ \mathcal{U}\ t2\ I.\ \neg\ P\ t2))$
⟨*proof*⟩

lemma *iUntil-disj-iUntil-conv*:
$(P\ t1 \vee Q\ t1.\ t1\ \mathcal{U}\ t2\ I.\ Q\ t2) =$
$(P\ t1.\ t1\ \mathcal{U}\ t2\ I.\ Q\ t2)$
⟨*proof*⟩

lemma *iWeakUntil-disj-iWeakUntil-conv*:
$(P\ t1 \vee Q\ t1.\ t1\ \mathcal{W}\ t2\ I.\ Q\ t2) =$
$(P\ t1.\ t1\ \mathcal{W}\ t2\ I.\ Q\ t2)$
⟨*proof*⟩

lemma *iWeakUntil-iUntil-conj-conv*:
$(P\ t1.\ t1\ \mathcal{W}\ t2\ I.\ Q\ t2) =$
$(\neg\ (\neg\ Q\ t1.\ t1\ \mathcal{U}\ t2\ I.\ (\neg\ P\ t2 \wedge \neg\ Q\ t2)))$
⟨*proof*⟩

Negation and temporal operators

lemma
not-iNext[*simp*]: $(\neg\ (\bigcirc\ t\ t0\ I.\ P\ t)) = (\bigcirc\ t\ t0\ I.\ \neg\ P\ t)$ **and**
not-iNextWeak[*simp*]: $(\neg\ (\bigcirc_W\ t\ t0\ I.\ P\ t)) = (\bigcirc_S\ t\ t0\ I.\ \neg\ P\ t)$ **and**
not-iNextStrong[*simp*]: $(\neg\ (\bigcirc_S\ t\ t0\ I.\ P\ t)) = (\bigcirc_W\ t\ t0\ I.\ \neg\ P\ t)$ **and**
not-iLast[*simp*]: $(\neg\ (\ominus\ t\ t0\ I.\ P\ t)) = (\ominus\ t\ t0\ I.\ \neg\ P\ t)$ **and**
not-iLastWeak[*simp*]: $(\neg\ (\ominus_W\ t\ t0\ I.\ P\ t)) = (\ominus_S\ t\ t0\ I.\ \neg\ P\ t)$ **and**
not-iLastStrong[*simp*]: $(\neg\ (\ominus_S\ t\ t0\ I.\ P\ t)) = (\ominus_W\ t\ t0\ I.\ \neg\ P\ t)$
⟨*proof*⟩

lemma *not-iWeakUntil*:
$(\neg\ (P\ t1.\ t1\ \mathcal{W}\ t2\ I.\ Q\ t2)) =$
$((\Box\ t\ I.\ (Q\ t \longrightarrow (\Diamond\ t'\ (I \downarrow< t).\ \neg\ P\ t'))) \wedge (\Diamond\ t\ I.\ \neg\ P\ t))$
⟨*proof*⟩
lemma *not-iWeakSince*:
$(\neg\ (P\ t1.\ t1\ \mathcal{B}\ t2\ I.\ Q\ t2)) =$
$((\Box\ t\ I.\ (Q\ t \longrightarrow (\Diamond\ t'\ (I \downarrow> t).\ \neg\ P\ t'))) \wedge (\Diamond\ t\ I.\ \neg\ P\ t))$
⟨*proof*⟩

lemma *not-iRelease*:
$(\neg\ (P\ t'.\ t'\ \mathcal{R}\ t\ I.\ Q\ t)) =$
$((\Diamond\ t\ I.\ \neg\ Q\ t) \wedge (\Box\ t\ I.\ P\ t \longrightarrow (\Diamond\ t\ I \downarrow\leq t.\ \neg\ Q\ t)))$
⟨*proof*⟩

lemma *not-iRelease-iUntil-conv*:

$(\neg\ (P\ t'.\ t'\ \mathcal{R}\ t\ I.\ Q\ t)) = (\neg\ P\ t'.\ t'\ \mathcal{U}\ t\ I.\ \neg\ Q\ t)$
⟨*proof*⟩

lemma *not-iTrigger*:
$(\neg\ (P\ t'.\ t'\ \mathcal{T}\ t\ I.\ Q\ t)) =$
$((\Diamond\ t\ I.\ \neg\ Q\ t) \wedge (\Box\ t\ I.\ \neg\ P\ t \vee (\Diamond\ t\ I \downarrow\geq t.\ \neg\ Q\ t)))$
⟨*proof*⟩

lemma *not-iTrigger-iSince-conv*:
finite $I \Longrightarrow (\neg\ (P\ t'.\ t'\ \mathcal{T}\ t\ I.\ Q\ t)) = (\neg\ P\ t'.\ t'\ \mathcal{S}\ t\ I.\ \neg\ Q\ t)$
⟨*proof*⟩

3.2.5 Some implication results

lemma *all-imp-iall*: $\forall x.\ P\ x \Longrightarrow \Box\ t\ I.\ P\ t$ ⟨*proof*⟩
lemma *bex-imp-lex*: $\Diamond\ t\ I.\ P\ t \Longrightarrow \exists x.\ P\ x$ ⟨*proof*⟩

lemma *iAll-imp-iEx*: $I \neq \{\} \Longrightarrow \Box\ t\ I.\ P\ t \Longrightarrow \Diamond\ t\ I.\ P\ t$ ⟨*proof*⟩
lemma *i-set-iAll-imp-iEx*: $I \in$ *i-set* $\Longrightarrow \Box\ t\ I.\ P\ t \Longrightarrow \Diamond\ t\ I.\ P\ t$
⟨*proof*⟩

lemmas *iT-iAll-imp-iEx* = *iT-not-empty*[*THEN iAll-imp-iEx*]

lemma *iUntil-imp-iEx*: $P\ t1.\ t1\ \mathcal{U}\ t2\ I.\ Q\ t2 \Longrightarrow \Diamond\ t\ I.\ Q\ t$
⟨*proof*⟩

lemma *iSince-imp-iEx*: $P\ t1.\ t1\ \mathcal{S}\ t2\ I.\ Q\ t2 \Longrightarrow \Diamond\ t\ I.\ Q\ t$
⟨*proof*⟩

lemma *iall-subset-imp-iall*: $[\![\ \Box\ t\ B.\ P\ t;\ A \subseteq B\ ]\!] \Longrightarrow \Box\ t\ A.\ P\ t$
⟨*proof*⟩

lemma *iex-subset-imp-iex*: $[\![\ \Diamond\ t\ A.\ P\ t;\ A \subseteq B\ ]\!] \Longrightarrow \Diamond\ t\ B.\ P\ t$
⟨*proof*⟩

lemma *iall-mp*: $[\![\ \Box\ t\ I.\ P\ t \longrightarrow Q\ t;\ \Box\ t\ I.\ P\ t\ ]\!] \Longrightarrow \Box\ t\ I.\ Q\ t$ ⟨*proof*⟩
lemma *iex-mp*: $[\![\ \Box\ t\ I.\ P\ t \longrightarrow Q\ t;\ \Diamond\ t\ I.\ P\ t\ ]\!] \Longrightarrow \Diamond\ t\ I.\ Q\ t$ ⟨*proof*⟩

lemma *iUntil-imp*:
$[\![\ P1\ t1.\ t1\ \mathcal{U}\ t2\ I.\ Q\ t2;\ \Box\ t\ I.\ P1\ t \longrightarrow P2\ t\ ]\!] \Longrightarrow P2\ t1.\ t1\ \mathcal{U}\ t2\ I.\ Q\ t2$
⟨*proof*⟩

lemma *iSince-imp*:
$[\![\ P1\ t1.\ t1\ \mathcal{S}\ t2\ I.\ Q\ t2;\ \Box\ t\ I.\ P1\ t \longrightarrow P2\ t\ ]\!] \Longrightarrow P2\ t1.\ t1\ \mathcal{S}\ t2\ I.\ Q\ t2$
⟨*proof*⟩

lemma *iWeakUntil-imp*:
$[\![\ P1\ t1.\ t1\ \mathcal{W}\ t2\ I.\ Q\ t2;\ \Box\ t\ I.\ P1\ t \longrightarrow P2\ t\ ]\!] \Longrightarrow P2\ t1.\ t1\ \mathcal{W}\ t2\ I.\ Q\ t2$

⟨*proof*⟩

lemma *iWeakSince-imp*:
$[\![\ P1\ t1.\ t1\ \mathcal{B}\ t2\ I.\ Q\ t2;\ \Box\ t\ I.\ P1\ t \longrightarrow P2\ t\ ]\!] \Longrightarrow P2\ t1.\ t1\ \mathcal{B}\ t2\ I.\ Q\ t2$
⟨*proof*⟩

lemma *iRelease-imp*:
$[\![\ P1\ t1.\ t1\ \mathcal{R}\ t2\ I.\ Q\ t2;\ \Box\ t\ I.\ P1\ t \longrightarrow P2\ t\ ]\!] \Longrightarrow P2\ t1.\ t1\ \mathcal{R}\ t2\ I.\ Q\ t2$
⟨*proof*⟩

lemma *iTrigger-imp*:
$[\![\ P1\ t1.\ t1\ \mathcal{T}\ t2\ I.\ Q\ t2;\ \Box\ t\ I.\ P1\ t \longrightarrow P2\ t\ ]\!] \Longrightarrow P2\ t1.\ t1\ \mathcal{T}\ t2\ I.\ Q\ t2$
⟨*proof*⟩

lemma *iMin-imp-iUntil*:
$[\![\ I \neq \{\};\ Q\ (iMin\ I)\ ]\!] \Longrightarrow P\ t'.\ t'\ \mathcal{U}\ t\ I.\ Q\ t$
⟨*proof*⟩

lemma *Max-imp-iSince*:
$[\![\ finite\ I;\ I \neq \{\};\ Q\ (Max\ I)\ ]\!] \Longrightarrow P\ t'.\ t'\ \mathcal{S}\ t\ I.\ Q\ t$
⟨*proof*⟩

3.2.6 Congruence rules for temporal operators' predicates

lemma *iAll-cong*: $\Box\ t\ I.\ f\ t = g\ t \Longrightarrow (\Box\ t\ I.\ P\ (f\ t)\ t) = (\Box\ t\ I.\ P\ (g\ t)\ t)$
⟨*proof*⟩

lemma *iEx-cong*: $\Box\ t\ I.\ f\ t = g\ t \Longrightarrow (\Diamond\ t\ I.\ P\ (f\ t)\ t) = (\Diamond\ t\ I.\ P\ (g\ t)\ t)$
⟨*proof*⟩

lemma *iUntil-cong1*:
$\Box\ t\ I.\ f\ t = g\ t \Longrightarrow$
$(P\ (f\ t1)\ t1.\ t1\ \mathcal{U}\ t2\ I.\ Q\ t2) = (P\ (g\ t1)\ t1.\ t1\ \mathcal{U}\ t2\ I.\ Q\ t2)$
⟨*proof*⟩

lemma *iUntil-cong2*:
$\Box\ t\ I.\ f\ t = g\ t \Longrightarrow$
$(P\ t1.\ t1\ \mathcal{U}\ t2\ I.\ Q\ (f\ t2)\ t2) = (P\ t1.\ t1\ \mathcal{U}\ t2\ I.\ Q\ (g\ t2)\ t2)$
⟨*proof*⟩

lemma *iSince-cong1*:
$\Box\ t\ I.\ f\ t = g\ t \Longrightarrow$
$(P\ (f\ t1)\ t1.\ t1\ \mathcal{S}\ t2\ I.\ Q\ t2) = (P\ (g\ t1)\ t1.\ t1\ \mathcal{S}\ t2\ I.\ Q\ t2)$
⟨*proof*⟩

lemma *iSince-cong2*:
$\Box\ t\ I.\ f\ t = g\ t \Longrightarrow$
$(P\ t1.\ t1\ \mathcal{S}\ t2\ I.\ Q\ (f\ t2)\ t2) = (P\ t1.\ t1\ \mathcal{S}\ t2\ I.\ Q\ (g\ t2)\ t2)$

⟨*proof*⟩

lemma *bex-subst*:
$\forall x \in A.\ P\ x \longrightarrow (Q\ x = Q'\ x) \Longrightarrow$
$(\exists x \in A.\ P\ x \wedge Q\ x) = (\exists x \in A.\ P\ x \wedge Q'\ x)$
⟨*proof*⟩

lemma *iEx-subst*:
$\Box\ t\ I.\ P\ t \longrightarrow (Q\ t = Q'\ t) \Longrightarrow$
$(\Diamond\ t\ I.\ P\ t \wedge Q\ t) = (\Diamond\ t\ I.\ P\ t \wedge Q'\ t)$
⟨*proof*⟩

3.2.7 Temporal operators with set unions/intersections and subsets

lemma *iAll-subset*: $[\![\ A \subseteq B;\ \Box\ t\ B.\ P\ t\ ]\!] \Longrightarrow \Box\ t\ A.\ P\ t$
⟨*proof*⟩

lemma *iEx-subset*: $[\![\ A \subseteq B;\ \Diamond\ t\ A.\ P\ t\ ]\!] \Longrightarrow \Diamond\ t\ B.\ P\ t$
⟨*proof*⟩

lemma *iUntil-append*:
$[\![\ \mathit{finite}\ A;\ \mathit{Max}\ A \le \mathit{iMin}\ B\ ]\!] \Longrightarrow$
$P\ t1.\ t1\ \mathcal{U}\ t2\ A.\ Q\ t2 \Longrightarrow P\ t1.\ t1\ \mathcal{U}\ t2\ (A \cup B).\ Q\ t2$
⟨*proof*⟩

lemma *iSince-prepend*:
$[\![\ \mathit{finite}\ A;\ \mathit{Max}\ A \le \mathit{iMin}\ B\ ]\!] \Longrightarrow$
$P\ t1.\ t1\ \mathcal{S}\ t2\ B.\ Q\ t2 \Longrightarrow P\ t1.\ t1\ \mathcal{S}\ t2\ (A \cup B).\ Q\ t2$
⟨*proof*⟩

lemma
iAll-union: $[\![\ \Box\ t\ A.\ P\ t;\ \Box\ t\ B.\ P\ t\ ]\!] \Longrightarrow \Box\ t\ (A \cup B).\ P\ t$ **and**
iAll-union-conv: $(\Box\ t\ A \cup B.\ P\ t) = ((\Box\ t\ A.\ P\ t) \wedge (\Box\ t\ B.\ P\ t))$
⟨*proof*⟩

lemma
iEx-union: $(\Diamond\ t\ A.\ P\ t) \vee (\Diamond\ t\ B.\ P\ t) \Longrightarrow \Diamond\ t\ (A \cup B).\ P\ t$ **and**
iEx-union-conv: $(\Diamond\ t\ (A \cup B).\ P\ t) = ((\Diamond\ t\ A.\ P\ t) \vee (\Diamond\ t\ B.\ P\ t))$
⟨*proof*⟩

lemma *iAll-inter*: $(\Box\ t\ A.\ P\ t) \vee (\Box\ t\ B.\ P\ t) \Longrightarrow \Box\ t\ (A \cap B).\ P\ t$ ⟨*proof*⟩

lemma *not-iEx-inter*:
$\exists A\ B\ P.\ (\Diamond\ t\ A.\ P\ t) \wedge (\Diamond\ t\ B.\ P\ t) \wedge \neg\ (\Diamond\ t\ (A \cap B).\ P\ t)$
⟨*proof*⟩

lemma

iAll-insert: $\llbracket\ P\ t;\ \Box\ t\ I.\ P\ t\ \rrbracket \Longrightarrow \Box\ t'\ (insert\ t\ I).\ P\ t'$ **and**
iAll-insert-conv: $(\Box\ t'\ (insert\ t\ I).\ P\ t') = (P\ t \wedge (\Box\ t'\ I.\ P\ t'))$
$\langle proof \rangle$

lemma
iEx-insert: $\llbracket\ P\ t \vee (\Diamond\ t\ I.\ P\ t)\ \rrbracket \Longrightarrow \Diamond\ t'\ (insert\ t\ I).\ P\ t'$ **and**
iEx-insert-conv: $(\Diamond\ t'\ (insert\ t\ I).\ P\ t') = (P\ t \vee (\Diamond\ t'\ I.\ P\ t'))$
$\langle proof \rangle$

3.3 Further results for temporal operators

lemma *Collect-minI-iEx*: $\Diamond\ t\ I.\ P\ t \Longrightarrow \Diamond\ t\ I.\ P\ t \wedge (\Box\ t'\ (I \downarrow< t).\ \neg\ P\ t')$
$\langle proof \rangle$

lemma *iUntil-disj-conv1*:
$I \neq \{\} \Longrightarrow$
$(P\ t'.\ t'\ \mathcal{U}\ t\ I.\ Q\ t) = (Q\ (iMin\ I) \vee (P\ t'.\ t'\ \mathcal{U}\ t\ I.\ Q\ t \wedge iMin\ I < t))$
$\langle proof \rangle$

lemma *iSince-disj-conv1*:
$\llbracket\ finite\ I;\ I \neq \{\}\ \rrbracket \Longrightarrow$
$(P\ t'.\ t'\ \mathcal{S}\ t\ I.\ Q\ t) = (Q\ (Max\ I) \vee (P\ t'.\ t'\ \mathcal{S}\ t\ I.\ Q\ t \wedge t < Max\ I))$
$\langle proof \rangle$

lemma *iUntil-next*:
$I \neq \{\} \Longrightarrow$
$(P\ t'.\ t'\ \mathcal{U}\ t\ I.\ Q\ t) =$
$(Q\ (iMin\ I) \vee (P\ (iMin\ I) \wedge (P\ t'.\ t'\ \mathcal{U}\ t\ (I \downarrow> (iMin\ I)).\ Q\ t)))$
$\langle proof \rangle$

lemma *iSince-prev*: $\llbracket\ finite\ I;\ I \neq \{\}\ \rrbracket \Longrightarrow$
$(P\ t'.\ t'\ \mathcal{S}\ t\ I.\ Q\ t) =$
$(Q\ (Max\ I) \vee (P\ (Max\ I) \wedge (P\ t'.\ t'\ \mathcal{S}\ t\ (I \downarrow< Max\ I).\ Q\ t)))$
$\langle proof \rangle$

lemma *iNext-induct-rule*:
$\llbracket\ P\ (iMin\ I);\ \Box\ t\ I.\ (P\ t \longrightarrow (\bigcirc\ t'\ t\ I.\ P\ t'));\ t \in I\ \rrbracket \Longrightarrow P\ t$
$\langle proof \rangle$

lemma *iNext-induct*:
$\llbracket\ P\ (iMin\ I);\ \Box\ t\ I.\ (P\ t \longrightarrow (\bigcirc\ t'\ t\ I.\ P\ t'))\ \rrbracket \Longrightarrow \Box\ t\ I.\ P\ t$
$\langle proof \rangle$

lemma *iLast-induct-rule*:
$\llbracket\ P\ (Max\ I);\ \Box\ t\ I.\ (P\ t \longrightarrow (\ominus\ t'\ t\ I.\ P\ t'));\ finite\ I;\ t \in I\ \rrbracket \Longrightarrow P\ t$
$\langle proof \rangle$

lemma *iLast-induct*:
$\llbracket\ P\ (Max\ I);\ \Box\ t\ I.\ (P\ t \longrightarrow (\ominus\ t'\ t\ I.\ P\ t'));\ finite\ I\ \rrbracket \Longrightarrow \Box\ t\ I.\ P\ t$

⟨*proof*⟩

lemma *iUntil-conj-not*: $((P\ t1 \wedge \neg\ Q\ t1).\ t1\ \mathcal{U}\ t2\ I.\ Q\ t2) = (P\ t1.\ t1\ \mathcal{U}\ t2\ I.\ Q\ t2)$
⟨*proof*⟩

lemma *iWeakUntil-conj-not*: $((P\ t1 \wedge \neg\ Q\ t1).\ t1\ \mathcal{W}\ t2\ I.\ Q\ t2) = (P\ t1.\ t1\ \mathcal{W}\ t2\ I.\ Q\ t2)$
⟨*proof*⟩

lemma *iSince-conj-not*: *finite* $I \Longrightarrow$
$((P\ t1 \wedge \neg\ Q\ t1).\ t1\ \mathcal{S}\ t2\ I.\ Q\ t2) = (P\ t1.\ t1\ \mathcal{S}\ t2\ I.\ Q\ t2)$
⟨*proof*⟩

lemma *iWeakSince-conj-not*: *finite* $I \Longrightarrow$
$((P\ t1 \wedge \neg\ Q\ t1).\ t1\ \mathcal{B}\ t2\ I.\ Q\ t2) = (P\ t1.\ t1\ \mathcal{B}\ t2\ I.\ Q\ t2)$
⟨*proof*⟩

lemma *iNextStrong-imp-iNextWeak*: $(\bigcirc_S\ t\ t0\ I.\ P\ t) \longrightarrow (\bigcirc_W\ t\ t0\ I.\ P\ t)$
⟨*proof*⟩
lemma *iLastStrong-imp-iLastWeak*: $(\ominus_S\ t\ t0\ I.\ P\ t) \longrightarrow (\ominus_W\ t\ t0\ I.\ P\ t)$
⟨*proof*⟩

lemma *infin-imp-iNextWeak-iNextStrong-eq-iNext*:
$[\![$ *infinite* $I;\ t0 \in I\ ]\!] \Longrightarrow$
$((\bigcirc_W\ t\ t0\ I.\ P\ t) = (\bigcirc\ t\ t0\ I.\ P\ t)) \wedge ((\bigcirc_S\ t\ t0\ I.\ P\ t) = (\bigcirc\ t\ t0\ I.\ P\ t))$
⟨*proof*⟩

lemma *infin-imp-iNextWeak-eq-iNext*: $[\![$ *infinite* $I;\ t0 \in I\ ]\!] \Longrightarrow (\bigcirc_W\ t\ t0\ I.\ P\ t) = (\bigcirc\ t\ t0\ I.\ P\ t)$
⟨*proof*⟩
lemma *infin-imp-iNextStrong-eq-iNext*: $[\![$ *infinite* $I;\ t0 \in I\ ]\!] \Longrightarrow (\bigcirc_S\ t\ t0\ I.\ P\ t) = (\bigcirc\ t\ t0\ I.\ P\ t)$
⟨*proof*⟩
lemma *infin-imp-iNextStrong-eq-iNextWeak*: $[\![$ *infinite* $I;\ t0 \in I\ ]\!] \Longrightarrow (\bigcirc_S\ t\ t0\ I.\ P\ t) = (\bigcirc_W\ t\ t0\ I.\ P\ t)$
⟨*proof*⟩

lemma
not-in-iNext-eq: $t0 \notin I \Longrightarrow (\bigcirc\ t\ t0\ I.\ P\ t) = (P\ t0)$ **and**
not-in-iLast-eq: $t0 \notin I \Longrightarrow (\ominus\ t\ t0\ I.\ P\ t) = (P\ t0)$
⟨*proof*⟩

lemma
not-in-iNextWeak-eq: $t0 \notin I \Longrightarrow (\bigcirc_W\ t\ t0\ I.\ P\ t)$ **and**
not-in-iLastWeak-eq: $t0 \notin I \Longrightarrow (\ominus_W\ t\ t0\ I.\ P\ t)$
⟨*proof*⟩

lemma
not-in-iNextStrong-eq: $t0 \notin I \Longrightarrow \neg (\bigcirc_S\ t\ t0\ I.\ P\ t)$ **and**
not-in-iLastStrong-eq: $t0 \notin I \Longrightarrow \neg (\ominus_S\ t\ t0\ I.\ P\ t)$
⟨*proof*⟩

lemma
iNext-UNIV: $(\bigcirc\ t\ t0\ UNIV.\ P\ t) = P\ (Suc\ t0)$ **and**
iNextWeak-UNIV: $(\bigcirc_W\ t\ t0\ UNIV.\ P\ t) = P\ (Suc\ t0)$ **and**
iNextStrong-UNIV: $(\bigcirc_S\ t\ t0\ UNIV.\ P\ t) = P\ (Suc\ t0)$
⟨*proof*⟩

lemma
iLast-UNIV: $(\ominus\ t\ t0\ UNIV.\ P\ t) = P\ (t0 - Suc\ 0)$ **and**
iLastWeak-UNIV: $(\ominus_W\ t\ t0\ UNIV.\ P\ t) = (if\ 0 < t0\ then\ P\ (t0 - Suc\ 0)\ else\ True)$ **and**
iLastStrong-UNIV: $(\ominus_S\ t\ t0\ UNIV.\ P\ t) = (if\ 0 < t0\ then\ P\ (t0 - Suc\ 0)\ else\ False)$
⟨*proof*⟩

lemmas *iTL-Next-UNIV* =
iNext-UNIV iNextWeak-UNIV iNextStrong-UNIV
iLast-UNIV iLastWeak-UNIV iLastStrong-UNIV

lemma *inext-nth-iNext-Suc*: $(\bigcirc\ t\ (I \rightarrow n)\ I.\ P\ t) = P\ (I \rightarrow Suc\ n)$
⟨*proof*⟩

lemma *iprev-nth-iLast-Suc*: $(\ominus\ t\ (I \leftarrow n)\ I.\ P\ t) = P\ (I \leftarrow Suc\ n)$
⟨*proof*⟩

3.4 Temporal operators and arithmetic interval operators

Shifting intervals through addition and subtraction of constants. Mirroring intervals through subtraction of intervals from constants. Expanding and compressing intervals through multiplication and division by constants.

Always operator

lemma *iT-Plus-iAll-conv*: $(\Box\ t\ I \oplus k.\ P\ t) = (\Box\ t\ I.\ P\ (t + k))$
⟨*proof*⟩

lemma *iT-Mult-iAll-conv*: $(\Box\ t\ I \otimes k.\ P\ t) = (\Box\ t\ I.\ P\ (t * k))$
⟨*proof*⟩

lemma *iT-Plus-neg-iAll-conv*: $(\Box\ t\ I \oplus- k.\ P\ t) = (\Box\ t\ (I \downarrow\geq k).\ P\ (t - k))$
⟨*proof*⟩

lemma *iT-Minus-iAll-conv*: $(\Box\ t\ k \ominus I.\ P\ t) = (\Box\ t\ (I \downarrow\leq k).\ P\ (k - t))$
⟨*proof*⟩

lemma *iT-Div-iAll-conv*: $(\Box\ t\ I \oslash k.\ P\ t) = (\Box\ t\ I.\ P\ (t\ div\ k))$
⟨*proof*⟩

lemmas *iT-arith-iAll-conv* =
iT-Plus-iAll-conv
iT-Mult-iAll-conv
iT-Plus-neg-iAll-conv
iT-Minus-iAll-conv
iT-Div-iAll-conv

Eventually operator

lemma
iT-Plus-iEx-conv: $(\Diamond\ t\ I \oplus k.\ P\ t) = (\Diamond\ t\ I.\ P\ (t + k))$ **and**
iT-Mult-iEx-conv: $(\Diamond\ t\ I \otimes k.\ P\ t) = (\Diamond\ t\ I.\ P\ (t * k))$ **and**
iT-Plus-neg-iEx-conv: $(\Diamond\ t\ I \oplus- k.\ P\ t) = (\Diamond\ t\ (I \downarrow\geq k).\ P\ (t - k))$ **and**
iT-Minus-iEx-conv: $(\Diamond\ t\ k \ominus I.\ P\ t) = (\Diamond\ t\ (I \downarrow\leq k).\ P\ (k - t))$ **and**
iT-Div-iEx-conv: $(\Diamond\ t\ I \oslash k.\ P\ t) = (\Diamond\ t\ I.\ P\ (t\ div\ k))$
⟨*proof*⟩

Until and Since operators

lemma *iT-Plus-iUntil-conv*: $(P\ t1.\ t1\ \mathcal{U}\ t2\ (I \oplus k).\ Q\ t2) = (P\ (t1 + k).\ t1\ \mathcal{U}\ t2\ I.\ Q\ (t2 + k))$
⟨*proof*⟩

lemma *iT-Mult-iUntil-conv*: $(P\ t1.\ t1\ \mathcal{U}\ t2\ (I \otimes k).\ Q\ t2) = (P\ (t1 * k).\ t1\ \mathcal{U}\ t2\ I.\ Q\ (t2 * k))$
⟨*proof*⟩

lemma *iT-Plus-neg-iUntil-conv*: $(P\ t1.\ t1\ \mathcal{U}\ t2\ (I \oplus- k).\ Q\ t2) = (P\ (t1 - k).\ t1\ \mathcal{U}\ t2\ (I \downarrow\geq k).\ Q\ (t2 - k))$
⟨*proof*⟩

lemma *iT-Minus-iUntil-conv*: $(P\ t1.\ t1\ \mathcal{U}\ t2\ (k \ominus I).\ Q\ t2) = (P\ (k - t1).\ t1\ \mathcal{S}\ t2\ (I \downarrow\leq k).\ Q\ (k - t2))$
⟨*proof*⟩

lemma *iT-Div-iUntil-conv*: $(P\ t1.\ t1\ \mathcal{U}\ t2\ (I \oslash k).\ Q\ t2) = (P\ (t1\ div\ k).\ t1\ \mathcal{U}\ t2\ I.\ Q\ (t2\ div\ k))$
⟨*proof*⟩

Until and Since operators can be converted into each other through substraction of intervals from constants

lemma *iUntil-iSince-conv*:
$[\![$ *finite* I; $Max\ I \leq k\ ]\!] \Longrightarrow$
$(P\ t1.\ t1\ \mathcal{U}\ t2\ I.\ Q\ t2) = (P\ (k - t1).\ t1\ \mathcal{S}\ t2\ (k \ominus I).\ Q\ (k - t2))$
⟨*proof*⟩

lemma *iSince-iUntil-conv*:
$[\![$ *finite* I; $Max\ I \leq k\ ]\!] \Longrightarrow$

$(P\ t1.\ t1\ \mathcal{S}\ t2\ I.\ Q\ t2) = (P\ (k - t1).\ t1\ \mathcal{U}\ t2\ (k \ominus I).\ Q\ (k - t2))$
⟨*proof*⟩

lemma *iT-Plus-iSince-conv*: $(P\ t1.\ t1\ \mathcal{S}\ t2\ (I \oplus k).\ Q\ t2) = (P\ (t1 + k).\ t1\ \mathcal{S}\ t2\ I.\ Q\ (t2 + k))$
⟨*proof*⟩

lemma *iT-Mult-iSince-conv*: $0 < k \Longrightarrow (P\ t1.\ t1\ \mathcal{S}\ t2\ (I \otimes k).\ Q\ t2) = (P\ (t1 * k).\ t1\ \mathcal{S}\ t2\ I.\ Q\ (t2 * k))$
⟨*proof*⟩

lemma *iT-Plus-neg-iSince-conv*: $(P\ t1.\ t1\ \mathcal{S}\ t2\ (I \oplus- k).\ Q\ t2) = (P\ (t1 - k).\ t1\ \mathcal{S}\ t2\ (I \downarrow\geq k).\ Q\ (t2 - k))$
⟨*proof*⟩
lemma *iT-Minus-iSince-conv*:
$(P\ t1.\ t1\ \mathcal{S}\ t2\ (k \ominus I).\ Q\ t2) = (P\ (k - t1).\ t1\ \mathcal{U}\ t2\ (I \downarrow\leq k).\ Q\ (k - t2))$
⟨*proof*⟩

lemma *iT-Div-iSince-conv*:
$0 < k \Longrightarrow (P\ t1.\ t1\ \mathcal{S}\ t2\ (I \oslash k).\ Q\ t2) = (P\ (t1\ div\ k).\ t1\ \mathcal{S}\ t2\ I.\ Q\ (t2\ div\ k))$
⟨*proof*⟩

Weak Until and Weak Since operators

lemma *iT-Plus-iWeakUntil-conv*: $(P\ t1.\ t1\ \mathcal{W}\ t2\ (I \oplus k).\ Q\ t2) = (P\ (t1 + k).\ t1\ \mathcal{W}\ t2\ I.\ Q\ (t2 + k))$
⟨*proof*⟩

lemma *iT-Mult-iWeakUntil-conv*: $(P\ t1.\ t1\ \mathcal{W}\ t2\ (I \otimes k).\ Q\ t2) = (P\ (t1 * k).\ t1\ \mathcal{W}\ t2\ I.\ Q\ (t2 * k))$
⟨*proof*⟩

lemma *iT-Plus-neg-iWeakUntil-conv*: $(P\ t1.\ t1\ \mathcal{W}\ t2\ (I \oplus- k).\ Q\ t2) = (P\ (t1 - k).\ t1\ \mathcal{W}\ t2\ (I \downarrow\geq k).\ Q\ (t2 - k))$
⟨*proof*⟩

lemma *iT-Minus-iWeakUntil-conv*: $(P\ t1.\ t1\ \mathcal{W}\ t2\ (k \ominus I).\ Q\ t2) = (P\ (k - t1).\ t1\ \mathcal{B}\ t2\ (I \downarrow\leq k).\ Q\ (k - t2))$
⟨*proof*⟩

lemma *iT-Div-iWeakUntil-conv*: $(P\ t1.\ t1\ \mathcal{W}\ t2\ (I \oslash k).\ Q\ t2) = (P\ (t1\ div\ k).\ t1\ \mathcal{W}\ t2\ I.\ Q\ (t2\ div\ k))$
⟨*proof*⟩

lemma *iT-Plus-iWeakSince-conv*: $(P\ t1.\ t1\ \mathcal{B}\ t2\ (I \oplus k).\ Q\ t2) = (P\ (t1 + k).\ t1\ \mathcal{B}\ t2\ I.\ Q\ (t2 + k))$
⟨*proof*⟩

lemma *iT-Mult-iWeakSince-conv*: $0 < k \Longrightarrow (P\ t1.\ t1\ \mathcal{B}\ t2\ (I \otimes k).\ Q\ t2) = (P\ (t1 * k).\ t1\ \mathcal{B}\ t2\ I.\ Q\ (t2 * k))$
⟨*proof*⟩

lemma *iT-Plus-neg-iWeakSince-conv*: $(P\ t1.\ t1\ \mathcal{B}\ t2\ (I \oplus\!- k).\ Q\ t2) = (P\ (t1 - k).\ t1\ \mathcal{B}\ t2\ (I \downarrow\geq k).\ Q\ (t2 - k))$
⟨*proof*⟩

lemma *iT-Minus-iWeakSince-conv*:
$(P\ t1.\ t1\ \mathcal{B}\ t2\ (k \ominus I).\ Q\ t2) = (P\ (k - t1).\ t1\ \mathcal{W}\ t2\ (I \downarrow\leq k).\ Q\ (k - t2))$
⟨*proof*⟩

lemma *iT-Div-iWeakSince-conv*:
$0 < k \Longrightarrow (P\ t1.\ t1\ \mathcal{B}\ t2\ (I \oslash k).\ Q\ t2) = (P\ (t1\ div\ k).\ t1\ \mathcal{B}\ t2\ I.\ Q\ (t2\ div\ k))$
⟨*proof*⟩

Release and Trigger operators

lemma *iT-Plus-iRelease-conv*: $(P\ t1.\ t1\ \mathcal{R}\ t2\ (I \oplus k).\ Q\ t2) = (P\ (t1 + k).\ t1\ \mathcal{R}\ t2\ I.\ Q\ (t2 + k))$
⟨*proof*⟩

lemma *iT-Mult-iRelease-conv*: $(P\ t1.\ t1\ \mathcal{R}\ t2\ (I \otimes k).\ Q\ t2) = (P\ (t1 * k).\ t1\ \mathcal{R}\ t2\ I.\ Q\ (t2 * k))$
⟨*proof*⟩

lemma *iT-Plus-neg-iRelease-conv*: $(P\ t1.\ t1\ \mathcal{R}\ t2\ (I \oplus\!- k).\ Q\ t2) = (P\ (t1 - k).\ t1\ \mathcal{R}\ t2\ (I \downarrow\geq k).\ Q\ (t2 - k))$
⟨*proof*⟩

lemma *iT-Minus-iRelease-conv*: $(P\ t1.\ t1\ \mathcal{R}\ t2\ (k \ominus I).\ Q\ t2) = (P\ (k - t1).\ t1\ \mathcal{T}\ t2\ (I \downarrow\leq k).\ Q\ (k - t2))$
⟨*proof*⟩

lemma *iT-Div-iRelease-conv*: $(P\ t1.\ t1\ \mathcal{R}\ t2\ (I \oslash k).\ Q\ t2) = (P\ (t1\ div\ k).\ t1\ \mathcal{R}\ t2\ I.\ Q\ (t2\ div\ k))$
⟨*proof*⟩

lemma *iT-Plus-iTrigger-conv*: $(P\ t1.\ t1\ \mathcal{T}\ t2\ (I \oplus k).\ Q\ t2) = (P\ (t1 + k).\ t1\ \mathcal{T}\ t2\ I.\ Q\ (t2 + k))$
⟨*proof*⟩

lemma *iT-Mult-iTrigger-conv*: $0 < k \Longrightarrow (P\ t1.\ t1\ \mathcal{T}\ t2\ (I \otimes k).\ Q\ t2) = (P\ (t1 * k).\ t1\ \mathcal{T}\ t2\ I.\ Q\ (t2 * k))$
⟨*proof*⟩

lemma *iT-Plus-neg-iTrigger-conv*: $(P\ t1.\ t1\ \mathcal{T}\ t2\ (I \oplus\!- k).\ Q\ t2) = (P\ (t1 -$

$k). t1\ \mathcal{T}\ t2\ (I \downarrow\geq k).\ Q\ (t2 - k))$
⟨*proof*⟩

lemma *iT-Minus-iTrigger-conv*:
$(P\ t1.\ t1\ \mathcal{T}\ t2\ (k \ominus I).\ Q\ t2) = (P\ (k - t1).\ t1\ \mathcal{R}\ t2\ (I \downarrow\leq k).\ Q\ (k - t2))$
⟨*proof*⟩

lemma *iT-Div-iTrigger-conv*:
$0 < k \Longrightarrow (P\ t1.\ t1\ \mathcal{T}\ t2\ (I \oslash k).\ Q\ t2) = (P\ (t1\ div\ k).\ t1\ \mathcal{T}\ t2\ I.\ Q\ (t2\ div\ k))$
⟨*proof*⟩

end