

# Interval Temporal Logic on Natural Numbers

David Trachtenherz

December 14, 2021

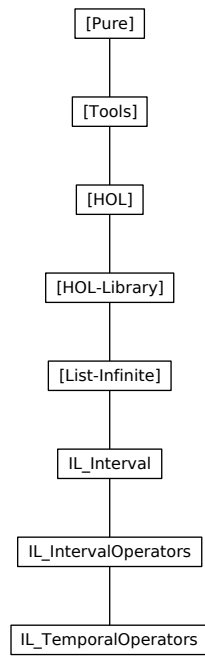
## Abstract

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

<b>1</b>	<b>Intervals and operations for temporal logic declarations</b>	<b>2</b>
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	<i>Min</i> and <i>Max</i> 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 <i>inext</i> and <i>iprev</i> with intervals . . . . .	32
1.7.1	Mirroring of intervals . . . . .	35
1.7.2	Functions <i>inext-nth</i> and <i>iprev-nth</i> on intervals . . . . .	36
1.8	Induction with intervals . . . . .	38

<b>2</b>	<b>Arithmetic operators on natural intervals</b>	<b>39</b>
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	<i>inext</i> and <i>iprev</i> 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
<b>3</b>	<b>Temporal logic operators on natural intervals</b>	<b>78</b>
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 . . . . .	92
3.2.6	Congruence rules for temporal operators' predicates .	94
3.2.7	Temporal operators with set unions/intersections and subsets . . . . .	94
3.3	Further results for temporal operators . . . . .	95
3.4	Temporal operators and arithmetic interval operators . . . . .	98



# 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  $\Rightarrow$  iT ([...])

```

**where**

```

[n...]  $\equiv$  {n.}

```

Finite interval starting at  $0::'a$  and ending at some natural  $n$ .

**definition**

```

iTILL :: Time  $\Rightarrow$  iT ([...])

```

**where**

```

[...n]  $\equiv$  {..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  $d + (1::'a)$  so that an empty interval cannot be defined.

**definition**

```

iIN :: Time  $\Rightarrow$  nat  $\Rightarrow$  iT ([...])

```

**where**

```

[n...,d]  $\equiv$  {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  $\Rightarrow$  nat  $\Rightarrow$  iT ([-, mod -])

```

**where**

```

[r, mod m]  $\equiv$  { x. x mod m = r mod m  $\wedge$  r  $\leq$  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  $c + (1::'a)$  so that an empty interval cannot be defined.

**definition**

$iMODb :: Time \Rightarrow nat \Rightarrow nat \Rightarrow iT ([ -, mod -, - ])$

**where**

$[r, mod m, c] \equiv \{ x. x \text{ mod } m = r \text{ mod } m \wedge r \leq x \wedge x \leq r + m * c \}$

**1.1.2 Membership in an interval**

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

**lemma**  $iFROM-iff: x \in [n..] = (n \leq x)$

$\langle proof \rangle$

**lemma**  $iTILL-iff: x \in [..n] = (x \leq n)$

$\langle proof \rangle$

**lemma**  $iIN-iff: x \in [n..,d] = (n \leq x \wedge x \leq n + d)$

$\langle proof \rangle$

**lemma**  $iMOD-iff: x \in [r, mod m] = (x \text{ mod } m = r \text{ mod } m \wedge r \leq x)$

$\langle proof \rangle$

**lemma**  $iMODb-iff: x \in [r, mod m, c] =$

$(x \text{ mod } m = r \text{ mod } m \wedge r \leq x \wedge x \leq r + m * c)$

$\langle proof \rangle$

**lemma**  $iFROM-D: x \in [n..] \Longrightarrow (n \leq x)$

$\langle proof \rangle$

**lemma**  $iTILL-D: x \in [..n] \Longrightarrow (x \leq n)$

$\langle proof \rangle$

**corollary**  $iIN-geD: x \in [n..,d] \Longrightarrow n \leq x$

$\langle proof \rangle$

**corollary**  $iIN-leD: x \in [n..,d] \Longrightarrow x \leq n + d$

$\langle proof \rangle$

**corollary**  $iMOD-modD: x \in [r, mod m] \Longrightarrow x \text{ mod } m = r \text{ mod } m$

$\langle proof \rangle$

**corollary**  $iMOD-geD: x \in [r, mod m] \Longrightarrow r \leq x$

$\langle proof \rangle$

**corollary**  $iMODb-modD: x \in [r, mod m, c] \Longrightarrow x \text{ mod } m = r \text{ mod } m$

$\langle proof \rangle$

**corollary**  $iMODb-geD: x \in [r, mod m, c] \Longrightarrow r \leq x$

$\langle proof \rangle$

**corollary**  $iMODb-leD: x \in [r, mod m, c] \Longrightarrow x \leq r + m * c$

$\langle proof \rangle$

**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]:  $n \leq x \implies x \in [n..]$  **and**  
*iTILL-I* [intro]:  $x \leq n \implies x \in [..n]$  **and**  
*iIN-I* [intro]:  $n \leq x \implies x \leq n + d \implies x \in [n..,d]$  **and**  
*iMOD-I* [intro]:  $x \bmod m = r \bmod m \implies r \leq x \implies x \in [r, \bmod m]$  **and**  
*iMODb-I* [intro]:  $x \bmod m = r \bmod m \implies r \leq x \implies x \leq r + m * c \implies x \in [r, \bmod m, c]$   
 ⟨proof⟩

**lemma**

*iFROM-E* [elim]:  $x \in [n..] \implies (n \leq x \implies P) \implies P$  **and**  
*iTILL-E* [elim]:  $x \in [..n] \implies (x \leq n \implies P) \implies P$  **and**  
*iIN-E* [elim]:  $x \in [n..,d] \implies (n \leq x \implies x \leq n + d \implies P) \implies P$  **and**  
*iMOD-E* [elim]:  $x \in [r, \bmod m] \implies (x \bmod m = r \bmod m \implies r \leq x \implies P) \implies P$  **and**  
*iMODb-E* [elim]:  $x \in [r, \bmod m, c] \implies (x \bmod m = r \bmod m \implies r \leq x \implies x \leq r + m * c \implies P) \implies P$   
 ⟨proof⟩

**lemma** *iIN-Suc-insert-conv*:

$\text{insert} (\text{Suc} (n + d)) [n..,d] = [n..,\text{Suc} d]$   
 ⟨proof⟩

**lemma** *iTILL-Suc-insert-conv*:  $\text{insert} (\text{Suc} n) [..n] = [..\text{Suc} n]$

⟨proof⟩

**lemma** *iMODb-Suc-insert-conv*:

$\text{insert} (r + m * \text{Suc} c) [r, \bmod m, c] = [r, \bmod m, \text{Suc} c]$   
 ⟨proof⟩

**lemma** *iFROM-pred-insert-conv*:  $\text{insert} (n - \text{Suc} 0) [n..] = [n - \text{Suc} 0..]$

⟨proof⟩

**lemma** *iIN-pred-insert-conv*:

$0 < n \implies \text{insert} (n - \text{Suc} 0) [n..,d] = [n - \text{Suc} 0..,\text{Suc} d]$   
 ⟨proof⟩

**lemma** *iMOD-pred-insert-conv*:

$m \leq r \implies \text{insert} (r - m) [r, \bmod m] = [r - m, \bmod m]$   
 ⟨proof⟩

**lemma** *iMODb-pred-insert-conv*:

$m \leq r \implies \text{insert} (r - m) [r, \bmod m, c] = [r - m, \bmod m, \text{Suc} c]$

*<proof>*

**lemma** *iFROM-Suc-pred-insert-conv*:  $\text{insert } n \text{ [Suc } n \dots] = [n \dots]$

*<proof>*

**lemma** *iIN-Suc-pred-insert-conv*:  $\text{insert } n \text{ [Suc } n \dots, d] = [n \dots, \text{Suc } d]$

*<proof>*

**lemma** *iMOD-Suc-pred-insert-conv*:  $\text{insert } r \text{ [} r + m, \text{ mod } m] = [r, \text{ mod } m]$

*<proof>*

**lemma** *iMODb-Suc-pred-insert-conv*:  $\text{insert } r \text{ [} r + m, \text{ mod } m, c] = [r, \text{ mod } m, \text{Suc } c]$

*<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*:  $\llbracket a \in [r, \text{ mod } m]; b \in [r, \text{ mod } m] \rrbracket \implies (a - b) \text{ mod } m = 0$

*<proof>*

**lemma** *iMODb-mem-diff*:  $\llbracket a \in [r, \text{ mod } m, c]; b \in [r, \text{ mod } m, c] \rrbracket \implies (a - b) \text{ mod } m = 0$

*<proof>*

### 1.1.3 Interval conversions

**lemma** *iIN-0-iTILL-conv*:  $[0 \dots, n] = [\dots n]$

*<proof>*

**lemma** *iIN-iTILL-iTILL-conv*:  $0 < n \implies [n \dots, d] = [\dots n + d] - [\dots n - \text{Suc } 0]$

*<proof>*

**lemma** *iIN-iFROM-iTILL-conv*:  $[n \dots, d] = [n \dots] \cap [\dots n + d]$

*<proof>*

**lemma** *iMODb-iMOD-iTILL-conv*:  $[r, \text{ mod } m, c] = [r, \text{ mod } m] \cap [\dots r + m * c]$

*<proof>*

**lemma** *iMODb-iMOD-iIN-conv*:  $[r, \text{ mod } m, c] = [r, \text{ mod } m] \cap [r \dots, m * c]$

*<proof>*

**lemma** *iFROM-iTILL-iIN-conv*:  $n \leq n' \implies [n \dots] \cap [\dots n'] = [n \dots, n' - n]$

*<proof>*

**lemma** *iMOD-iTILL-iMODb-conv*:

$$r \leq n \implies [r, \text{mod } m] \cap [\dots n] = [r, \text{mod } m, (n - r) \text{ div } m]$$

*<proof>*

**lemma** *iMOD-iIN-iMODb-conv*:

$$[r, \text{mod } m] \cap [r\dots, d] = [r, \text{mod } m, d \text{ div } m]$$

*<proof>*

**lemma** *iFROM-0*:  $[0\dots] = \text{UNIV}$

*<proof>*

**lemma** *iTILL-0*:  $[\dots 0] = \{0\}$

*<proof>*

**lemma** *iIN-0*:  $[n\dots, 0] = \{n\}$

*<proof>*

**lemma** *iMOD-0*:  $[r, \text{mod } 0] = [r\dots, 0]$

*<proof>*

**lemma** *iMODb-mod-0*:  $[r, \text{mod } 0, c] = [r\dots, 0]$

*<proof>*

**lemma** *iMODb-0*:  $[r, \text{mod } m, 0] = [r\dots, 0]$

*<proof>*

**lemmas** *iT-0* =

*iFROM-0*

*iTILL-0*

*iIN-0*

*iMOD-0*

*iMODb-mod-0*

*iMODb-0*

**lemma** *iMOD-1*:  $[r, \text{mod } \text{Suc } 0] = [r\dots]$

*<proof>*

**lemma** *iMODb-mod-1*:  $[r, \text{mod } \text{Suc } 0, c] = [r\dots, c]$

*<proof>*

#### 1.1.4 Finiteness and emptiness of intervals

**lemma**

*iFROM-not-empty*:  $[n\dots] \neq \{\}$  **and**

*iTILL-not-empty*:  $[\dots n] \neq \{\}$  **and**

*iIN-not-empty*:  $[n\dots, d] \neq \{\}$  **and**

*iMOD-not-empty*:  $[r, \text{mod } m] \neq \{\}$  **and**

*iMODb-not-empty*:  $[r, \text{mod } m, c] \neq \{\}$

*<proof>*



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

**lemma**  
*iTILL-finite*: *finite* [ $\dots n$ ] **and**  
*iIN-finite*: *finite* [ $n \dots d$ ] **and**  
*iMODb-finite*: *finite* [ $r, \text{mod } m, c$ ] **and**  
*iMOD-0-finite*: *finite* [ $r, \text{mod } 0$ ]  
⟨*proof*⟩

**lemma** *iFROM-infinite*: *infinite* [ $n \dots$ ]  
⟨*proof*⟩

**lemma** *iMOD-infinite*:  $0 < m \implies \textit{infinite}$  [ $r, \text{mod } m$ ]  
⟨*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* [ $\dots n$ ] = 0 **and**  
*iFROM-Min*: *iMin* [ $n \dots$ ] =  $n$  **and**  
*iIN-Min*: *iMin* [ $n \dots d$ ] =  $n$  **and**  
*iMOD-Min*: *iMin* [ $r, \text{mod } m$ ] =  $r$  **and**  
*iMODb-Min*: *iMin* [ $r, \text{mod } m, c$ ] =  $r$   
⟨*proof*⟩

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

**lemma**  
*iTILL-Max*: *Max* [ $\dots n$ ] =  $n$  **and**

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

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

**lemma**  
*iTILL-iMax*:  $i\text{Max } [..n] = \text{enat } n$  **and**  
*iIN-iMax*:  $i\text{Max } [n..,d] = \text{enat } (n+d)$  **and**  
*iMODb-iMax*:  $i\text{Max } [r, \text{mod } m, c] = \text{enat } (r + m * c)$  **and**  
*iMOD-0-iMax*:  $i\text{Max } [r, \text{mod } 0] = \text{enat } r$  **and**  
*iFROM-iMax*:  $i\text{Max } [n..] = \infty$  **and**  
*iMOD-iMax*:  $0 < m \implies i\text{Max } [r, \text{mod } m] = \infty$   
 ⟨*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*:  $x \in [n..] \implies x + k \in [n..]$  **and**  
*iFROM-Suc*:  $x \in [n..] \implies \text{Suc } x \in [n..]$  **and**  
*iFROM-minus*:  $\llbracket x \in [n..]; k \leq x - n \rrbracket \implies x - k \in [n..]$  **and**  
*iFROM-pred*:  $n < x \implies x - \text{Suc } 0 \in [n..]$   
 ⟨*proof*⟩

**lemma**  
*iTILL-plus*:  $\llbracket x \in [..n]; k \leq n - x \rrbracket \implies x + k \in [..n]$  **and**  
*iTILL-Suc*:  $x < n \implies \text{Suc } x \in [..n]$  **and**  
*iTILL-minus*:  $x \in [..n] \implies x - k \in [..n]$  **and**  
*iTILL-pred*:  $x \in [..n] \implies x - \text{Suc } 0 \in [..n]$   
 ⟨*proof*⟩

**lemma** *iIN-plus*:  $\llbracket x \in [n..,d]; k \leq n + d - x \rrbracket \implies x + k \in [n..,d]$   
 ⟨*proof*⟩

**lemma** *iIN-Suc*:  $\llbracket x \in [n..,d]; x < n + d \rrbracket \implies \text{Suc } x \in [n..,d]$   
 ⟨*proof*⟩

**lemma** *iIN-minus*:  $\llbracket x \in [n..d]; k \leq x - n \rrbracket \implies x - k \in [n..d]$   
 ⟨proof⟩

**lemma** *iIN-pred*:  $\llbracket x \in [n..d]; n < x \rrbracket \implies x - \text{Suc } 0 \in [n..d]$   
 ⟨proof⟩

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

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

**lemma** *iMOD-minus-divisor-mult*:  
 $\llbracket x \in [r, \text{mod } m]; k * m \leq x - r \rrbracket \implies x - k * m \in [r, \text{mod } m]$   
 ⟨proof⟩

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

**corollary** *iMOD-minus-divisor*:  
 $\llbracket x \in [r, \text{mod } m]; m + r \leq x \rrbracket \implies x - m \in [r, \text{mod } m]$   
 ⟨proof⟩

**lemma** *iMOD-plus*:  
 $x \in [r, \text{mod } m] \implies (x + k \in [r, \text{mod } m]) = (k \text{ mod } m = 0)$   
 ⟨proof⟩

**corollary** *iMOD-Suc*:  
 $x \in [r, \text{mod } m] \implies (\text{Suc } x \in [r, \text{mod } m]) = (m = \text{Suc } 0)$   
 ⟨proof⟩

**lemma** *iMOD-minus*:  
 $\llbracket x \in [r, \text{mod } m]; k \leq x - r \rrbracket \implies (x - k \in [r, \text{mod } m]) = (k \text{ mod } m = 0)$   
 ⟨proof⟩

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

**lemma** *iMODb-plus-divisor-mult*:  
 $\llbracket x \in [r, \text{mod } m, c]; k * m \leq r + m * c - x \rrbracket \implies x + k * m \in [r, \text{mod } m, c]$   
 ⟨proof⟩

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

**lemma** *iMODb-plus-divisor*:

$\llbracket x \in [r, \text{mod } m, c]; x < r + m * c \rrbracket \implies x + m \in [r, \text{mod } m, c]$   
 ⟨proof⟩

**lemma** *iMODb-minus-divisor-mult*:

$\llbracket x \in [r, \text{mod } m, c]; r + k * m \leq x \rrbracket \implies x - k * m \in [r, \text{mod } m, c]$   
 ⟨proof⟩

**lemma** *iMODb-plus*:

$\llbracket x \in [r, \text{mod } m, c]; k \leq r + m * c - x \rrbracket \implies$   
 $(x + k \in [r, \text{mod } m, c]) = (k \text{ mod } m = 0)$   
 ⟨proof⟩

**corollary** *iMODb-Suc*:

$\llbracket x \in [r, \text{mod } m, c]; x < r + m * c \rrbracket \implies$   
 $(\text{Suc } x \in [r, \text{mod } m, c]) = (m = \text{Suc } 0)$   
 ⟨proof⟩

**lemma** *iMODb-minus*:

$\llbracket x \in [r, \text{mod } m, c]; k \leq x - r \rrbracket \implies$   
 $(x - k \in [r, \text{mod } m, c]) = (k \text{ mod } m = 0)$   
 ⟨proof⟩

**corollary** *iMODb-pred*:

$\llbracket x \in [r, \text{mod } m, c]; r < x \rrbracket \implies$   
 $(x - \text{Suc } 0 \in [r, \text{mod } m, c]) = (m = \text{Suc } 0)$   
 ⟨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*:

$\llbracket n \in S; \text{Suc } n \in S; m \neq \text{Suc } 0 \rrbracket \implies \neg S \subseteq [r, \text{mod } m]$   
 ⟨proof⟩

**corollary** *Suc-in-imp-neq-iMOD*:

$\llbracket n \in S; \text{Suc } n \in S; m \neq \text{Suc } 0 \rrbracket \implies S \neq [r, \text{mod } m]$   
 ⟨proof⟩

**lemma** *Suc-in-imp-not-subset-iMODb*:

$\llbracket n \in S; \text{Suc } n \in S; m \neq \text{Suc } 0 \rrbracket \implies \neg S \subseteq [r, \text{mod } m, c]$   
 ⟨proof⟩

**corollary** *Suc-in-imp-neq-iMODb*:

$\llbracket n \in S; \text{Suc } n \in S; m \neq \text{Suc } 0 \rrbracket \implies S \neq [r, \text{mod } m, c]$   
 ⟨proof⟩

### 1.3.2 Subset relation between intervals

**lemma**

*iIN-iFROM-subset-same*:  $[n\dots,d] \subseteq [n\dots]$  **and**  
*iIN-iTILL-subset-same*:  $[n\dots,d] \subseteq [\dots,n+d]$  **and**  
*iMOD-iFROM-subset-same*:  $[r, \text{mod } m] \subseteq [r\dots]$  **and**  
*iMODb-iTILL-subset-same*:  $[r, \text{mod } m, c] \subseteq [\dots,r+m*c]$  **and**  
*iMODb-iIN-subset-same*:  $[r, \text{mod } m, c] \subseteq [r\dots,m*c]$  **and**  
*iMODb-iMOD-subset-same*:  $[r, \text{mod } m, c] \subseteq [r, \text{mod } m]$

*<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*:  $x \in [r, \text{mod } m, c] \implies x \in [r, \text{mod } m]$

*<proof>*

**lemma** *iMOD-imp-iMODb*:

$\llbracket x \in [r, \text{mod } m]; x \leq r + m * c \rrbracket \implies x \in [r, \text{mod } m, c]$

*<proof>*

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

*<proof>*

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

*<proof>*

**lemma** *iMODb-singleton-subset-conv*:

$([r, \text{mod } m, c] \subseteq \{a\}) = (r = a \wedge (m = 0 \vee c = 0))$

*<proof>*

**lemma** *iMODb-singleton-eq-conv*:

$([r, \text{mod } m, c] = \{a\}) = (r = a \wedge (m = 0 \vee c = 0))$

*<proof>*

**lemma** *iMODb-subset-imp-divisor-mod-0*:

$\llbracket 0 < c'; [r', \text{mod } m', c'] \subseteq [r, \text{mod } m, c] \rrbracket \implies m' \text{ mod } m = 0$

*<proof>*

**lemma** *iMOD-subset-imp-divisor-mod-0*:

$[r', \text{mod } m'] \subseteq [r, \text{mod } m] \implies m' \text{ mod } m = 0$

*<proof>*

**lemma** *iMOD-subset-imp-iMODb-subset*:

$\llbracket [r', \text{mod } m'] \subseteq [r, \text{mod } m]; r' + m' * c' \leq r + m * c \rrbracket \implies [r', \text{mod } m', c'] \subseteq [r, \text{mod } m, c]$

$\langle proof \rangle$

**lemma** *iMODb-subset-imp-iMOD-subset*:

$$\llbracket [r', \text{mod } m', c'] \subseteq [r, \text{mod } m, c]; 0 < c' \rrbracket \implies \\ [r', \text{mod } m'] \subseteq [r, \text{mod } m]$$

$\langle proof \rangle$

**lemma** *iMODb-0-iMOD-subset-conv*:

$$([r', \text{mod } m', 0] \subseteq [r, \text{mod } m]) = \\ (r' \text{ mod } m = r \text{ mod } m \wedge r \leq r')$$

$\langle proof \rangle$

**lemma** *iFROM-subset-conv*:  $([n' \dots] \subseteq [n \dots]) = (n \leq n')$

$\langle proof \rangle$

**lemma** *iFROM-iMOD-subset-conv*:  $([n' \dots] \subseteq [r, \text{mod } m]) = (r \leq n' \wedge m = \text{Suc } 0)$

$\langle proof \rangle$

**lemma** *iIN-subset-conv*:  $([n' \dots, d'] \subseteq [n \dots, d]) = (n \leq n' \wedge n' + d' \leq n + d)$

$\langle proof \rangle$

**lemma** *iIN-iFROM-subset-conv*:  $([n' \dots, d'] \subseteq [n \dots]) = (n \leq n')$

$\langle proof \rangle$

**lemma** *iIN-iTILL-subset-conv*:  $([n' \dots, d'] \subseteq [\dots n]) = (n' + d' \leq n)$

$\langle proof \rangle$

**lemma** *iIN-iMOD-subset-conv*:

$$0 < d' \implies ([n' \dots, d'] \subseteq [r, \text{mod } m]) = (r \leq n' \wedge m = \text{Suc } 0)$$

$\langle proof \rangle$

**lemma** *iIN-iMODb-subset-conv*:

$$0 < d' \implies \\ ([n' \dots, d'] \subseteq [r, \text{mod } m, c]) = \\ (r \leq n' \wedge m = \text{Suc } 0 \wedge n' + d' \leq r + m * c)$$

$\langle proof \rangle$

**lemma** *iTILL-subset-conv*:  $([\dots n'] \subseteq [\dots n]) = (n' \leq n)$

$\langle proof \rangle$

**lemma** *iTILL-iFROM-subset-conv*:  $([\dots n'] \subseteq [n \dots]) = (n = 0)$

$\langle proof \rangle$

**lemma** *iTILL-iIN-subset-conv*:  $([\dots n'] \subseteq [n \dots, d]) = (n = 0 \wedge n' \leq d)$

$\langle proof \rangle$

**lemma** *iTILL-iMOD-subset-conv*:

$0 < n' \implies ([..n'] \subseteq [r, \text{mod } m]) = (r = 0 \wedge m = \text{Suc } 0)$   
 ⟨proof⟩

**lemma** *iTILL-iMODb-subset-conv*:

$0 < n' \implies ([..n'] \subseteq [r, \text{mod } m, c]) = (r = 0 \wedge m = \text{Suc } 0 \wedge n' \leq r + m * c)$   
 ⟨proof⟩

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

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

**lemma** *iMODb-iIN-subset-conv*:

$([r', \text{mod } m', c'] \subseteq [n..,d]) = (n \leq r' \wedge r' + m' * c' \leq n + d)$   
 ⟨proof⟩

**lemma** *iMODb-iTILL-subset-conv*:

$([r', \text{mod } m', c'] \subseteq [..n]) = (r' + m' * c' \leq n)$   
 ⟨proof⟩

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

**lemma** *iMOD-subset-conv*:  $0 < m \implies$

$([r', \text{mod } m'] \subseteq [r, \text{mod } m]) =$   
 $(r' \text{ mod } m = r \text{ mod } m \wedge r \leq r' \wedge m' \text{ mod } m = 0)$   
 ⟨proof⟩

**lemma** *iMODb-subset-mod-0-conv*:

$([r', \text{mod } m', c'] \subseteq [r, \text{mod } 0, c]) = (r'=r \wedge (m'=0 \vee c'=0))$   
 ⟨proof⟩

**lemma** *iMODb-subset-0-conv*:

$([r', \text{mod } m', c'] \subseteq [r, \text{mod } m, 0]) = (r'=r \wedge (m'=0 \vee c'=0))$   
 ⟨proof⟩

**lemma** *iMODb-0-subset-conv*:

$([r', \text{mod } m', 0] \subseteq [r, \text{mod } m, c]) = (r' \in [r, \text{mod } m, c])$   
 ⟨proof⟩

**lemma** *iMODb-mod-0-subset-conv*:

$([r', \text{mod } 0, c'] \subseteq [r, \text{mod } m, c]) = (r' \in [r, \text{mod } m, c])$   
 ⟨proof⟩

**lemma** *iMODb-subset-conv'*:  $\llbracket 0 < c; 0 < c' \rrbracket \implies$

$([r', \text{mod } m', c'] \subseteq [r, \text{mod } m, c]) =$   
 $(r' \text{ mod } m = r \text{ mod } m \wedge r \leq r' \wedge m' \text{ mod } m = 0 \wedge$



$r' + m' * c' \leq r + m * c$   
 ⟨proof⟩

**lemma** *iMODb-subset-conv*:  $\llbracket 0 < m'; 0 < c' \rrbracket \implies$   
 $([r', \text{mod } m', c'] \subseteq [r, \text{mod } m, c]) =$   
 $(r' \text{ mod } m = r \text{ mod } m \wedge r \leq r' \wedge m' \text{ mod } m = 0 \wedge$   
 $r' + m' * c' \leq r + m * c)$   
 ⟨proof⟩

**lemma** *iMODb-iMOD-subset-conv*:  $0 < c' \implies$   
 $([r', \text{mod } m', c'] \subseteq [r, \text{mod } m]) =$   
 $(r' \text{ mod } m = r \text{ mod } m \wedge r \leq r' \wedge m' \text{ mod } m = 0)$   
 ⟨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*:  $n \leq n' \implies [n'..] \subseteq [n..]$   
 ⟨proof⟩

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

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

**lemma** *not-iFROM-iMOD-subset*:  $m \neq \text{Suc } 0 \implies \neg [n'..] \subseteq [r, \text{mod } m]$   
 ⟨proof⟩

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

*<proof>*

**lemma** *iIN-subset*:  $\llbracket n \leq n'; n' + d' \leq n + d \rrbracket \implies [n' \dots, d'] \subseteq [n \dots, d]$   
*<proof>*

**lemma** *iIN-iFROM-subset*:  $n \leq n' \implies [n' \dots, d'] \subseteq [n \dots]$   
*<proof>*

**lemma** *iIN-iTILL-subset*:  $n' + d' \leq n \implies [n' \dots, d'] \subseteq [\dots n]$   
*<proof>*

**lemma** *not-iIN-iMODb-subset*:  $\llbracket 0 < d'; m \neq \text{Suc } 0 \rrbracket \implies \neg [n' \dots, d'] \subseteq [r, \text{mod } m, c]$   
*<proof>*

**lemma** *not-iIN-iMOD-subset*:  $\llbracket 0 < d'; m \neq \text{Suc } 0 \rrbracket \implies \neg [n' \dots, d'] \subseteq [r, \text{mod } m]$   
*<proof>*

**lemma** *iTILL-subset*:  $n' \leq n \implies [\dots n'] \subseteq [\dots n]$   
*<proof>*

**lemma** *iTILL-iFROM-subset*:  $([\dots n'] \subseteq [0 \dots])$   
*<proof>*

**lemma** *iTILL-iIN-subset*:  $n' \leq d \implies ([\dots n'] \subseteq [0 \dots, d])$   
*<proof>*

**lemma** *not-iTILL-iMOD-subset*:  
 $\llbracket 0 < n'; m \neq \text{Suc } 0 \rrbracket \implies \neg [\dots n'] \subseteq [r, \text{mod } m]$   
*<proof>*

**lemma** *not-iTILL-iMODb-subset*:  
 $\llbracket 0 < n'; m \neq \text{Suc } 0 \rrbracket \implies \neg [\dots n'] \subseteq [r, \text{mod } m, c]$   
*<proof>*

**lemma** *iMOD-iFROM-subset*:  $n \leq r' \implies [r', \text{mod } m'] \subseteq [n \dots]$   
*<proof>*

**lemma** *not-iMOD-iIN-subset*:  $0 < m' \implies \neg [r', \text{mod } m'] \subseteq [n \dots, d]$   
*<proof>*

**lemma** *not-iMOD-iTILL-subset*:  $0 < m' \implies \neg [r', \text{mod } m'] \subseteq [\dots n]$   
*<proof>*

**lemma** *iMOD-subset*:  
 $\llbracket r \leq r'; r' \text{ mod } m = r \text{ mod } m; m' \text{ mod } m = 0 \rrbracket \implies [r', \text{mod } m'] \subseteq [r, \text{mod } m]$   
*<proof>*

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

**lemma** *iMODb-iFROM-subset*:  $n \leq r' \implies [r', \text{mod } m', c'] \subseteq [n..]$   
 ⟨proof⟩

**lemma** *iMODb-iTILL-subset*:  
 $r' + m' * c' \leq n \implies [r', \text{mod } m', c'] \subseteq [..n]$   
 ⟨proof⟩

**lemma** *iMODb-iIN-subset*:  
 $\llbracket n \leq r'; r' + m' * c' \leq n + d \rrbracket \implies [r', \text{mod } m', c'] \subseteq [n..,d]$   
 ⟨proof⟩

**lemma** *iMODb-iMOD-subset*:  
 $\llbracket r \leq r'; r' \text{ mod } m = r \text{ mod } m; m' \text{ mod } m = 0 \rrbracket \implies [r', \text{mod } m', c'] \subseteq [r, \text{mod } m]$   
 ⟨proof⟩

**lemma** *iMODb-subset*:  
 $\llbracket r \leq r'; r' \text{ mod } m = r \text{ mod } m; m' \text{ mod } m = 0; r' + m' * c' \leq r + m * c \rrbracket \implies$   
 $[r', \text{mod } m', c'] \subseteq [r, \text{mod } m, c]$   
 ⟨proof⟩

**lemma** *iFROM-trans*:  $\llbracket y \in [x..]; z \in [y..] \rrbracket \implies z \in [x..]$   
 ⟨proof⟩

**lemma** *iTILL-trans*:  $\llbracket y \in [..x]; z \in [..y] \rrbracket \implies z \in [..x]$   
 ⟨proof⟩

**lemma** *iIN-trans*:  
 $\llbracket y \in [x..,d]; z \in [y..,d']; d' \leq x + d - y \rrbracket \implies z \in [x..,d]$   
 ⟨proof⟩

**lemma** *iMOD-trans*:  
 $\llbracket y \in [x, \text{mod } m]; z \in [y, \text{mod } m] \rrbracket \implies z \in [x, \text{mod } m]$   
 ⟨proof⟩

**lemma** *iMODb-trans*:  
 $\llbracket y \in [x, \text{mod } m, c]; z \in [y, \text{mod } m, c']; m * c' \leq x + m * c - y \rrbracket \implies$   
 $z \in [x, \text{mod } m, c]$   
 ⟨proof⟩

**lemma** *iMODb-trans'*:  
 $\llbracket y \in [x, \text{mod } m, c]; z \in [y, \text{mod } m, c']; c' \leq x \text{ div } m + c - y \text{ div } m \rrbracket \implies$   
 $z \in [x, \text{mod } m, c]$   
 ⟨proof⟩

### 1.3.3 Equality of intervals

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

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

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

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

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

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

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

**lemma** *iMODb-eq-conv*:  $\llbracket 0 < m; 0 < c \rrbracket \implies$   
 $([r, \text{mod } m, c] = [r', \text{mod } m', c']) = (r = r' \wedge m = m' \wedge c = c')$   
 ⟨proof⟩

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

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

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

### 1.3.4 Inequality of intervals

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

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

**corollary** *iFROM-iMOD-neg*:  $m \neq \text{Suc } 0 \implies [n\dots] \neq [r, \text{mod } m]$

*<proof>*

**corollary** *iFROM-iMODb-neq*:  $[n\dots] \neq [r, \text{mod } m, c]$

*<proof>*

**corollary** *iIN-iMOD-neq*:  $0 < m \implies [n\dots, d] \neq [r, \text{mod } m]$

*<proof>*

**corollary** *iIN-iMODb-neq2*:  $\llbracket m \neq \text{Suc } 0; 0 < d \rrbracket \implies [n\dots, d] \neq [r, \text{mod } m, c]$

*<proof>*

**lemma** *iIN-iMODb-neq*:  $\llbracket 2 \leq m; 0 < c \rrbracket \implies [n\dots, d] \neq [r, \text{mod } m, c]$

*<proof>*

**lemma** *iTILL-iIN-neq*:  $0 < n \implies [\dots n^\wedge] \neq [n\dots, d]$

*<proof>*

**corollary** *iTILL-iMOD-neq*:  $0 < m \implies [\dots n] \neq [r, \text{mod } m]$

*<proof>*

**corollary** *iTILL-iMODb-neq*:

$\llbracket m \neq \text{Suc } 0; 0 < n \rrbracket \implies [\dots n] \neq [r, \text{mod } m, c]$

*<proof>*

**lemma** *iMOD-iMODb-neq*:  $0 < m \implies [r, \text{mod } m] \neq [r', \text{mod } m', c^\wedge]$

*<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'*:  $[n\dots] \cup [n'\dots] = [\min n n'\dots]$

*<proof>*

**corollary** *iFROM-union*:  $n \leq n' \implies [n\dots] \cup [n'\dots] = [n\dots]$

*<proof>*

**lemma** *iFROM-inter'*:  $[n\dots] \cap [n'\dots] = [\max n n'\dots]$

*<proof>*

**corollary** *iFROM-inter*:  $n' \leq n \implies [n\dots] \cap [n'\dots] = [n\dots]$

*<proof>*

**lemma** *iTILL-union'*:  $[\dots n] \cup [\dots n^\wedge] = [\dots \max n n^\wedge]$

*<proof>*

**corollary** *iTILL-union*:  $n' \leq n \implies [\dots n] \cup [\dots n'] = [\dots n]$   
 ⟨proof⟩

**lemma** *iTILL-iFROM-union*:  $n \leq n' \implies [\dots n'] \cup [n\dots] = UNIV$   
 ⟨proof⟩

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

**corollary** *iTILL-inter*:  $n \leq n' \implies [\dots n] \cap [\dots n'] = [\dots n]$   
 ⟨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*:  
 $\llbracket n \leq n'; n' \leq \text{Suc } (n + d); n + d \leq n' + d' \rrbracket \implies$   
 $[n\dots, d] \cup [n'\dots, d'] = [n\dots, n' - n + d']$   
 ⟨proof⟩

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

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

**lemma** *iIN-append-union-pred*:  
 $0 < d \implies [n\dots, d - \text{Suc } 0] \cup [n + d\dots, d'] = [n\dots, d + d']$   
 ⟨proof⟩

**lemma** *iIN-iFROM-union*:  
 $n' \leq \text{Suc } (n + d) \implies [n\dots, d] \cup [n'\dots] = [\min n n'\dots]$   
 ⟨proof⟩

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

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

**lemma** *iIN-iFROM-append-union-pred*:  
 $0 < d \implies [n\dots, d - \text{Suc } 0] \cup [n + d\dots] = [n\dots]$   
 ⟨proof⟩

**lemma** *iIN-inter*:

$$\begin{aligned} & \llbracket n \leq n'; n' \leq n + d; n + d \leq n' + d' \rrbracket \implies \\ & [n \dots, d] \cap [n' \dots, d'] = [n' \dots, n + d - n'] \\ & \langle \text{proof} \rangle \end{aligned}$$

**lemma** *iMOD-union*:

$$\begin{aligned} & \llbracket r \leq r'; r \bmod m = r' \bmod m \rrbracket \implies \\ & [r, \bmod m] \cup [r', \bmod m] = [r, \bmod m] \\ & \langle \text{proof} \rangle \end{aligned}$$

**lemma** *iMOD-union'*:

$$\begin{aligned} & r \bmod m = r' \bmod m \implies \\ & [r, \bmod m] \cup [r', \bmod m] = [\min r r', \bmod m] \\ & \langle \text{proof} \rangle \end{aligned}$$

**lemma** *iMOD-inter*:

$$\begin{aligned} & \llbracket r \leq r'; r \bmod m = r' \bmod m \rrbracket \implies \\ & [r, \bmod m] \cap [r', \bmod m] = [r', \bmod m] \\ & \langle \text{proof} \rangle \end{aligned}$$

**lemma** *iMOD-inter'*:

$$\begin{aligned} & r \bmod m = r' \bmod m \implies \\ & [r, \bmod m] \cap [r', \bmod m] = [\max r r', \bmod m] \\ & \langle \text{proof} \rangle \end{aligned}$$

**lemma** *iMODb-union*:

$$\begin{aligned} & \llbracket r \leq r'; r \bmod m = r' \bmod m; r' \leq r + m * c; r + m * c \leq r' + m * c' \rrbracket \implies \\ & [r, \bmod m, c] \cup [r', \bmod m, c'] = [r, \bmod m, r' \text{ div } m - r \text{ div } m + c'] \\ & \langle \text{proof} \rangle \end{aligned}$$

**lemma** *iMODb-append-union*:

$$\begin{aligned} & [r, \bmod m, c] \cup [r + m * c, \bmod m, c'] = [r, \bmod m, c + c'] \\ & \langle \text{proof} \rangle \end{aligned}$$

**lemma** *iMODb-iMOD-append-union'*:

$$\begin{aligned} & \llbracket r \bmod m = r' \bmod m; r' \leq r + m * \text{Suc } c \rrbracket \implies \\ & [r, \bmod m, c] \cup [r', \bmod m] = [\min r r', \bmod m] \\ & \langle \text{proof} \rangle \end{aligned}$$

**lemma** *iMODb-iMOD-append-union*:

$$\begin{aligned} & \llbracket r \leq r'; r \bmod m = r' \bmod m; r' \leq r + m * \text{Suc } c \rrbracket \implies \\ & [r, \bmod m, c] \cup [r', \bmod m] = [r, \bmod m] \\ & \langle \text{proof} \rangle \end{aligned}$$

**lemma** *iMODb-append-union-Suc*:

$$\begin{aligned} & [r, \bmod m, c] \cup [r + m * \text{Suc } c, \bmod m, c'] = [r, \bmod m, \text{Suc } (c + c')] \\ & \langle \text{proof} \rangle \end{aligned}$$

**lemma** *iMODb-append-union-pred*:

$$0 < c \implies [r, \bmod m, c - \text{Suc } 0] \cup [r + m * c, \bmod m, c'] = [r, \bmod m, c +$$

$c'$   
 $\langle proof \rangle$

**lemma** *iMODb-inter*:

$\llbracket r \leq r'; r \bmod m = r' \bmod m; r' \leq r + m * c; r + m * c \leq r' + m * c' \rrbracket \implies$   
 $[r, \bmod m, c] \cap [r', \bmod m, c'] = [r', \bmod m, c - (r'-r) \text{ div } m]$   
 $\langle proof \rangle$

**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*:

$0 < m \implies (\bigcup k. A \cap [k * m .. m - \text{Suc } 0]) = A$



$\langle \text{proof} \rangle$

**lemma** *finite-mod-partition-Union*:

$\llbracket 0 < m; \text{finite } A \rrbracket \implies$   
 $(\bigcup_{k \leq \text{Max } A \text{ div } m} A \cap [k * m \dots, m - \text{Suc } 0]) = A$   
 $\langle \text{proof} \rangle$

**lemma** *mod-partition-is-disjoint*:

$\llbracket 0 < (m :: \text{nat}); k \neq k' \rrbracket \implies$   
 $(A \cap [k * m \dots, m - \text{Suc } 0]) \cap (A \cap [k' * m \dots, m - \text{Suc } 0]) = \{\}$   
 $\langle \text{proof} \rangle$

## 1.5 Cutting intervals

**lemma** *iTILL-cut-le*:  $[\dots n] \downarrow \leq t = (\text{if } t \leq n \text{ then } [\dots t] \text{ else } [\dots n])$   
 $\langle \text{proof} \rangle$

**corollary** *iTILL-cut-le1*:  $t \in [\dots n] \implies [\dots n] \downarrow \leq t = [\dots t]$   
 $\langle \text{proof} \rangle$

**corollary** *iTILL-cut-le2*:  $t \notin [\dots n] \implies [\dots n] \downarrow \leq t = [\dots n]$   
 $\langle \text{proof} \rangle$

**lemma** *iFROM-cut-le*:

$[n \dots] \downarrow \leq t =$   
 $(\text{if } t < n \text{ then } \{\} \text{ else } [n \dots, t - n])$   
 $\langle \text{proof} \rangle$

**corollary** *iFROM-cut-le1*:  $t \in [n \dots] \implies [n \dots] \downarrow \leq t = [n \dots, t - n]$   
 $\langle \text{proof} \rangle$

**lemma** *iIN-cut-le*:

$[n \dots, d] \downarrow \leq t =$   
 $(\text{if } t < n \text{ then } \{\} \text{ else}$   
 $\text{if } t \leq n + d \text{ then } [n \dots, t - n]$   
 $\text{else } [n \dots, d])$   
 $\langle \text{proof} \rangle$

**corollary** *iIN-cut-le1*:

$t \in [n \dots, d] \implies [n \dots, d] \downarrow \leq t = [n \dots, t - n]$   
 $\langle \text{proof} \rangle$

**lemma** *iMOD-cut-le*:

$[r, \text{mod } m] \downarrow \leq t =$   
 $(\text{if } t < r \text{ then } \{\}$   
 $\text{else } [r, \text{mod } m, (t - r) \text{ div } m])$   
 $\langle \text{proof} \rangle$

**lemma** *iMOD-cut-le1*:

$t \in [r, \text{mod } m] \implies$   
 $[r, \text{mod } m] \downarrow \leq t = [r, \text{mod } m, (t - r) \text{ div } m]$   
 ⟨proof⟩

**lemma** *iMODb-cut-le*:

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

**lemma** *iMODb-cut-le1*:

$t \in [r, \text{mod } m, c] \implies$   
 $[r, \text{mod } m, c] \downarrow \leq t = [r, \text{mod } m, (t - r) \text{ div } m]$   
 ⟨proof⟩

**lemma** *iTILL-cut-less*:

$[..n] \downarrow < t =$  (  
   if  $n < t$  then  $[..n]$  else  
   if  $t = 0$  then {}  
   else  $[..t - \text{Suc } 0]$ )  
 ⟨proof⟩

**lemma** *iTILL-cut-less1*:

$\llbracket t \in [..n]; 0 < t \rrbracket \implies [..n] \downarrow < t = [..t - \text{Suc } 0]$   
 ⟨proof⟩

**lemma** *iFROM-cut-less*:

$[n..] \downarrow < t =$  (  
   if  $t \leq n$  then {}  
   else  $[n.., t - \text{Suc } n]$ )  
 ⟨proof⟩

**lemma** *iFROM-cut-less1*:

$n < t \implies [n..] \downarrow < t = [n.., t - \text{Suc } n]$   
 ⟨proof⟩

**lemma** *iIN-cut-less*:

$[n.., d] \downarrow < t =$  (  
   if  $t \leq n$  then {} else  
   if  $t \leq n + d$  then  $[n.., t - \text{Suc } n]$   
   else  $[n.., d]$ )  
 ⟨proof⟩

**lemma** *iIN-cut-less1*:

$\llbracket t \in [n..d]; n < t \rrbracket \implies [n..d] \downarrow < t = [n.., t - \text{Suc } n]$   
 ⟨proof⟩

**lemma** *iMOD-cut-less*:

$[r, \text{mod } m] \downarrow < t = ($   
   if  $t \leq r$  then  $\{\}$   
   else  $[r, \text{mod } m, (t - \text{Suc } r) \text{ div } m]$   
 ⟨proof⟩

**lemma** *iMOD-cut-less1*:

$\llbracket t \in [r, \text{mod } m]; r < t \rrbracket \implies$   
 $[r, \text{mod } m] \downarrow < t = [r, \text{mod } m, (t - r) \text{ div } m - \text{Suc } 0]$   
 ⟨proof⟩

**lemma** *iMODb-cut-less*:

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

**lemma** *iMODb-cut-less1*:  $\llbracket t \in [r, \text{mod } m, c]; r < t \rrbracket \implies$

$[r, \text{mod } m, c] \downarrow < t = [r, \text{mod } m, (t - r) \text{ div } m - \text{Suc } 0]$   
 ⟨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*:  
 $[\dots n] \downarrow_{\geq} t = (\text{if } n < t \text{ then } \{\} \text{ else } [t \dots, n-t])$   
 $\langle \text{proof} \rangle$

**corollary** *iTILL-cut-ge1*:  $t \in [\dots n] \implies [\dots n] \downarrow_{\geq} t = [t \dots, n-t]$   
 $\langle \text{proof} \rangle$

**corollary** *iTILL-cut-ge2*:  $t \notin [\dots n] \implies [\dots n] \downarrow_{\geq} t = \{\}$   
 $\langle \text{proof} \rangle$

**lemma** *iTILL-cut-greater*:  
 $[\dots n] \downarrow_{>} t = (\text{if } n \leq t \text{ then } \{\} \text{ else } [Suc t \dots, n - Suc t])$   
 $\langle \text{proof} \rangle$

**corollary** *iTILL-cut-greater1*:  
 $t \in [\dots n] \implies t < n \implies [\dots n] \downarrow_{>} t = [Suc t \dots, n - Suc t]$   
 $\langle \text{proof} \rangle$

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

**lemma** *iFROM-cut-ge*:

$[\dots] \downarrow \geq t = (\text{if } n \leq t \text{ then } [t\dots] \text{ else } [n\dots])$

⟨proof⟩

**corollary** *iFROM-cut-ge1*:  $t \in [n\dots] \implies [n\dots] \downarrow \geq t = [t\dots]$

⟨proof⟩

**lemma** *iFROM-cut-greater*:

$[\dots] \downarrow > t = (\text{if } n \leq t \text{ then } [\text{Suc } t\dots] \text{ else } [n\dots])$

⟨proof⟩

**corollary** *iFROM-cut-greater1*:

$t \in [n\dots] \implies [n\dots] \downarrow > t = [\text{Suc } t\dots]$

⟨proof⟩

**lemma** *iIN-cut-ge*:

$[\dots, d] \downarrow \geq t = (\text{if } t < n \text{ then } [n\dots, d] \text{ else } \text{if } t \leq n+d \text{ then } [t\dots, n+d-t] \text{ else } \{\})$

⟨proof⟩

**corollary** *iIN-cut-ge1*:  $t \in [n\dots, d] \implies$

$[\dots, d] \downarrow \geq t = [t\dots, n+d-t]$

⟨proof⟩

**corollary** *iIN-cut-ge2*:  $t \notin [n\dots, d] \implies$

$[\dots, d] \downarrow \geq t = (\text{if } t < n \text{ then } [n\dots, d] \text{ else } \{\})$

⟨proof⟩

**lemma** *iIN-cut-greater*:

$[\dots, d] \downarrow > t = (\text{if } t < n \text{ then } [n\dots, d] \text{ else } \text{if } t < n+d \text{ then } [\text{Suc } t\dots, n+d - \text{Suc } t] \text{ else } \{\})$

⟨proof⟩

**corollary** *iIN-cut-greater1*:

$\llbracket t \in [n\dots, d]; t < n+d \rrbracket \implies$

$[\dots, d] \downarrow > t = [\text{Suc } t\dots, n+d - \text{Suc } t]$

⟨proof⟩

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

$$\llbracket 0 < (m::nat); r \leq t \rrbracket \implies$$

$$t < t + m - (t - r) \text{ mod } m$$
 <proof>

**lemma** *mod-cut-greater-aux-le-x*:  

$$\llbracket (r::nat) \leq t; t < x; x \text{ mod } m = r \text{ mod } m \rrbracket \implies$$

$$t + m - (t - r) \text{ mod } m \leq x$$
 <proof>

**lemma** *iMOD-cut-greater*:  

$$[r, \text{ mod } m] \downarrow > t = ($$

$$\text{if } t < r \text{ then } [r, \text{ mod } m] \text{ else}$$

$$\text{if } m = 0 \text{ then } \{\}$$

$$\text{else } [t + m - (t - r) \text{ mod } m, \text{ mod } m])$$
 <proof>

**lemma** *iMOD-cut-greater1*:  

$$t \in [r, \text{ mod } m] \implies$$

$$[r, \text{ mod } m] \downarrow > t = ($$

$$\text{if } m = 0 \text{ then } \{\}$$

$$\text{else } [t + m, \text{ mod } m])$$
 <proof>

**lemma** *iMODb-cut-greater-aux*:  

$$\llbracket 0 < m; t < r + m * c; r \leq t \rrbracket \implies$$

$$(r + m * c - (t + m - (t - r) \text{ mod } m)) \text{ div } m =$$

$$c - \text{Suc } ((t - r) \text{ div } m)$$
 <proof>

**lemma** *iMODb-cut-greater*:  

$$[r, \text{ mod } m, c] \downarrow > t = ($$

$$\text{if } t < r \text{ then } [r, \text{ mod } m, c] \text{ else}$$

$$\text{if } r + m * c \leq t \text{ then } \{\}$$

$$\text{else } [t + m - (t - r) \text{ mod } m, \text{ mod } m, c - \text{Suc } ((t - r) \text{ div } m)])$$
 <proof>

**lemma** *iMODb-cut-greater1*:  

$$t \in [r, \text{ mod } m, c] \implies$$

$$[r, \text{ mod } m, c] \downarrow > t = ($$

$$\text{if } r + m * c \leq t \text{ then } \{\}$$

$$\text{else } [t + m, \text{ mod } m, c - \text{Suc } ((t - r) \text{ div } m)])$$
 <proof>

**lemma** *iMOD-cut-ge*:  

$$[r, \text{ mod } m] \downarrow \geq t = ($$

$$\text{if } t \leq r \text{ then } [r, \text{ mod } m] \text{ else}$$

$$\text{if } m = 0 \text{ then } \{\}$$

else  $[t + m - \text{Suc} ((t - \text{Suc } r) \bmod m), \bmod m]$   
 ⟨proof⟩

**lemma** *iMOD-cut-ge1*:

$t \in [r, \bmod m] \implies$   
 $[r, \bmod m] \downarrow_{\geq} t = [t, \bmod m]$   
 ⟨proof⟩

**lemma** *iMODb-cut-ge*:

$[r, \bmod m, c] \downarrow_{\geq} t = ($   
   if  $t \leq r$  then  $[r, \bmod m, c]$  else  
   if  $r + m * c < t$  then  $\{\}$   
   else  $[t + m - \text{Suc} ((t - \text{Suc } r) \bmod m), \bmod m, c - (t + m - \text{Suc } r) \text{ div } m]$   
 ⟨proof⟩

**lemma** *iMODb-cut-ge1*:

$t \in [r, \bmod m, c] \implies$   
 $[r, \bmod m, c] \downarrow_{\geq} t = ($   
   if  $r + m * c < t$  then  $\{\}$   
   else  $[t, \bmod m, c - (t - r) \text{ div } m]$   
 ⟨proof⟩

**lemma** *iMOD-0-cut-greater*:

$t \in [r, \bmod 0] \implies [r, \bmod 0] \downarrow_{>} t = \{\}$   
 ⟨proof⟩

**lemma** *iMODb-0-cut-greater*:  $t \in [r, \bmod 0, c] \implies$

$[r, \bmod 0, c] \downarrow_{>} t = \{\}$   
 ⟨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*:  $\text{card } [n..] = 0$   
*<proof>*

**lemma** *iTILL-card*:  $\text{card } [..n] = \text{Suc } n$   
*<proof>*

**lemma** *iIN-card*:  $\text{card } [n..,d] = \text{Suc } d$   
*<proof>*

**lemma** *iMOD-0-card*:  $\text{card } [r, \text{mod } 0] = \text{Suc } 0$   
*<proof>*

**lemma** *iMOD-card*:  $0 < m \implies \text{card } [r, \text{mod } m] = 0$   
*<proof>*



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

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

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

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

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

Cardinality with *icard*

**lemma** *iFROM-icard*:  $\text{icard } [n..] = \infty$   
 ⟨proof⟩

**lemma** *iTILL-icard*:  $\text{icard } [..n] = \text{enat } (\text{Suc } n)$   
 ⟨proof⟩

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

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

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

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

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

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

**lemma** *iMODb-icard-if*:  $\text{icard } [r, \text{mod } m, c] = \text{enat } (\text{if } m = 0 \text{ then } \text{Suc } 0 \text{ else } \text{Suc } c)$   
 ⟨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*:  $t \in [n..] \implies \text{inext } t [n..] = \text{Suc } t$  **and**  
*iTILL-inext*:  $t < n \implies \text{inext } t [..n] = \text{Suc } t$  **and**  
*iIN-inext*:  $\llbracket n \leq t; t < n + d \rrbracket \implies \text{inext } t [n..,d] = \text{Suc } t$   
⟨proof⟩

**lemma**

*iFROM-iprev'*:  $t \in [n..] \implies \text{iprev } (\text{Suc } t) [n..] = t$  **and**  
*iFROM-iprev*:  $n < t \implies \text{iprev } t [n..] = t - \text{Suc } 0$  **and**  
*iTILL-iprev*:  $t \in [..n] \implies \text{iprev } t [..n] = t - \text{Suc } 0$  **and**  
*iIN-iprev*:  $\llbracket n < t; t \leq n + d \rrbracket \implies \text{iprev } t [n..,d] = t - \text{Suc } 0$  **and**  
*iIN-iprev'*:  $\llbracket n \leq t; t < n + d \rrbracket \implies \text{iprev } (\text{Suc } t) [n..,d] = t$   
⟨proof⟩

**lemma** *iMOD-inext*:  $t \in [r, \text{mod } m] \implies \text{inext } t [r, \text{mod } m] = t + m$   
⟨proof⟩

**lemma** *iMOD-iprev*:  $\llbracket t \in [r, \text{mod } m]; r < t \rrbracket \implies \text{iprev } t [r, \text{mod } m] = t - m$   
⟨proof⟩

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

**lemma** *iMODb-inext*:

$\llbracket t \in [r, \text{mod } m, c]; t < r + m * c \rrbracket \implies$   
 $\text{inext } t [r, \text{mod } m, c] = t + m$   
⟨proof⟩

**lemma** *iMODb-iprev*:

$\llbracket t \in [r, \text{mod } m, c]; r < t \rrbracket \implies$   
 $\text{iprev } t [r, \text{mod } m, c] = t - m$   
⟨proof⟩

**lemma** *iMODb-iprev'*:

$\llbracket t \in [r, \text{mod } m, c]; t < r + m * c \rrbracket \implies$   
 $\text{iprev } (t + m) [r, \text{mod } m, c] = t$   
⟨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*:

$$\text{iprev } t [n \dots, d] = (\text{if } n < t \wedge t \leq n + d \text{ then } t - \text{Suc } 0 \text{ else } t)$$

*<proof>*

**lemma** *iMOD-iprev-if*:

$$\text{iprev } t [r, \text{mod } m] =$$

$$(\text{if } t \in [r, \text{mod } m] \wedge r < t \text{ then } t - m \text{ else } t)$$

*<proof>*

**lemma** *iMODb-iprev-if*:

$$\text{iprev } t [r, \text{mod } m, c] =$$

$$(\text{if } t \in [r, \text{mod } m, c] \wedge r < t \text{ then } t - m \text{ else } t)$$

*<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*:

$$t \in [n \dots] \implies \text{inext } t [n \dots] - t = \text{Suc } 0$$

*<proof>*

**lemma** *iFROM-iprev-diff-const*:

$$n < t \implies t - \text{iprev } t [n \dots] = \text{Suc } 0$$

*<proof>*

**lemma** *iFROM-iprev-diff-const'*:

$$t \in [n \dots] \implies \text{Suc } t - \text{iprev } (\text{Suc } t) [n \dots] = \text{Suc } 0$$

*<proof>*

**lemma** *iTILL-inext-diff-const*:

$$t < n \implies \text{inext } t [\dots n] - t = \text{Suc } 0$$

*<proof>*

**lemma** *iTILL-iprev-diff-const*:

$$\llbracket t \in [\dots n]; 0 < t \rrbracket \implies t - \text{iprev } t [\dots n] = \text{Suc } 0$$

*<proof>*

**lemma** *iIN-inext-diff-const*:

$$\llbracket n \leq t; t < n + d \rrbracket \implies \text{inext } t [n \dots, d] - t = \text{Suc } 0$$

*<proof>*

**lemma** *iIN-iprev-diff-const*:

$$\llbracket n < t; t \leq n + d \rrbracket \implies t - \text{iprev } t [n \dots, d] = \text{Suc } 0$$

*<proof>*

**lemma** *iIN-iprev-diff-const'*:

$$\llbracket n \leq t; t < n + d \rrbracket \implies \text{Suc } t - \text{iprev } (\text{Suc } t) [n \dots, d] = \text{Suc } 0$$

*<proof>*

**lemma** *iMOD-inext-diff-const:*

$$t \in [r, \text{mod } m] \implies \text{inext } t [r, \text{mod } m] - t = m$$

*<proof>*

**lemma** *iMOD-iprev-diff-const':*

$$t \in [r, \text{mod } m] \implies (t + m) - \text{iprev } (t + m) [r, \text{mod } m] = m$$

*<proof>*

**lemma** *iMOD-iprev-diff-const:*

$$\llbracket t \in [r, \text{mod } m]; r < t \rrbracket \implies t - \text{iprev } t [r, \text{mod } m] = m$$

*<proof>*

**lemma** *iMODb-inext-diff-const:*

$$\llbracket t \in [r, \text{mod } m, c]; t < r + m * c \rrbracket \implies \text{inext } t [r, \text{mod } m, c] - t = m$$

*<proof>*

**lemma** *iMODb-iprev-diff-const':*

$$\llbracket t \in [r, \text{mod } m, c]; t < r + m * c \rrbracket \implies (t + m) - \text{iprev } (t + m) [r, \text{mod } m, c] = m$$

*<proof>*

**lemma** *iMODb-iprev-diff-const:*

$$\llbracket t \in [r, \text{mod } m, c]; r < t \rrbracket \implies t - \text{iprev } t [r, \text{mod } m, c] = m$$

*<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-lem: mirror-lem*  $x [n..d] = n + d - x$  **and**

*iTILL-mirror-lem: mirror-lem*  $x [..n] = n - x$  **and**

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

**lemma** *iMODb-imirror-bounds*:

$r' + m' * c' \leq l + r \implies$   
 $\text{imirror-bounds } [r', \text{mod } m', c'] l r = [l + r - r' - m' * c', \text{mod } m', c']$   
 ⟨proof⟩

**lemma** *iIN-imirror-bounds*:

$n + d \leq l + r \implies \text{imirror-bounds } [n \dots, d] l r = [l + r - n - d \dots, d]$   
 ⟨proof⟩

**lemma** *iTILL-imirror-bounds*:

$n \leq l + r \implies \text{imirror-bounds } [\dots, n] l r = [l + r - n \dots, n]$   
 ⟨proof⟩

**lemmas** *iT-imirror-bounds =*

*iTILL-imirror-bounds*  
*iIN-imirror-bounds*  
*iMODb-imirror-bounds*

**lemma** *iMODb-imirror-ident*:  $\text{imirror } [r, \text{mod } m, c] = [r, \text{mod } m, c]$   
 ⟨proof⟩

**lemma** *iIN-imirror-ident*:  $\text{imirror } [n \dots, d] = [n \dots, d]$   
 ⟨proof⟩

**lemma** *iTILL-imirror-ident*:  $\text{imirror } [\dots, n] = [\dots, n]$   
 ⟨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* :  $[n \dots] \rightarrow a = n + a$   
 ⟨proof⟩

**lemma** *iIN-inext-nth* :  $a \leq d \implies [n \dots, d] \rightarrow a = n + a$   
 ⟨proof⟩

**lemma** *iIN-iprev-nth*:  $a \leq d \implies [n \dots, d] \leftarrow a = n + d - a$   
 ⟨proof⟩

**lemma** *iIN-inext-nth-if* :

$[n \dots, d] \rightarrow a = (\text{if } a \leq d \text{ then } n + a \text{ else } n + d)$   
 ⟨proof⟩

**lemma** *iIN-iprev-nth-if*:

$$[n\dots,d] \leftarrow a = (\text{if } a \leq d \text{ then } n + d - a \text{ else } n)$$

*<proof>*

**lemma** *iTILL-inext-nth* :  $a \leq n \implies [\dots n] \rightarrow a = a$

*<proof>*

**lemma** *iTILL-inext-nth-if* :

$$[\dots n] \rightarrow a = (\text{if } a \leq n \text{ then } a \text{ else } n)$$

*<proof>*

**lemma** *iTILL-iprev-nth*:  $a \leq n \implies [\dots n] \leftarrow a = n - a$

*<proof>*

**lemma** *iTILL-iprev-nth-if*:

$$[\dots n] \leftarrow a = (\text{if } a \leq n \text{ then } n - a \text{ else } 0)$$

*<proof>*

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

*<proof>*

**lemma** *iMODb-inext-nth*:  $a \leq c \implies [r, \text{mod } m, c] \rightarrow a = r + m * a$

*<proof>*

**lemma** *iMODb-inext-nth-if*:

$$[r, \text{mod } m, c] \rightarrow a = (\text{if } a \leq c \text{ then } r + m * a \text{ else } r + m * c)$$

*<proof>*

**lemma** *iMODb-iprev-nth*:

$$a \leq c \implies [r, \text{mod } m, c] \leftarrow a = r + m * (c - a)$$

*<proof>*

**lemma** *iMODb-iprev-nth-if*:

$$[r, \text{mod } m, c] \leftarrow a = (\text{if } a \leq c \text{ then } r + m * (c - a) \text{ else } r)$$

*<proof>*

**lemma** *iIN-iFROM-inext-nth*:

$$a \leq d \implies [n\dots,d] \rightarrow a = [n\dots] \rightarrow a$$

*<proof>*

**lemma** *iIN-iFROM-inext*:

$$a < n + d \implies \text{inext } a [n\dots,d] = \text{inext } a [n\dots]$$

*<proof>*

**lemma** *iMOD-iMODb-inext-nth*:

$$a \leq c \implies [r, \text{mod } m, c] \rightarrow a = [r, \text{mod } m] \rightarrow a$$

*<proof>*

**lemma** *iMOD-iMODb-inext*:

$$a < r + m * c \implies \text{inext } a [r, \text{mod } m, c] = \text{inext } a [r, \text{mod } m]$$

*<proof>*

**lemma** *iMOD-inext-nth-Suc-diff*:

$$([r, \text{mod } m] \rightarrow (\text{Suc } n)) - ([r, \text{mod } m] \rightarrow n) = m$$

*<proof>*

**lemma** *iMOD-inext-nth-diff*:

$$([r, \text{mod } m] \rightarrow a) - ([r, \text{mod } m] \rightarrow b) = (a - b) * m$$

*<proof>*

**lemma** *iMODb-inext-nth-diff*:  $\llbracket a \leq c; b \leq c \rrbracket \implies$

$$([r, \text{mod } m, c] \rightarrow a) - ([r, \text{mod } m, c] \rightarrow b) = (a - b) * m$$

*<proof>*

## 1.8 Induction with intervals

**lemma** *iFROM-induct*:

$$\llbracket P n; \bigwedge k. \llbracket k \in [n..]; P k \rrbracket \implies P (\text{Suc } k); a \in [n..] \rrbracket \implies P a$$

*<proof>*

**lemma** *iIN-induct*:

$$\llbracket P n; \bigwedge k. \llbracket k \in [n..,d]; k \neq n + d; P k \rrbracket \implies P (\text{Suc } k); a \in [n..,d] \rrbracket \implies P a$$

*<proof>*

**lemma** *iTILL-induct*:

$$\llbracket P 0; \bigwedge k. \llbracket k \in [..n]; k \neq n; P k \rrbracket \implies P (\text{Suc } k); a \in [..n] \rrbracket \implies P a$$

*<proof>*

**lemma** *iMOD-induct*:

$$\llbracket P r; \bigwedge k. \llbracket k \in [r, \text{mod } m]; P k \rrbracket \implies P (k + m); a \in [r, \text{mod } m] \rrbracket \implies P a$$

*<proof>*

**lemma** *iMODb-induct*:

$$\llbracket P r; \bigwedge k. \llbracket k \in [r, \text{mod } m, c]; k \neq r + m * c; P k \rrbracket \implies P (k + m); a \in [r, \text{mod } m, c] \rrbracket \implies P a$$

*<proof>*

**lemma** *iIN-rev-induct*:

$$\llbracket P (n + d); \bigwedge k. \llbracket k \in [n..,d]; k \neq n; P k \rrbracket \implies P (k - \text{Suc } 0); a \in [n..,d] \rrbracket \implies P a$$

*<proof>*

**lemma** *iTILL-rev-induct*:

$$\llbracket P n; \bigwedge k. \llbracket k \in [..n]; 0 < k; P k \rrbracket \implies P (k - \text{Suc } 0); a \in [..n] \rrbracket \implies P a$$

*<proof>*



**lemma** *iMODb-rev-induct*:  

$$\llbracket P (r + m * c); \bigwedge k. \llbracket k \in [r, \text{mod } m, c]; k \neq r; P k \rrbracket \implies P (k - m); a \in [r, \text{mod } m, c] \rrbracket \implies P a$$
 $\langle \text{proof} \rangle$

**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**  $\oplus$  55)  
**where**  $I \oplus k \equiv (\lambda n. (n + k)) \text{ ' } I$

**definition** *iT-Mult* :: *iT*  $\Rightarrow$  *Time*  $\Rightarrow$  *iT* (**infixl**  $\otimes$  55)  
**where**  $iT\text{-Mult-def} : I \otimes k \equiv (\lambda n. (n * k)) \text{ ' } I$

**lemma** *iT-Plus-image-conv*:  $I \oplus k = (\lambda n. (n + k)) \text{ ' } I$   
 $\langle \text{proof} \rangle$

**lemma** *iT-Mult-image-conv*:  $I \otimes k = (\lambda n. (n * k)) \text{ ' } I$   
 $\langle \text{proof} \rangle$

**lemma** *iT-Plus-empty*:  $\{\} \oplus k = \{\}$   
 $\langle \text{proof} \rangle$

**lemma** *iT-Mult-empty*:  $\{\} \otimes k = \{\}$   
 $\langle \text{proof} \rangle$

**lemma** *iT-Plus-not-empty*:  $I \neq \{\} \implies I \oplus k \neq \{\}$   
 $\langle \text{proof} \rangle$

**lemma** *iT-Mult-not-empty*:  $I \neq \{\} \implies I \otimes k \neq \{\}$   
 $\langle \text{proof} \rangle$

**lemma** *iT-Plus-empty-iff*:  $(I \oplus k = \{\}) = (I = \{\})$   
 $\langle \text{proof} \rangle$

**lemma** *iT-Mult-empty-iff*:  $(I \otimes k = \{\}) = (I = \{\})$   
 $\langle \text{proof} \rangle$

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

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

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

**corollary** *iT-Mult-0-if*:  $I \otimes 0 = (\text{if } I = \{\} \text{ then } \{\} \text{ else } [\dots 0])$   
 ⟨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*:  
 $0 < k \implies x \in (I \otimes k) = (x \text{ mod } k = 0 \wedge x \text{ div } k \in I)$   
 ⟨proof⟩

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

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

**lemma** *iT-Mult-image*:  $f \text{ ' } I \otimes n = (\lambda x. f x * n) \text{ ' } I$   
 ⟨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*:  $\text{finite } (I \oplus k) = \text{finite } I$   
 ⟨proof⟩

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

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

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

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

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

**lemma** *iT-Mult-Max*:  $\llbracket \text{finite } I; I \neq \{\} \rrbracket \implies Max (I \otimes k) = Max I * k$   
 ⟨proof⟩

**corollary**

*iMOD-mult-0*:  $[r, \text{mod } m] \otimes 0 = [\dots 0]$  **and**

*iMODb-mult-0*:  $[r, \text{mod } m, c] \otimes 0 = [\dots 0]$  **and**

*iFROM-mult-0*:  $[n \dots] \otimes 0 = [\dots 0]$  **and**

*iIN-mult-0*:  $[n \dots, d] \otimes 0 = [\dots 0]$  **and**

*iTILL-mult-0*:  $[\dots n] \otimes 0 = [\dots 0]$

⟨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*:  $I \otimes \text{Suc } 0 = I$

*<proof>*

**corollary**

*iFROM-add-Min*:  $iMin ([n..] \oplus k) = n + k$  **and**  
*iIN-add-Min*:  $iMin ([n..,d] \oplus k) = n + k$  **and**  
*iTILL-add-Min*:  $iMin ([..n] \oplus k) = k$  **and**  
*iMOD-add-Min*:  $iMin ([r, \text{mod } m] \oplus k) = r + k$  **and**  
*iMODb-add-Min*:  $iMin ([r, \text{mod } m, c] \oplus k) = r + k$   
*<proof>*

**corollary**

*iFROM-mult-Min*:  $iMin ([n..] \otimes k) = n * k$  **and**  
*iIN-mult-Min*:  $iMin ([n..,d] \otimes k) = n * k$  **and**  
*iTILL-mult-Min*:  $iMin ([..n] \otimes k) = 0$  **and**  
*iMOD-mult-Min*:  $iMin ([r, \text{mod } m] \otimes k) = r * k$  **and**  
*iMODb-mult-Min*:  $iMin ([r, \text{mod } m, c] \otimes k) = r * k$   
*<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*:  $[n..] \oplus k = [n+k..]$

*<proof>*

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

*<proof>*

**lemma** *iTILL-add*:  $[ \dots i ] \oplus k = [ k \dots, i ]$   
*<proof>*

**lemma** *iMOD-add*:  $[ r, \text{mod } m ] \oplus k = [ r + k, \text{mod } m ]$   
*<proof>*

**lemma** *iMODb-add*:  $[ r, \text{mod } m, c ] \oplus k = [ r + k, \text{mod } m, c ]$   
*<proof>*

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

**lemma** *iFROM-mult*:  $[ n \dots ] \otimes k = [ n * k, \text{mod } k ]$   
*<proof>*

**lemma** *iIN-mult*:  $[ n \dots, d ] \otimes k = [ n * k, \text{mod } k, d ]$   
*<proof>*

**lemma** *iTILL-mult*:  $[ \dots n ] \otimes k = [ 0, \text{mod } k, n ]$   
*<proof>*

**lemma** *iMOD-mult*:  $[ r, \text{mod } m ] \otimes k = [ r * k, \text{mod } m * k ]$   
*<proof>*

**lemma** *iMODb-mult*:  
 $[ r, \text{mod } m, c ] \otimes k = [ r * k, \text{mod } m * k, c ]$   
*<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*:  $[ n \dots ] = UNIV \oplus n$   
*<proof>*

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

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

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

Some examples showing the utility of `iMODb_conv`

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

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

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

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

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

### 2.1.3 Subtraction of constants

**definition** *iT-Plus-neg* ::  $iT \Rightarrow \text{Time} \Rightarrow iT$  (**infixl**  $\oplus -$  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*:  $k \leq x \implies (x - k \in I \oplus - k) = (x \in I)$   
 ⟨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*:  $t \leq k \implies (I \downarrow \geq t) \oplus - k = I \oplus - k$   
 ⟨proof⟩

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

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

**lemma** *iT-Plus-neg-Max-less-empty*:

$\llbracket \text{finite } I; \text{Max } I < k \rrbracket \implies I \oplus - k = \{\}$   
 ⟨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*:  
 $(I \oplus - k = \{\}) = (I = \{\}) \vee (\text{finite } I \wedge \text{Max } I < k)$   
 ⟨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*:  $b \leq a \implies I \oplus a \oplus - b = I \oplus (a - b)$   
 ⟨proof⟩

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

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

**corollary** *iT-Plus-neg-Plus-le-Min-eq*:  
 $\llbracket a \leq b; a \leq iMin I \rrbracket \implies (I \oplus - a) \oplus b = I \oplus (b - a)$   
 ⟨proof⟩

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

**corollary** *iT-Plus-neg-Plus-ge-Min-eq*:  
 $\llbracket b \leq a; a \leq iMin I \rrbracket \implies (I \oplus - a) \oplus b = I \oplus - (a - b)$   
 ⟨proof⟩

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

**lemma** *iT-Plus-neg-Plus-le-inverse*:  $k \leq iMin I \implies I \oplus - k \oplus k = I$   
 ⟨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*:  $\llbracket \text{finite } I; I \neq \{\} \rrbracket \implies I \oplus - \text{Max } I = \{0\}$   
*<proof>*

**lemma** *iT-Plus-neg-singleton*:  $\{a\} \oplus - k = (\text{if } k \leq a \text{ then } \{a - k\} \text{ else } \{\})$   
*<proof>*

**corollary** *iT-Plus-neg-singleton1*:  $k \leq a \implies \{a\} \oplus - k = \{a - k\}$   
*<proof>*

**corollary** *iT-Plus-neg-singleton2*:  $a < k \implies \{a\} \oplus - k = \{\}$   
*<proof>*

**lemma** *iT-Plus-neg-finite-iff*:  $\text{finite } (I \oplus - k) = \text{finite } I$   
*<proof>*

**lemma** *iT-Plus-neg-Min*:  
 $I \oplus - k \neq \{\} \implies i\text{Min } (I \oplus - k) = i\text{Min } (I \downarrow \geq k) - k$   
*<proof>*

**lemma** *iT-Plus-neg-Max*:  
 $\llbracket \text{finite } I; I \oplus - k \neq \{\} \rrbracket \implies \text{Max } (I \oplus - k) = \text{Max } I - k$   
*<proof>*

Subtractions of constants from intervals

**lemma** *iFROM-add-neg*:  $[n \dots] \oplus - k = [n - k \dots]$   
*<proof>*

**lemma** *iTILL-add-neg*:  $[\dots n] \oplus - k = (\text{if } k \leq n \text{ then } [\dots n - k] \text{ else } \{\})$   
*<proof>*

**lemma** *iTILL-add-neg1*:  $k \leq n \implies [\dots n] \oplus - k = [\dots n - k]$   
*<proof>*

**lemma** *iTILL-add-neg2*:  $n < k \implies [\dots n] \oplus - k = \{\}$   
*<proof>*

**lemma** *iIN-add-neg*:  
 $[n \dots, d] \oplus - k = (\text{if } k \leq n \text{ then } [n - k \dots, d])$



else if  $k \leq n + d$  then  $[\dots n + d - k]$  else  $\{\}$   
 ⟨proof⟩

**lemma** *iIN-add-neg1*:  $k \leq n \implies [n\dots, d] \oplus -k = [n - k\dots, d]$   
 ⟨proof⟩

**lemma** *iIN-add-neg2*:  $\llbracket n \leq k; k \leq n + d \rrbracket \implies [n\dots, d] \oplus -k = [\dots n + d - k]$   
 ⟨proof⟩

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

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

**lemma** *iMOD-gr0-add-neg*:

$0 < m \implies$   
 $[r, \text{mod } m] \oplus -k =$   
 if  $k \leq r$  then  $[r - k, \text{mod } m]$   
 else  $[(m + r \text{ mod } m - k \text{ mod } m) \text{ mod } m, \text{mod } m]$   
 ⟨proof⟩

**lemma** *iMOD-add-neg*:

$[r, \text{mod } m] \oplus -k =$   
 if  $k \leq r$  then  $[r - k, \text{mod } m]$   
 else if  $0 < m$  then  $[(m + r \text{ mod } m - k \text{ mod } m) \text{ mod } m, \text{mod } m]$  else  $\{\}$   
 ⟨proof⟩

**corollary** *iMOD-add-neg1*:

$k \leq r \implies [r, \text{mod } m] \oplus -k = [r - k, \text{mod } m]$   
 ⟨proof⟩

**lemma** *iMOD-add-neg2*:

$\llbracket 0 < m; r < k \rrbracket \implies [r, \text{mod } m] \oplus -k = [(m + r \text{ mod } m - k \text{ mod } m) \text{ mod } m, \text{mod } m]$   
 ⟨proof⟩

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

**lemma** *iMODb-add-neg*:

$[r, \text{mod } m, c] \oplus -k =$   
 if  $k \leq r$  then  $[r - k, \text{mod } m, c]$   
 else

$$\text{if } k \leq r + m * c \text{ then}$$

$$[(m + r \text{ mod } m - k \text{ mod } m) \text{ mod } m, \text{ mod } m, (r + m * c - k) \text{ div } m]$$

$$\text{else } \{\}$$
 (proof)

**lemma** *iMODb-add-neg'*:

$$[r, \text{ mod } m, c] \oplus - k = ($$

$$\text{if } k \leq r \text{ then } [r - k, \text{ mod } m, c]$$

$$\text{else if } k \leq r + m * c \text{ then}$$

$$\text{if } k \text{ mod } m \leq r \text{ mod } m$$

$$\text{then } [r \text{ mod } m - k \text{ mod } m, \text{ mod } m, c + r \text{ div } m - k \text{ div } m]$$

$$\text{else } [m + r \text{ mod } m - k \text{ mod } m, \text{ mod } m, c + r \text{ div } m - \text{Suc } (k \text{ div } m) ]$$

$$\text{else } \{\}$$
 (proof)

**corollary** *iMODb-add-neg1*:

$$k \leq r \implies [r, \text{ mod } m, c] \oplus - k = [r - k, \text{ mod } m, c]$$
 (proof)

**corollary** *iMODb-add-neg2*:

$$[[r < k; k \leq r + m * c] \implies$$

$$[r, \text{ mod } m, c] \oplus - k =$$

$$[(m + r \text{ mod } m - k \text{ mod } m) \text{ mod } m, \text{ mod } m, (r + m * c - k) \text{ div } m]$$
 (proof)

**corollary** *iMODb-add-neg2-mod-le*:

$$[[r < k; k \leq r + m * c; k \text{ mod } m \leq r \text{ mod } m] \implies$$

$$[r, \text{ mod } m, c] \oplus - k =$$

$$[r \text{ mod } m - k \text{ mod } m, \text{ mod } m, c + r \text{ div } m - k \text{ div } m]$$
 (proof)

**corollary** *iMODb-add-neg2-mod-less*:

$$[[r < k; k \leq r + m * c; r \text{ mod } m < k \text{ mod } m] \implies$$

$$[r, \text{ mod } m, c] \oplus - k =$$

$$[m + r \text{ mod } m - k \text{ mod } m, \text{ mod } m, c + r \text{ div } m - \text{Suc } (k \text{ div } m) ]$$
 (proof)

**lemma** *iMODb-add-neg3*:  $r + m * c < k \implies [r, \text{ mod } m, c] \oplus - k = \{\}$   
 (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*  $\Rightarrow$  *iT*  $\Rightarrow$  *iT* (**infixl**  $\ominus$  55)

where  $k \ominus I \equiv \{x. x \leq k \wedge (k - x) \in I\}$

**lemma** *iT-Minus-mem-iff*:  $(x \in k \ominus I) = (x \leq k \wedge k - x \in I)$

*<proof>*

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

*<proof>*

**lemma** *iT-Minus-image-conv*:  $k \ominus I = (\lambda x. k - x) ` (I \downarrow \leq k)$

*<proof>*

**lemma** *iT-Minus-cut-eq*:  $k \leq t \Longrightarrow k \ominus (I \downarrow \leq t) = k \ominus I$

*<proof>*

**lemma** *iT-Minus-Minus-cut-eq*:  $k \ominus (k \ominus (I \downarrow \leq k)) = I \downarrow \leq k$

*<proof>*

**lemma**  $10 \ominus [\dots 3] = [7\dots, 3]$

*<proof>*

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

*<proof>*

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

*<proof>*

**lemma** *iT-Minus-bound*:  $x \in k \ominus I \Longrightarrow x \leq k$

*<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*:  $I \neq \{\} \Longrightarrow (iMin I) \ominus I = \{0\}$

*<proof>*

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

*<proof>*

**lemma** *iT-Minus-imirror-conv*:

$k \ominus I = imirror (I \downarrow \leq k) \oplus k \oplus - (iMin I + Max (I \downarrow \leq k))$

*<proof>*

**lemma** *iT-Minus-imirror-conv'*:

$$k \ominus I = \text{imirror } (I \downarrow \leq k) \oplus k \oplus - (iMin (I \downarrow \leq k) + Max (I \downarrow \leq k))$$

*<proof>*

**lemma** *iT-Minus-Max*:

$$\llbracket I \neq \{\} ; iMin I \leq k \rrbracket \implies Max (k \ominus I) = k - (iMin I)$$

*<proof>*

**lemma** *iT-Minus-Min*:

$$\llbracket I \neq \{\} ; iMin I \leq k \rrbracket \implies iMin (k \ominus I) = k - (Max (I \downarrow \leq k))$$

*<proof>*

**lemma** *iT-Minus-Minus-eq*:  $\llbracket \text{finite } I ; Max I \leq k \rrbracket \implies k \ominus (k \ominus I) = I$

*<proof>*

**lemma** *iT-Minus-Minus-eq2*:  $I \subseteq [\dots k] \implies k \ominus (k \ominus I) = I$

*<proof>*

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

*<proof>*

**lemma** *iT-Minus-Plus-empty*:  $k < n \implies k \ominus (I \oplus n) = \{\}$

*<proof>*

**lemma** *iT-Minus-Plus-commute*:  $n \leq k \implies k \ominus (I \oplus n) = (k - n) \ominus I$

*<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*:

$$\llbracket \text{finite } I ; Max I \leq k \rrbracket \implies (k \ominus I) \oplus n = (k + n) \ominus I$$

*<proof>*

**lemma** *iT-Minus-Plus-assoc2*:

$$I \subseteq [\dots k] \implies (k \ominus I) \oplus n = (k + n) \ominus I$$

*<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*:  $k \ominus \{a\} = (\text{if } a \leq k \text{ then } \{k - a\} \text{ else } \{\})$

*<proof>*

**corollary** *iT-Minus-singleton1*:  $a \leq k \implies k \ominus \{a\} = \{k - a\}$

*<proof>*

**corollary** *iT-Minus-singleton2*:  $k < a \implies k \ominus \{a\} = \{\}$

*<proof>*

**lemma** *iMOD-sub*:

$k \ominus [r, \text{mod } m] =$   
 (if  $r \leq k$  then  $[(k - r) \text{ mod } m, \text{mod } m, (k - r) \text{ div } m]$  else  $\{\}$ )  
*<proof>*

**corollary** *iMOD-sub1*:

$r \leq k \implies k \ominus [r, \text{mod } m] = [(k - r) \text{ mod } m, \text{mod } m, (k - r) \text{ div } m]$   
*<proof>*

**corollary** *iMOD-sub2*:  $k < r \implies k \ominus [r, \text{mod } m] = \{\}$   
*<proof>*

**lemma** *iTILL-sub*:  $k \ominus [\dots n] = (\text{if } n \leq k \text{ then } [k - n \dots, n] \text{ else } [\dots k])$   
*<proof>*

**corollary** *iTILL-sub1*:  $n \leq k \implies k \ominus [\dots n] = [k - n \dots, n]$   
*<proof>*

**corollary** *iTILL-sub2*:  $k \leq n \implies k \ominus [\dots n] = [\dots k]$   
*<proof>*

**lemma** *iMODb-sub*:

$k \ominus [r, \text{mod } m, c] =$   
 (if  $r + m * c \leq k$  then  $[k - r - m * c, \text{mod } m, c]$  else  
 if  $r \leq k$  then  $[(k - r) \text{ mod } m, \text{mod } m, (k - r) \text{ div } m]$  else  $\{\}$ )  
*<proof>*

**corollary** *iMODb-sub1*:

$\llbracket r \leq k; k \leq r + m * c \rrbracket \implies$   
 $k \ominus [r, \text{mod } m, c] = [(k - r) \text{ mod } m, \text{mod } m, (k - r) \text{ div } m]$   
*<proof>*

**corollary** *iMODb-sub2*:  $k < r \implies k \ominus [r, \text{mod } m, c] = \{\}$   
*<proof>*

**corollary** *iMODb-sub3*:

$r + m * c \leq k \implies k \ominus [r, \text{mod } m, c] = [k - r - m * c, \text{mod } m, c]$   
*<proof>*

**lemma** *iFROM-sub*:  $k \ominus [n \dots] = (\text{if } n \leq k \text{ then } [\dots k - n] \text{ else } \{\})$   
*<proof>*

**corollary** *iFROM-sub1*:  $n \leq k \implies k \ominus [n \dots] = [\dots k - n]$   
*<proof>*

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

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

**lemma** *iIN-sub1*:  $n + d \leq k \implies k \ominus [n..,d] = [k - (n + d)..,d]$   
 ⟨proof⟩

**lemma** *iIN-sub2*:  $\llbracket n \leq k; k \leq n + d \rrbracket \implies k \ominus [n..,d] = [..k - n]$   
 ⟨proof⟩

**lemma** *iIN-sub3*:  $k < n \implies k \ominus [n..,d] = \{\}$   
 ⟨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 arithmetic operators

**lemma** *iMOD-div-right-strict-mono-on*:  
 $\llbracket 0 < k; k \leq m \rrbracket \implies \text{strict-mono-on } (\lambda x. x \text{ div } k) [r, \text{mod } m]$   
 ⟨proof⟩

**corollary** *iMOD-div-right-inj-on*:  
 $\llbracket 0 < k; k \leq m \rrbracket \implies \text{inj-on } (\lambda x. x \text{ div } k) [r, \text{mod } m]$   
 ⟨proof⟩

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

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

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

$\llbracket 0 < k; k \leq m \rrbracket \implies \text{strict-mono-on } (\lambda x. x \text{ div } k) [r, \text{mod } m, c]$   
 $\langle \text{proof} \rangle$

**corollary** *iMODb-div-right-inj-on*:

$\llbracket 0 < k; k \leq m \rrbracket \implies \text{inj-on } (\lambda x. x \text{ div } k) [r, \text{mod } m, c]$   
 $\langle \text{proof} \rangle$

**lemma** *iMODb-mult-div-right-inj-on*:

$\text{inj-on } (\lambda x. x \text{ div } (k::\text{nat})) [r, \text{mod } (k * m), c]$   
 $\langle \text{proof} \rangle$

**corollary** *iMODb-mult-div-right-inj-on2*:

$m \text{ mod } k = 0 \implies \text{inj-on } (\lambda x. x \text{ div } k) [r, \text{mod } m, c]$   
 $\langle \text{proof} \rangle$

**definition** *iT-Div* ::  $iT \Rightarrow \text{Time} \Rightarrow iT$  (**infixl**  $\circ$  55)

where  $I \circ k \equiv (\lambda n. (n \text{ div } k)) \text{ ` } I$

**lemma** *iT-Div-image-conv*:  $I \circ k = (\lambda n. (n \text{ div } k)) \text{ ` } I$   
 $\langle \text{proof} \rangle$

**lemma** *iT-Div-mono*:  $A \subseteq B \implies A \circ k \subseteq B \circ k$   
 $\langle \text{proof} \rangle$

**lemma** *iT-Div-empty*:  $\{\} \circ k = \{\}$

$\langle \text{proof} \rangle$

**lemma** *iT-Div-not-empty*:  $I \neq \{\} \implies I \circ k \neq \{\}$

$\langle \text{proof} \rangle$

**lemma** *iT-Div-empty-iff*:  $(I \circ k = \{\}) = (I = \{\})$

$\langle \text{proof} \rangle$

**lemma** *iT-Div-0*:  $I \neq \{\} \implies I \circ 0 = [\dots 0]$

$\langle \text{proof} \rangle$

**corollary** *iT-Div-0-if*:  $I \circ 0 = (\text{if } I = \{\} \text{ then } \{\} \text{ else } [\dots 0])$

$\langle \text{proof} \rangle$

**corollary**

*iFROM-div-0*:  $[n\dots] \circ 0 = [\dots 0]$  **and**

*iTILL-div-0*:  $[\dots n] \circ 0 = [\dots 0]$  **and**

*iIN-div-0*:  $[n\dots, d] \circ 0 = [\dots 0]$  **and**

*iMOD-div-0*:  $[r, \text{mod } m] \circ 0 = [\dots 0]$  **and**

*iMODb-div-0*:  $[r, \text{mod } m, c] \circ 0 = [\dots 0]$

$\langle \text{proof} \rangle$

**lemmas** *iT-div-0* =

*iTILL-div-0*

*iFROM-div-0*

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

**lemma** *iT-Div-1*:  $I \circ \text{Suc } 0 = I$   
 ⟨proof⟩

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

**lemma** *iT-Div-mem-iff*:  
 $0 < k \implies x \in (I \circ k) = (\exists y \in I. y \text{ div } k = x)$   
 ⟨proof⟩

**lemma** *iT-Div-mem-iff2*:  
 $0 < k \implies x \text{ div } k \in (I \circ k) = (\exists y \in I. y \text{ div } k = x \text{ div } k)$   
 ⟨proof⟩

**lemma** *iT-Div-mem-iff-Int*:  
 $0 < k \implies x \in (I \circ k) = (I \cap [x * k \dots k - \text{Suc } 0] \neq \{\})$   
 ⟨proof⟩

**lemma** *iT-Div-imp-mem*:  
 $0 < k \implies x \in I \implies x \text{ div } k \in (I \circ k)$   
 ⟨proof⟩

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

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

**lemma** *iT-Div-insert*:  $(\text{insert } n \ I) \circ k = \text{insert } (n \text{ div } k) \ (I \circ k)$   
 ⟨proof⟩

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

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

**lemma** *iFROM-iT-Div-Int*:  
 $\llbracket 0 < k; n \leq iMin \ A \rrbracket \implies (A \cap [n \dots]) \circ k = (A \circ k) \cap ([n \dots] \circ k)$   
 ⟨proof⟩

**lemma** *iIN-iT-Div-Int*:  
 $\llbracket 0 < k; n \leq iMin \ A; \forall x \in A. x \text{ div } k \leq (n + d) \text{ div } k \implies x \leq n + d \rrbracket \implies$



$$(A \cap [n..d]) \otimes k = (A \otimes k) \cap ([n..d] \otimes k)$$

*<proof>*

**corollary** *iTILL-iT-Div-Int:*

$$\llbracket 0 < k; \forall x \in A. x \text{ div } k \leq n \text{ div } k \longrightarrow x \leq n \rrbracket \implies$$

$$(A \cap [..n]) \otimes k = (A \otimes k) \cap ([..n] \otimes k)$$

*<proof>*

**lemma** *iIN-iT-Div-Int-mod-0:*

$$\llbracket 0 < k; n \bmod k = 0; \forall x \in A. x \text{ div } k \leq (n + d) \text{ div } k \longrightarrow x \leq n + d \rrbracket \implies$$

$$(A \cap [n..d]) \otimes k = (A \otimes k) \cap ([n..d] \otimes k)$$

*<proof>*

**lemma** *mod-partition-iT-Div-Int:*

$$\llbracket 0 < k; 0 < d \rrbracket \implies$$

$$(A \cap [n * k..d * k - \text{Suc } 0]) \otimes k =$$

$$(A \otimes k) \cap ([n * k..d * k - \text{Suc } 0] \otimes k)$$

*<proof><proof>*

**corollary** *mod-partition-iT-Div-Int2:*

$$\llbracket 0 < k; 0 < d; n \bmod k = 0; d \bmod k = 0 \rrbracket \implies$$

$$(A \cap [n..d - \text{Suc } 0]) \otimes k =$$

$$(A \otimes k) \cap ([n..d - \text{Suc } 0] \otimes k)$$

*<proof>*

**corollary** *mod-partition-iT-Div-Int-one-segment:*

$$0 < k \implies$$

$$(A \cap [n * k..k - \text{Suc } 0]) \otimes k = (A \otimes k) \cap ([n * k..k - \text{Suc } 0] \otimes k)$$

*<proof>*

**corollary** *mod-partition-iT-Div-Int-one-segment2:*

$$\llbracket 0 < k; n \bmod k = 0 \rrbracket \implies$$

$$(A \cap [n..k - \text{Suc } 0]) \otimes k = (A \otimes k) \cap ([n..k - \text{Suc } 0] \otimes k)$$

*<proof>*

**lemma** *iT-Div-assoc:*  $I \otimes a \otimes b = I \otimes (a * b)$

*<proof>*

**lemma** *iT-Div-commute:*  $I \otimes a \otimes b = I \otimes b \otimes a$

*<proof>*

**lemma** *iT-Mult-Div-self:*  $0 < k \implies I \otimes k \otimes k = I$

*<proof>*

**lemma** *iT-Mult-Div:*

$$\llbracket 0 < d; k \bmod d = 0 \rrbracket \implies I \otimes k \otimes d = I \otimes (k \text{ div } d)$$

*<proof>*

**lemma** *iT-Div-Mult-self:*

$$0 < k \implies I \otimes k \otimes k = \{y. \exists x \in I. y = x - x \bmod k\}$$

*<proof>*

**lemma** *iT-Plus-Div-distrib-mod-less*:

$$\forall x \in I. x \bmod m + n \bmod m < m \implies I \oplus n \otimes m = I \otimes m \oplus n \operatorname{div} m$$

*<proof>*

**corollary** *iT-Plus-Div-distrib-mod-0*:

$$n \bmod m = 0 \implies I \oplus n \otimes m = I \otimes m \oplus n \operatorname{div} m$$

*<proof>*

**lemma** *iT-Div-Min*:  $I \neq \{\}$   $\implies iMin (I \otimes k) = iMin I \operatorname{div} k$

*<proof>*

**corollary**

*iFROM-div-Min*:  $iMin ([n..] \otimes k) = n \operatorname{div} k$  **and**

*iIN-div-Min*:  $iMin ([n..,d] \otimes k) = n \operatorname{div} k$  **and**

*iTILL-div-Min*:  $iMin ([..n] \otimes k) = 0$  **and**

*iMOD-div-Min*:  $iMin ([r, \operatorname{mod} m] \otimes k) = r \operatorname{div} k$  **and**

*iMODb-div-Min*:  $iMin ([r, \operatorname{mod} m, c] \otimes k) = r \operatorname{div} k$

*<proof>*

**lemmas** *iT-div-Min* =

*iFROM-div-Min*

*iIN-div-Min*

*iTILL-div-Min*

*iMOD-div-Min*

*iMODb-div-Min*

**lemma** *iT-Div-Max*:  $\llbracket \text{finite } I; I \neq \{\} \rrbracket \implies \operatorname{Max} (I \otimes k) = \operatorname{Max} I \operatorname{div} k$

*<proof>*

**corollary**

*iIN-div-Max*:  $\operatorname{Max} ([n..,d] \otimes k) = (n + d) \operatorname{div} k$  **and**

*iTILL-div-Max*:  $\operatorname{Max} ([..n] \otimes k) = n \operatorname{div} k$  **and**

*iMODb-div-Max*:  $\operatorname{Max} ([r, \operatorname{mod} m, c] \otimes k) = (r + m * c) \operatorname{div} k$

*<proof>*

**lemma** *iT-Div-0-finite*:  $\text{finite} (I \otimes 0)$

*<proof>*

**lemma** *iT-Div-infinite-iff*:  $0 < k \implies \text{infinite} (I \otimes k) = \text{infinite } I$

*<proof>*

**lemma** *iT-Div-finite-iff*:  $0 < k \implies \text{finite} (I \otimes k) = \text{finite } I$

*<proof>*

**lemma** *iFROM-div*:  $0 < k \implies [n..] \otimes k = [n \operatorname{div} k..]$

*<proof>*

**lemma** *iIN-div*:

$$0 < k \implies$$

$$[n..,d] \otimes k = [n \operatorname{div} k.., d \operatorname{div} k + (n \bmod k + d \bmod k) \operatorname{div} k ]$$

$\langle \text{proof} \rangle$

**corollary** *iIN-div-if*:

$$0 < k \implies [n \dots, d] \odot k = [n \operatorname{div} k \dots, d \operatorname{div} k + (\text{if } n \operatorname{mod} k + d \operatorname{mod} k < k \text{ then } 0 \text{ else } \operatorname{Suc} 0)]$$

$\langle \text{proof} \rangle$

**corollary** *iIN-div-eq1*:

$$\llbracket 0 < k; n \operatorname{mod} k + d \operatorname{mod} k < k \rrbracket \implies [n \dots, d] \odot k = [n \operatorname{div} k \dots, d \operatorname{div} k]$$

$\langle \text{proof} \rangle$

**corollary** *iIN-div-eq2*:

$$\llbracket 0 < k; k \leq n \operatorname{mod} k + d \operatorname{mod} k \rrbracket \implies [n \dots, d] \odot k = [n \operatorname{div} k \dots, \operatorname{Suc} (d \operatorname{div} k)]$$

$\langle \text{proof} \rangle$

**corollary** *iIN-div-mod-eq-0*:

$$\llbracket 0 < k; n \operatorname{mod} k = 0 \rrbracket \implies [n \dots, d] \odot k = [n \operatorname{div} k \dots, d \operatorname{div} k]$$

$\langle \text{proof} \rangle$

**lemma** *iTILL-div*:

$$0 < k \implies [\dots n] \odot k = [\dots n \operatorname{div} k]$$

$\langle \text{proof} \rangle$

**lemma** *iMOD-div-ge*:

$$\llbracket 0 < m; m \leq k \rrbracket \implies [r, \operatorname{mod} m] \odot k = [r \operatorname{div} k \dots]$$

$\langle \text{proof} \rangle$

**corollary** *iMOD-div-self*:

$$0 < m \implies [r, \operatorname{mod} m] \odot m = [r \operatorname{div} m \dots]$$

$\langle \text{proof} \rangle$

**lemma** *iMOD-div*:

$$\llbracket 0 < k; m \operatorname{mod} k = 0 \rrbracket \implies [r, \operatorname{mod} m] \odot k = [r \operatorname{div} k, \operatorname{mod} (m \operatorname{div} k)]$$

$\langle \text{proof} \rangle$

**lemma** *iMODb-div-self*:

$$0 < m \implies [r, \operatorname{mod} m, c] \odot m = [r \operatorname{div} m \dots, c]$$

$\langle \text{proof} \rangle$

**lemma** *iMODb-div-ge*:

$$\llbracket 0 < m; m \leq k \rrbracket \implies [r, \operatorname{mod} m, c] \odot k = [r \operatorname{div} k \dots, (r + m * c) \operatorname{div} k - r \operatorname{div} k]$$

$\langle \text{proof} \rangle$

**corollary** *iMODb-div-ge-if*:

$$\llbracket 0 < m; m \leq k \rrbracket \implies$$

$[r, \text{mod } m, c] \odot k =$   
 $[r \text{ div } k \dots, m * c \text{ div } k + (\text{if } r \text{ mod } k + m * c \text{ mod } k < k \text{ then } 0 \text{ else } \text{Suc } 0)]$   
 <proof>

**lemma** *iMODb-div*:

$\llbracket 0 < k; m \text{ mod } k = 0 \rrbracket \implies$   
 $[r, \text{mod } m, c] \odot k = [r \text{ div } k, \text{mod } (m \text{ div } k), c]$   
 <proof>

**lemmas** *iT-div =*

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

This lemma is valid for all  $k \leq m$ , i. e., also for  $k$  with  $m \text{ mod } k \neq (0::'a)$ .

**lemma** *iMODb-div-unique*:

$\llbracket 0 < k; k \leq m; k \leq c; [r', \text{mod } m', c'] = [r, \text{mod } m, c] \odot k \rrbracket \implies$   
 $r' = r \text{ div } k \wedge m' = m \text{ div } k \wedge c' = c$   
 <proof>

**lemma** *iMODb-div-mod-gr0-is-0-not-ex0*:

$\llbracket 0 < k; k < m; 0 < m \text{ mod } k; k \leq c; r \text{ mod } k = 0 \rrbracket \implies$   
 $\neg(\exists r' m' c'. [r', \text{mod } m', c'] = [r, \text{mod } m, c] \odot k)$   
 <proof>

**lemma** *iMODb-div-mod-gr0-not-ex--arith-aux1*:

$\llbracket (0::\text{nat}) < k; k < m; 0 < x1 \rrbracket \implies$   
 $x1 * m + x2 - x \text{ mod } k + x3 + x \text{ mod } k = x1 * m + x2 + x3$   
 <proof>

**lemma** *iMODb-div-mod-gr0-not-ex*:

$\llbracket 0 < k; k < m; 0 < m \text{ mod } k; k \leq c \rrbracket \implies$   
 $\neg(\exists r' m' c'. [r', \text{mod } m', c'] = [r, \text{mod } m, c] \odot k)$   
 <proof>

**lemma** *iMOD-div-eq-imp-iMODb-div-eq*:

$\llbracket 0 < k; k \leq m; [r', \text{mod } m'] = [r, \text{mod } m] \odot k \rrbracket \implies$   
 $[r', \text{mod } m', c] = [r, \text{mod } m, c] \odot k$   
 <proof>

**lemma** *iMOD-div-unique*:

$\llbracket 0 < k; k \leq m; [r', \text{mod } m'] = [r, \text{mod } m] \odot k \rrbracket \implies$   
 $r' = r \text{ div } k \wedge m' = m \text{ div } k$

$\langle \text{proof} \rangle$

**lemma** *iMOD-div-mod-gr0-not-ex*:

$$\llbracket 0 < k; k < m; 0 < m \bmod k \rrbracket \implies \\ \neg (\exists r' m'. [r', \bmod m'] = [r, \bmod m] \otimes k)$$

$\langle \text{proof} \rangle$

## 2.2 Interval cut operators with arithmetic interval operators

**lemma**

$$\begin{aligned} \textit{iT-Plus-cut-le2}: & \quad (I \oplus k) \downarrow_{\leq} (t + k) = (I \downarrow_{\leq} t) \oplus k \textbf{ and} \\ \textit{iT-Plus-cut-less2}: & \quad (I \oplus k) \downarrow_{<} (t + k) = (I \downarrow_{<} t) \oplus k \textbf{ and} \\ \textit{iT-Plus-cut-ge2}: & \quad (I \oplus k) \downarrow_{\geq} (t + k) = (I \downarrow_{\geq} t) \oplus k \textbf{ and} \\ \textit{iT-Plus-cut-greater2}: & \quad (I \oplus k) \downarrow_{>} (t + k) = (I \downarrow_{>} t) \oplus k \end{aligned}$$

$\langle \text{proof} \rangle$

**lemma** *iT-Plus-cut-le*:

$$(I \oplus k) \downarrow_{\leq} t = (\textit{if } t < k \textit{ then } \{\} \textit{ else } I \downarrow_{\leq} (t - k) \oplus k)$$

$\langle \text{proof} \rangle$

**lemma** *iT-Plus-cut-less*:  $(I \oplus k) \downarrow_{<} t = I \downarrow_{<} (t - k) \oplus k$

$\langle \text{proof} \rangle$

**lemma** *iT-Plus-cut-ge*:  $(I \oplus k) \downarrow_{\geq} t = I \downarrow_{\geq} (t - k) \oplus k$

$\langle \text{proof} \rangle$

**lemma** *iT-Plus-cut-greater*:

$$(I \oplus k) \downarrow_{>} t = (\textit{if } t < k \textit{ then } I \oplus k \textit{ else } I \downarrow_{>} (t - k) \oplus k)$$

$\langle \text{proof} \rangle$

**lemma**

$$\begin{aligned} \textit{iT-Mult-cut-le2}: & \quad 0 < k \implies (I \otimes k) \downarrow_{\leq} (t * k) = (I \downarrow_{\leq} t) \otimes k \textbf{ and} \\ \textit{iT-Mult-cut-less2}: & \quad 0 < k \implies (I \otimes k) \downarrow_{<} (t * k) = (I \downarrow_{<} t) \otimes k \textbf{ and} \\ \textit{iT-Mult-cut-ge2}: & \quad 0 < k \implies (I \otimes k) \downarrow_{\geq} (t * k) = (I \downarrow_{\geq} t) \otimes k \textbf{ and} \\ \textit{iT-Mult-cut-greater2}: & \quad 0 < k \implies (I \otimes k) \downarrow_{>} (t * k) = (I \downarrow_{>} t) \otimes k \end{aligned}$$

$\langle \text{proof} \rangle$

**lemma** *iT-Mult-cut-le*:

$$0 < k \implies (I \otimes k) \downarrow_{\leq} t = (I \downarrow_{\leq} (t \textit{ div } k)) \otimes k$$

$\langle \text{proof} \rangle$

**lemma** *iT-Mult-cut-less*:

$$0 < k \implies (I \otimes k) \downarrow_{<} t = \\ (\textit{if } t \bmod k = 0 \textit{ then } (I \downarrow_{<} (t \textit{ div } k)) \textit{ else } I \downarrow_{<} \textit{Suc } (t \textit{ div } k)) \otimes k$$

$\langle \text{proof} \rangle$

**lemma** *iT-Mult-cut-greater*:

$$0 < k \implies (I \otimes k) \downarrow_{>} t = (I \downarrow_{>} (t \textit{ div } k)) \otimes k$$

*<proof>*

**lemma** *iT-Mult-cut-ge:*

$$0 < k \implies (I \otimes k) \downarrow_{\geq} t =$$

$$(if\ t\ mod\ k = 0\ then\ (I \downarrow_{\geq} (t \div k))\ else\ I \downarrow_{\geq} Suc\ (t \div k)) \otimes k$$

*<proof>*

**lemma** *iT-Plus-neg-cut-le2:*  $k \leq t \implies (I \oplus - k) \downarrow_{\leq} (t - k) = (I \downarrow_{\leq} t) \oplus - k$

*<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:*  $k \leq t \implies (I \oplus - k) \downarrow_{>} (t - k) = (I \downarrow_{>} t) \oplus - k$

*<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:*  $t \leq k \implies (k \ominus I) \downarrow_{\leq} (k - t) = k \ominus (I \downarrow_{\geq} t)$

*<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:*  $t \leq k \implies (k \ominus I) \downarrow_{>} (k - t) = k \ominus (I \downarrow_{<} t)$

*<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:*

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

**lemma** *iT-Minus-cut-ge*:

$(k \ominus I) \downarrow \geq t = (\text{if } t \leq k \text{ then } k \ominus (I \downarrow \leq (k - t)) \text{ else } \{\})$   
 ⟨proof⟩

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

**lemma** *iT-Div-cut-le*:

$0 < k \implies (I \circledast k) \downarrow \leq t = I \downarrow \leq (t * k + (k - \text{Suc } 0)) \circledast k$   
 ⟨proof⟩

**lemma** *iT-Div-cut-less*:

$0 < k \implies (I \circledast k) \downarrow < t = I \downarrow < (t * k) \circledast k$   
 ⟨proof⟩

**lemma** *iT-Div-cut-ge*:

$0 < k \implies (I \circledast k) \downarrow \geq t = I \downarrow \geq (t * k) \circledast k$   
 ⟨proof⟩

**lemma** *iT-Div-cut-greater*:

$0 < k \implies (I \circledast k) \downarrow > t = I \downarrow > (t * k + (k - \text{Suc } 0)) \circledast k$   
 ⟨proof⟩

**lemma** *iT-Div-cut-le2*:

$0 < k \implies (I \circledast k) \downarrow \leq (t \text{ div } k) = I \downarrow \leq (t - t \text{ mod } k + (k - \text{Suc } 0)) \circledast k$   
 ⟨proof⟩

**lemma** *iT-Div-cut-less2*:

$0 < k \implies (I \circledast k) \downarrow < (t \text{ div } k) = I \downarrow < (t - t \text{ mod } k) \circledast k$   
 ⟨proof⟩

**lemma** *iT-Div-cut-ge2*:

$0 < k \implies (I \circledast k) \downarrow \geq (t \text{ div } k) = I \downarrow \geq (t - t \text{ mod } k) \circledast k$   
 ⟨proof⟩

**lemma** *iT-Div-cut-greater2*:

$0 < k \implies (I \circledast k) \downarrow > (t \text{ div } k) = I \downarrow > (t - t \text{ mod } k + (k - \text{Suc } 0)) \circledast k$   
 ⟨proof⟩

### 2.3 *inext* and *iprev* with interval operators

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

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

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

**lemma** *iT-Plus-prev2*:  $k \leq n \implies iprev n (I \oplus k) = (iprev (n - k) I) + k$   
 ⟨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*:  
 $inext n (I \otimes k) = (if n \text{ mod } k = 0 \text{ then } (inext (n \text{ div } k) I) * k \text{ else } n)$   
 ⟨proof⟩

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

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

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

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

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

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

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

**lemma** *iT-Minus-inext*:  
 $\llbracket k \ominus I \neq \{\}; n \leq k \rrbracket \implies inext (k - n) (k \ominus I) = k - iprev n I$   
 ⟨proof⟩



**corollary** *iT-Minus-inext2*:

$$\llbracket k \ominus I \neq \{\}; n \leq k \rrbracket \implies \text{inext } n (k \ominus I) = k - \text{iprev } (k - n) I$$

*<proof>*

**lemma** *iT-Minus-iprev*:

$$\llbracket k \ominus I \neq \{\}; n \leq k \rrbracket \implies \text{iprev } (k - n) (k \ominus I) = k - \text{inext } n (I \downarrow \leq k)$$

*<proof>*

**lemma** *iT-Minus-iprev2*:

$$\llbracket k \ominus I \neq \{\}; n \leq k \rrbracket \implies \text{iprev } n (k \ominus I) = k - \text{inext } (k - n) (I \downarrow \leq k)$$

*<proof>*

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

*<proof>*

**lemma** *iT-Plus-iprev-nth*:  $\llbracket \text{finite } I; I \neq \{\} \rrbracket \implies (I \oplus k) \leftarrow n = (I \leftarrow n) + k$

*<proof>*

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

*<proof>*

**lemma** *iT-Mult-iprev-nth*:  $\llbracket \text{finite } I; I \neq \{\} \rrbracket \implies (I \otimes k) \leftarrow n = (I \leftarrow n) * k$

*<proof>*

**lemma** *iT-Plus-neg-inext-nth*:

$$I \oplus - k \neq \{\} \implies (I \oplus - k) \rightarrow n = (I \downarrow \geq k \rightarrow n) - k$$

*<proof>*

**lemma** *iT-Plus-neg-iprev-nth*:

$$\llbracket \text{finite } I; I \oplus - k \neq \{\} \rrbracket \implies (I \oplus - k) \leftarrow n = (I \downarrow \geq k \leftarrow n) - k$$

*<proof>*

**lemma** *iT-Minus-inext-nth*:

$$k \ominus I \neq \{\} \implies (k \ominus I) \rightarrow n = k - ((I \downarrow \leq k) \leftarrow n)$$

*<proof>*

**lemma** *iT-Minus-iprev-nth*:

$$k \ominus I \neq \{\} \implies (k \ominus I) \leftarrow n = k - ((I \downarrow \leq k) \rightarrow n)$$

*<proof>*

**lemma** *iT-Div-ge-inext-nth*:

$$\llbracket I \neq \{\}; \forall x \in I. \forall y \in I. x < y \implies x + k \leq y \rrbracket \implies$$

$$(I \oslash k) \rightarrow n = (I \rightarrow n) \text{ div } k$$

*<proof>*

**lemma** *iT-Div-mod-inext-nth*:

$$\llbracket I \neq \{\}; \forall x \in I. \forall y \in I. x \text{ mod } k = y \text{ mod } k \rrbracket \implies$$

$(I \otimes k) \rightarrow n = (I \rightarrow n) \text{ div } k$   
 ⟨proof⟩

**lemma** *iT-Div-ge-iprev-nth*:

$\llbracket \text{finite } I; I \neq \{\}; \forall x \in I. \forall y \in I. x < y \longrightarrow x + k \leq y \rrbracket \implies$   
 $(I \otimes k) \leftarrow n = (I \leftarrow n) \text{ div } k$   
 ⟨proof⟩

**lemma** *iT-Div-mod-iprev-nth*:

$\llbracket \text{finite } I; I \neq \{\}; \forall x \in I. \forall y \in I. x \bmod k = y \bmod k \rrbracket \implies$   
 $(I \otimes k) \leftarrow n = (I \leftarrow n) \text{ div } k$   
 ⟨proof⟩

## 2.4 Cardinality of intervals with interval operators

**lemma** *iT-Plus-card*:  $\text{card } (I \oplus k) = \text{card } I$

⟨proof⟩

**lemma** *iT-Mult-card*:  $0 < k \implies \text{card } (I \otimes k) = \text{card } I$

⟨proof⟩

**lemma** *iT-Plus-neg-card*:  $\text{card } (I \oplus - k) = \text{card } (I \downarrow \geq k)$

⟨proof⟩

**lemma** *iT-Plus-neg-card-le*:  $\text{card } (I \oplus - k) \leq \text{card } I$

⟨proof⟩

**lemma** *iT-Minus-card*:  $\text{card } (k \ominus I) = \text{card } (I \downarrow \leq k)$

⟨proof⟩

**lemma** *iT-Minus-card-le*:  $\text{finite } I \implies \text{card } (k \ominus I) \leq \text{card } I$

⟨proof⟩

**lemma** *iT-Div-0-card-if*:

$\text{card } (I \otimes 0) = (\text{if } I = \{\} \text{ then } 0 \text{ else } \text{Suc } 0)$   
 ⟨proof⟩

**lemma** *Int-empty-sum*:

$(\sum_{k \leq (n :: \text{nat})} \text{if } \{\} \cap (I k) = \{\} \text{ then } 0 \text{ else } \text{Suc } 0) = 0$   
 ⟨proof⟩

**lemma** *iT-Div-mod-partition-card*:

$\text{card } (I \cap [n * d \dots, d - \text{Suc } 0] \otimes d) =$   
 $(\text{if } I \cap [n * d \dots, d - \text{Suc } 0] = \{\} \text{ then } 0 \text{ else } \text{Suc } 0)$   
 ⟨proof⟩

**lemma** *iT-Div-conv-count*:

$0 < d \implies I \otimes d = \{k. I \cap [k * d \dots, d - \text{Suc } 0] \neq \{\}\}$   
 ⟨proof⟩

**lemma** *iT-Div-conv-count2*:

$\llbracket 0 < d; \text{finite } I; \text{Max } I \text{ div } d \leq n \rrbracket \implies$   
 $I \circ d = \{k. k \leq n \wedge I \cap [k * d \dots, d - \text{Suc } 0] \neq \{\}\}$   
 ⟨proof⟩

**lemma** *mod-partition-count-Suc*:

$\{k. k \leq \text{Suc } n \wedge I \cap [k * d \dots, d - \text{Suc } 0] \neq \{\}\} =$   
 $\{k. k \leq n \wedge I \cap [k * d \dots, d - \text{Suc } 0] \neq \{\}\} \cup$   
 (if  $I \cap [\text{Suc } n * d \dots, d - \text{Suc } 0] \neq \{\}$  then  $\{\text{Suc } n\}$  else  $\{\}$ )  
 ⟨proof⟩

**lemma** *iT-Div-card*:

$\bigwedge I. \llbracket 0 < d; \text{finite } I; \text{Max } I \text{ div } d \leq n \rrbracket \implies$   
 $\text{card } (I \circ d) = (\sum k \leq n.$   
 if  $I \cap [k * d \dots, d - \text{Suc } 0] = \{\}$  then 0 else  $\text{Suc } 0$ )  
 ⟨proof⟩

**corollary** *iT-Div-card-Suc*:

$\bigwedge I. \llbracket 0 < d; \text{finite } I; \text{Max } I \text{ div } d \leq n \rrbracket \implies$   
 $\text{card } (I \circ d) = (\sum k < \text{Suc } n.$   
 if  $I \cap [k * d \dots, d - \text{Suc } 0] = \{\}$  then 0 else  $\text{Suc } 0$ )  
 ⟨proof⟩

**corollary** *iT-Div-Max-card*:  $\llbracket 0 < d; \text{finite } I \rrbracket \implies$

$\text{card } (I \circ d) = (\sum k \leq \text{Max } I \text{ div } d.$   
 if  $I \cap [k * d \dots, d - \text{Suc } 0] = \{\}$  then 0 else  $\text{Suc } 0$ )  
 ⟨proof⟩

**lemma** *iT-Div-card-le*:  $0 < k \implies \text{card } (I \circ k) \leq \text{card } I$

⟨proof⟩

**lemma** *iT-Div-card-inj-on*:

*inj-on*  $(\lambda n. n \text{ div } k) I \implies \text{card } (I \circ k) = \text{card } I$   
 ⟨proof⟩

**lemma** *mod-Suc'*:

$0 < n \implies \text{Suc } m \text{ mod } n = (\text{if } m \text{ mod } n < n - \text{Suc } 0 \text{ then } \text{Suc } (m \text{ mod } n) \text{ else } 0)$   
 ⟨proof⟩

**lemma** *div-Suc*:

$0 < n \implies \text{Suc } m \text{ div } n = (\text{if } \text{Suc } (m \text{ mod } n) = n \text{ then } \text{Suc } (m \text{ div } n) \text{ else } m \text{ div } n)$   
 ⟨proof⟩

**lemma** *div-Suc'*:

$0 < n \implies \text{Suc } m \text{ div } n = (\text{if } m \text{ mod } n < n - \text{Suc } 0 \text{ then } m \text{ div } n \text{ else } \text{Suc } (m \text{ div } n))$   
 ⟨proof⟩

**lemma** *iT-Div-card-ge-aux*:

$\bigwedge I. \llbracket 0 < d; \text{finite } I; \text{Max } I \text{ div } d \leq n \rrbracket \implies$   
 $\text{card } I \text{ div } d + (\text{if } \text{card } I \text{ mod } d = 0 \text{ then } 0 \text{ else } \text{Suc } 0) \leq \text{card } (I \circlearrowleft d)$   
 ⟨proof⟩

**lemma** *iT-Div-card-ge*:

$\text{card } I \text{ div } d + (\text{if } \text{card } I \text{ mod } d = 0 \text{ then } 0 \text{ else } \text{Suc } 0) \leq \text{card } (I \circlearrowleft d)$   
 ⟨proof⟩

**corollary** *iT-Div-card-ge-div*:  $\text{card } I \text{ div } d \leq \text{card } (I \circlearrowleft d)$

⟨proof⟩

There is no better lower bound function  $f$  for  $i \circlearrowleft d$  with  $\text{card } i$  and  $d$  as arguments.

**lemma** *iT-Div-card-ge--is-maximal-lower-bound*:

$\forall I d. \text{card } I \text{ div } d + (\text{if } \text{card } I \text{ mod } d = 0 \text{ then } 0 \text{ else } \text{Suc } 0) \leq f (\text{card } I) d \wedge$   
 $f (\text{card } I) d \leq \text{card } (I \circlearrowleft d) \implies$   
 $f (\text{card } (I::\text{nat set})) d = \text{card } I \text{ div } d + (\text{if } \text{card } I \text{ mod } d = 0 \text{ then } 0 \text{ else } \text{Suc } 0)$   
 ⟨proof⟩

**lemma** *iT-Plus-icard*:  $\text{icard } (I \oplus k) = \text{icard } I$

⟨proof⟩

**lemma** *iT-Mult-icard*:  $0 < k \implies \text{icard } (I \otimes k) = \text{icard } I$

⟨proof⟩

**lemma** *iT-Plus-neg-icard*:  $\text{icard } (I \oplus - k) = \text{icard } (I \downarrow \geq k)$

⟨proof⟩

**lemma** *iT-Plus-neg-icard-le*:  $\text{icard } (I \oplus - k) \leq \text{icard } I$

⟨proof⟩

**lemma** *iT-Minus-icard*:  $\text{icard } (k \ominus I) = \text{icard } (I \downarrow \leq k)$

⟨proof⟩

**lemma** *iT-Minus-icard-le*:  $\text{icard } (k \ominus I) \leq \text{icard } I$

⟨proof⟩

**lemma** *iT-Div-0-icard-if*:  $\text{icard } (I \circlearrowleft 0) = \text{enat } (\text{if } I = \{\} \text{ then } 0 \text{ else } \text{Suc } 0)$

⟨proof⟩

**lemma** *iT-Div-mod-partition-icard*:

$\text{icard } (I \cap [n * d.., d - \text{Suc } 0] \circlearrowleft d) =$

$enat (if I \cap [n * d \dots, d - Suc 0] = \{\} then 0 else Suc 0)$   
 <proof>

**lemma** *iT-Div-icard*:

$\llbracket 0 < d; finite I \implies Max I div d \leq n \rrbracket \implies$   
 $icard (I \circ d) =$   
 $(if finite I then enat (\sum k \leq n. if I \cap [k * d \dots, d - Suc 0] = \{\} then 0 else Suc$   
 $0) else \infty)$   
 <proof>

**corollary** *iT-Div-Max-icard*:  $0 < d \implies$

$icard (I \circ d) = (if finite I$   
 $then enat (\sum k \leq Max I div d. if I \cap [k * d \dots, d - Suc 0] = \{\} then 0 else Suc$   
 $0) else \infty)$   
 <proof>

**lemma** *iT-Div-icard-le*:  $0 < k \implies icard (I \circ k) \leq icard I$

<proof>

**lemma** *iT-Div-icard-inj-on*:  $inj-on (\lambda n. n div k) I \implies icard (I \circ k) = icard I$

<proof>

**lemma** *iT-Div-icard-ge*:  $icard I div (enat d) + enat (if icard I mod (enat d) = 0$   
 $then 0 else Suc 0) \leq icard (I \circ d)$

<proof>

**corollary** *iT-Div-icard-ge-div*:  $icard I div (enat d) \leq icard (I \circ d)$

<proof>

**lemma** *iT-Div-icard-ge-is-maximal-lower-bound*:

$\forall I d. icard I div (enat d) + enat (if icard I mod (enat d) = 0 then 0 else Suc 0)$   
 $\leq f (icard I) d \wedge$   
 $f (icard I) d \leq icard (I \circ d) \implies$   
 $f (icard (I :: nat set)) d =$   
 $icard I div (enat d) + enat (if icard I mod (enat d) = 0 then 0 else Suc 0)$   
 <proof>

## 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  $iFROM-UN-set \equiv \bigcup n. \{[n \dots]\}$

**definition** *iTILL-UN-set* ::  $(nat set) set$

where  $iTILL-UN-set \equiv \bigcup n. \{[..n]\}$

**definition** *iIN-UN-set* ::  $(nat set) set$

where  $iIN-UN-set \equiv \bigcup n d. \{[n \dots, d]\}$

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

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

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

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

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

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

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

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

**definition** *iMODb2-set* :: (nat set) set  
 where *iMODb2-set*  $\equiv \{[r, \text{mod } m, c] \mid r m c. 2 \leq m \wedge 1 \leq c\}$

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

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

**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*:  $iTILL\text{-set} = iTILL\text{-UN-set}$  **and**  
*iIN-set-UN-set-eq*:  $iIN\text{-set} = iIN\text{-UN-set}$  **and**  
*iMOD-set-UN-set-eq*:  $iMOD\text{-set} = iMOD\text{-UN-set}$  **and**  
*iMODb-set-UN-set-eq*:  $iMODb\text{-set} = iMODb\text{-UN-set}$

*<proof>*

**lemma** *iMOD2-set-UN-set-eq*:  $iMOD2\text{-set} = iMOD2\text{-UN-set}$

*<proof>*

**lemma** *iMODb2-set-UN-set-eq*:  $iMODb2\text{-set} = iMODb2\text{-UN-set}$

*<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*:  $iMOD2\text{-set} \subseteq iMOD\text{-set}$

*<proof>*

**lemma** *iMODb2-set-iMODb-set-subset*:  $iMODb2\text{-set} \subseteq iMODb\text{-set}$

*<proof>*

**definition** *i-set* :: (nat set) set

**where**  $i\text{-set} \equiv iFROM\text{-set} \cup iTILL\text{-set} \cup iIN\text{-set} \cup iMOD\text{-set} \cup iMODb\text{-set}$

**definition** *i-UN-set* :: (nat set) set

**where**  $i\text{-UN-set} \equiv iFROM\text{-UN-set} \cup iTILL\text{-UN-set} \cup iIN\text{-UN-set} \cup iMOD\text{-UN-set} \cup iMODb\text{-UN-set}$

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

**definition** *i-set-min* :: (nat set) set

**where**  $i\text{-set-min} \equiv iFROM\text{-set} \cup iIN\text{-set} \cup iMOD2\text{-set} \cup iMODb2\text{-set}$

**definition** *i-UN-set-min* :: (nat set) set

**where**  $i\text{-UN-set-min} \equiv iFROM\text{-UN-set} \cup iIN\text{-UN-set} \cup iMOD2\text{-UN-set} \cup iMODb2\text{-UN-set}$

**definition** *i-set0* :: (nat set) set

**where**  $i\text{-set0} \equiv \text{insert } \{\} \text{ } i\text{-set}$

**lemma** *i-set-i-UN-set-eq*:  $i\text{-set} = i\text{-UN-set}$

*<proof>*

**lemma** *i-set-min-i-UN-set-min-eq*:  $i\text{-set-min} = i\text{-UN-set-min}$   
 ⟨proof⟩

**lemma** *i-set-min-eq*:  $i\text{-set} = i\text{-set-min}$   
 ⟨proof⟩

**corollary** *i-UN-set-i-UN-min-set-eq*:  $i\text{-UN-set} = i\text{-UN-set-min}$   
 ⟨proof⟩

**lemma** *i-set-min-is-minimal-let*:  
 let  $s1 = i\text{FROM-set}$ ;  $s2 = i\text{IN-set}$ ;  $s3 = i\text{MOD2-set}$ ;  $s4 = i\text{MODb2-set}$  in  
 $s1 \cap s2 = \{\}$   $\wedge$   $s1 \cap s3 = \{\}$   $\wedge$   $s1 \cap s4 = \{\}$   $\wedge$   
 $s2 \cap s3 = \{\}$   $\wedge$   $s2 \cap s4 = \{\}$   $\wedge$   $s3 \cap s4 = \{\}$   
 ⟨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!]:  $[n..] \in i\text{-set-ind}$   
 | *i-set-ind-TILL*[intro!]:  $[..n] \in i\text{-set-ind}$   
 | *i-set-ind-IN*[intro!]:  $[n..,d] \in i\text{-set-ind}$   
 | *i-set-ind-MOD*[intro!]:  $[r, \text{mod } m] \in i\text{-set-ind}$   
 | *i-set-ind-MODb*[intro!]:  $[r, \text{mod } m, c] \in i\text{-set-ind}$

**inductive-set** *i-set0-ind* :: (nat set) set  
 where

*i-set0-ind-empty*[intro!]:  $\{\} \in i\text{-set0-ind}$   
 | *i-set0-ind-i-set*[intro!]:  $I \in i\text{-set-ind} \implies I \in i\text{-set0-ind}$

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*:  $i\text{-set-ind} \subseteq i\text{-set0-ind}$   
 ⟨proof⟩

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

**lemma** *i-set0-i-set0-ind-eq*:  $i\text{-set0} = i\text{-set0-ind}$   
 ⟨proof⟩

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

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

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

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

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

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

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

**corollary**

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

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

**lemmas**  $i\text{-set0-intros} =$

$i\text{-set0-empty}$   
 $i\text{-set0-FROM}$   
 $i\text{-set0-TILL}$   
 $i\text{-set0-IN}$   
 $i\text{-set0-MOD}$   
 $i\text{-set0-MODb}$

**lemma**  $i\text{-set-infinite-as-iMOD}$ :

$\llbracket I \in i\text{-set}; \text{infinite } I \rrbracket \implies \exists r m. I = [r, \text{mod } m]$   
 ⟨proof⟩

**lemma**  $i\text{-set-finite-as-iMODb}$ :

$\llbracket I \in i\text{-set}; \text{finite } I \rrbracket \implies \exists r m c. I = [r, \text{mod } m, c]$   
 ⟨proof⟩

**lemma**  $i\text{-set-as-iMOD-iMODb}$ :

$I \in i\text{-set} \implies (\exists r m. I = [r, \text{mod } m]) \vee (\exists r m c. I = [r, \text{mod } m, c])$   
 ⟨proof⟩

## 2.5.2 Interval sets are closed under cutting

**lemma**  $i\text{-set-cut-le-ge-closed-disj}$ :

$\llbracket I \in i\text{-set}; t \in I; \text{cut-op} = (\downarrow \leq) \vee \text{cut-op} = (\downarrow \geq) \rrbracket \implies$   
 $\text{cut-op } I t \in i\text{-set}$   
 ⟨proof⟩

**corollary**

$i\text{-set-cut-le-closed}$ :  $\llbracket I \in i\text{-set}; t \in I \rrbracket \implies I \downarrow \leq t \in i\text{-set}$  **and**  
 $i\text{-set-cut-ge-closed}$ :  $\llbracket I \in i\text{-set}; t \in I \rrbracket \implies I \downarrow \geq t \in i\text{-set}$   
 ⟨proof⟩

**lemmas**  $i\text{-set-cut-le-ge-closed} = i\text{-set-cut-le-closed } i\text{-set-cut-ge-closed}$

**lemma**  $i\text{-set0-cut-closed-disj}$ :

$\llbracket I \in i\text{-set0};$   
 $\text{cut-op} = (\downarrow \leq) \vee \text{cut-op} = (\downarrow \geq) \vee$   
 $\text{cut-op} = (\downarrow <) \vee \text{cut-op} = (\downarrow >) \rrbracket \implies$   
 $\text{cut-op } I t \in i\text{-set0}$   
 ⟨proof⟩

**corollary**

*i-set0-cut-le-closed*:  $I \in i\text{-set0} \implies I \downarrow \leq t \in i\text{-set0}$  **and**  
*i-set0-cut-less-closed*:  $I \in i\text{-set0} \implies I \downarrow < t \in i\text{-set0}$  **and**  
*i-set0-cut-ge-closed*:  $I \in i\text{-set0} \implies I \downarrow \geq t \in i\text{-set0}$  **and**  
*i-set0-cut-greater-closed*:  $I \in i\text{-set0} \implies I \downarrow > t \in i\text{-set0}$   
 ⟨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*:  $I \in i\text{-set} \implies I \oplus k \in i\text{-set}$   
 ⟨proof⟩

**lemma** *i-set-Mult-closed*:  $I \in i\text{-set} \implies I \otimes k \in i\text{-set}$   
 ⟨proof⟩

**lemma** *i-set0-Plus-closed*:  $I \in i\text{-set0} \implies I \oplus k \in i\text{-set0}$   
 ⟨proof⟩

**lemma** *i-set0-Mult-closed*:  $I \in i\text{-set0} \implies I \otimes k \in i\text{-set0}$   
 ⟨proof⟩

### 2.5.4 Interval sets are closed with certain conditions under subtraction

**lemma** *i-set-Plus-neg-closed*:  
 $\llbracket I \in i\text{-set}; \exists x \in I. k \leq x \rrbracket \implies I \oplus -k \in i\text{-set}$   
 ⟨proof⟩

**lemma** *i-set-Minus-closed*:  
 $\llbracket I \in i\text{-set}; i\text{Min } I \leq k \rrbracket \implies k \ominus I \in i\text{-set}$   
 ⟨proof⟩

**lemma** *i-set0-Plus-neg-closed*:  $I \in i\text{-set0} \implies I \oplus -k \in i\text{-set0}$   
 ⟨proof⟩

**lemma** *i-set0-Minus-closed*:  $I \in i\text{-set0} \implies k \ominus I \in i\text{-set0}$   
 ⟨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*:  
 $\llbracket 0 < k; k < m; 0 < m \bmod k \rrbracket \implies [r, \bmod m] \circlearrowleft k \notin i\text{-set}$   
*<proof>*

**lemma** *iMODb-div-mod-gr0-not-in-i-set*:  
 $\llbracket 0 < k; k < m; 0 < m \bmod k; k \leq c \rrbracket \implies [r, \bmod m, c] \circlearrowleft k \notin i\text{-set}$   
*<proof>*

**lemma**  $[0, \bmod 3] \circlearrowleft 2 \notin i\text{-set}$   
*<proof>*

**lemma** *i-set-Div-not-closed*:  $\text{Suc } 0 < k \implies \exists I \in i\text{-set}. I \circlearrowleft k \notin i\text{-set}$   
*<proof>*

**lemma** *i-set0-Div-not-closed*:  $\text{Suc } 0 < k \implies \exists I \in i\text{-set0}. I \circlearrowleft k \notin i\text{-set0}$   
*<proof>*

### 2.5.6 Sets of intervals closed under division

**inductive-set** *NatMultiples* :: *nat set*  $\Rightarrow$  *nat set*  
**for** *F* :: *nat set*

**where**

*NatMultiples-Factor*:  $k \in F \implies k \in \text{NatMultiples } F$

*NatMultiples-Product*:  $\llbracket k \in F; m \in \text{NatMultiples } F \rrbracket \implies k * m \in \text{NatMultiples } F$

**lemma** *NatMultiples-ex-divisor*:  $m \in \text{NatMultiples } F \implies \exists k \in F. m \bmod k = 0$   
*<proof>*

**lemma** *NatMultiples-product-factor*:  $\llbracket a \in F; b \in F \rrbracket \implies a * b \in \text{NatMultiples } F$   
*<proof>*

**lemma** *NatMultiples-product-factor-multiple*:  
 $\llbracket a \in F; b \in \text{NatMultiples } F \rrbracket \implies a * b \in \text{NatMultiples } F$   
*<proof>*

**lemma** *NatMultiples-product-multiple-factor*:  
 $\llbracket a \in \text{NatMultiples } F; b \in F \rrbracket \implies a * b \in \text{NatMultiples } F$   
*<proof>*

**lemma** *NatMultiples-product-multiple*:  
 $\llbracket a \in \text{NatMultiples } F; b \in \text{NatMultiples } F \rrbracket \implies a * b \in \text{NatMultiples } F$   
*<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..] ∈ *i-set-cont*  
 | *i-set-cont-TILL*[intro]: [...n] ∈ *i-set-cont*  
 | *i-set-cont-IN*[intro]: [n..,d] ∈ *i-set-cont*

**definition** *i-set0-cont* :: (nat set) set

**where** *i-set0-cont* ≡ insert {} *i-set-cont*

**lemma** *i-set-cont-subset-i-set*: *i-set-cont* ⊆ *i-set*

⟨proof⟩

**lemma** *i-set0-cont-subset-i-set0*: *i-set0-cont* ⊆ *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..] ∈ *i-set-cont-min*  
 | *i-set-cont-IN*[intro]: [n..,d] ∈ *i-set-cont-min*

**definition** *i-set0-cont-min* :: (nat set) set

**where** *i-set0-cont-min* ≡ 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* ∈ *i-set-cont*; *n* ∈ *I*; finite *I* ⇒ *n* < Max *I* ]] ⇒ *inext* *n* *I* = Suc *n*

⟨proof⟩

**lemma** *i-set-cont-iprev*:

[[ *I* ∈ *i-set-cont*; *n* ∈ *I*; iMin *I* < *n* ]] ⇒ *iprev* *n* *I* = *n* - Suc 0

⟨proof⟩

**lemma** *i-set-cont-inext--less*:

[[ *I* ∈ *i-set-cont*; *n* ∈ *I*; *n* < *n0*; *n0* ∈ *I* ]] ⇒ *inext* *n* *I* = Suc *n*

⟨proof⟩

**lemma** *i-set-cont-iprev--greater*:

[[ *I* ∈ *i-set-cont*; *n* ∈ *I*; *n0* < *n*; *n0* ∈ *I* ]] ⇒ *iprev* *n* *I* = *n* - Suc 0

⟨proof⟩

Sets of modulo intervals

**inductive-set** *i-set-mult* :: nat ⇒ (nat set) set

**for**  $k :: \text{nat}$   
**where**  
 $i\text{-set-mult-FROM}[intro!]: [n..] \in i\text{-set-mult } k$   
 $| i\text{-set-mult-TILL}[intro!]: [..n] \in i\text{-set-mult } k$   
 $| i\text{-set-mult-IN}[intro!]: [n..,d] \in i\text{-set-mult } k$   
 $| i\text{-set-mult-MOD}[intro!]: [r, \text{mod } m * k] \in i\text{-set-mult } k$   
 $| i\text{-set-mult-MODb}[intro!]: [r, \text{mod } m * k, c] \in i\text{-set-mult } k$

**definition**  $i\text{-set0-mult} :: \text{nat} \Rightarrow (\text{nat set}) \text{ set}$   
**where**  $i\text{-set0-mult } k \equiv \text{insert } \{\} (i\text{-set-mult } k)$

**lemma**  
 $i\text{-set0-mult-empty}[intro!]: \{\} \in i\text{-set0-mult } k$  **and**  
 $i\text{-set0-mult-FROM}[intro!]: [n..] \in i\text{-set0-mult } k$  **and**  
 $i\text{-set0-mult-TILL}[intro!]: [..n] \in i\text{-set0-mult } k$  **and**  
 $i\text{-set0-mult-IN}[intro!]: [n..,d] \in i\text{-set0-mult } k$  **and**  
 $i\text{-set0-mult-MOD}[intro!]: [r, \text{mod } m * k] \in i\text{-set0-mult } k$  **and**  
 $i\text{-set0-mult-MODb}[intro!]: [r, \text{mod } m * k, c] \in i\text{-set0-mult } k$   
 $\langle \text{proof} \rangle$

**lemmas**  $i\text{-set0-mult-intros} =$   
 $i\text{-set0-mult-empty}$   
 $i\text{-set0-mult-FROM}$   
 $i\text{-set0-mult-TILL}$   
 $i\text{-set0-mult-IN}$   
 $i\text{-set0-mult-MOD}$   
 $i\text{-set0-mult-MODb}$

**lemma**  $i\text{-set-mult-subset-i-set0-mult}: i\text{-set-mult } k \subseteq i\text{-set0-mult } k$   
 $\langle \text{proof} \rangle$

**lemma**  $i\text{-set-mult-subset-i-set}: i\text{-set-mult } k \subseteq i\text{-set}$   
 $\langle \text{proof} \rangle$

**lemma**  $i\text{-set0-mult-subset-i-set0}: i\text{-set0-mult } k \subseteq i\text{-set0}$   
 $\langle \text{proof} \rangle$

**lemma**  $i\text{-set-mult-0-eq-i-set-cont}: i\text{-set-mult } 0 = i\text{-set-cont}$   
 $\langle \text{proof} \rangle$

**lemma**  $i\text{-set0-mult-0-eq-i-set0-cont}: i\text{-set0-mult } 0 = i\text{-set0-cont}$   
 $\langle \text{proof} \rangle$

**lemma**  $i\text{-set-mult-1-eq-i-set}: i\text{-set-mult } (\text{Suc } 0) = i\text{-set}$   
 $\langle \text{proof} \rangle$

**lemma**  $i\text{-set0-mult-1-eq-i-set0}: i\text{-set0-mult } (\text{Suc } 0) = i\text{-set0}$   
 $\langle \text{proof} \rangle$

**lemma** *i-set-mult-imp-not-empty*:  $I \in i\text{-set-mult } k \implies I \neq \{\}$   
 ⟨proof⟩

**lemma** *iMOD-in-i-set-mult-imp-divisor-mod-0*:  
 $\llbracket m \neq \text{Suc } 0; [r, \text{mod } m] \in i\text{-set-mult } k \rrbracket \implies m \text{ mod } k = 0$   
 ⟨proof⟩

**lemma**  
*divisor-mod-0-imp-iMOD-in-i-set-mult*:  $m \text{ mod } k = 0 \implies [r, \text{mod } m] \in i\text{-set-mult } k$  **and**  
*divisor-mod-0-imp-iMODb-in-i-set-mult*:  $m \text{ mod } k = 0 \implies [r, \text{mod } m, c] \in i\text{-set-mult } k$   
 ⟨proof⟩

**lemma** *iMOD-in-i-set-mult--divisor-mod-0-conv*:  
 $m \neq \text{Suc } 0 \implies ([r, \text{mod } m] \in i\text{-set-mult } k) = (m \text{ mod } k = 0)$   
 ⟨proof⟩

**lemma** *i-set-mult-neq1-subset-i-set*:  $k \neq \text{Suc } 0 \implies i\text{-set-mult } k \subset i\text{-set}$   
 ⟨proof⟩

**lemma** *mod-0-imp-i-set-mult-subset*:  
 $a \text{ mod } b = 0 \implies i\text{-set-mult } a \subseteq i\text{-set-mult } b$   
 ⟨proof⟩

**lemma** *i-set-mult-subset-imp-mod-0*:  
 $\llbracket a \neq \text{Suc } 0; (i\text{-set-mult } a \subseteq i\text{-set-mult } b) \rrbracket \implies a \text{ mod } b = 0$   
 ⟨proof⟩

**lemma** *i-set-mult-subset-conv*:  
 $a \neq \text{Suc } 0 \implies (i\text{-set-mult } a \subseteq i\text{-set-mult } b) = (a \text{ mod } b = 0)$   
 ⟨proof⟩

**lemma** *i-set-mult-mod-0-div*:  
 $\llbracket I \in i\text{-set-mult } k; k \text{ mod } d = 0 \rrbracket \implies I \odot d \in i\text{-set-mult } (k \text{ div } d)$   
 ⟨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*:  
 $\llbracket I \in i\text{-set-mult } k; k \text{ mod } d = 0 \rrbracket \implies I \odot d \in i\text{-set}$   
 ⟨proof⟩

**corollary** *i-set-mult-div-self-i-set*:  
 $I \in i\text{-set-mult } k \implies I \odot k \in i\text{-set}$   
 ⟨proof⟩

**lemma** *i-set-mod-0-mult-in-i-set-mult*:  
 $\llbracket I \in i\text{-set}; m \text{ mod } k = 0 \rrbracket \implies I \otimes m \in i\text{-set-mult } k$

*<proof>*

**lemma** *i-set-self-mult-in-i-set-mult*:

$I \in i\text{-set} \implies I \otimes k \in i\text{-set-mult } k$

*<proof>*

**lemma** *i-set-mult-mod-gr0-div-not-in-i-set*:

$\llbracket 0 < k; 0 < d; 0 < k \bmod d \rrbracket \implies \exists I \in i\text{-set-mult } k. I \otimes d \notin i\text{-set}$

*<proof>*

**lemma** *i-set0-mult-mod-0-div*:

$\llbracket I \in i\text{-set0-mult } k; k \bmod d = 0 \rrbracket \implies I \otimes d \in i\text{-set0-mult } (k \text{ div } d)$

*<proof>*

**corollary** *i-set0-mult-mod-0-div-i-set0*:

$\llbracket I \in i\text{-set0-mult } k; k \bmod d = 0 \rrbracket \implies I \otimes d \in i\text{-set0}$

*<proof>*

**corollary** *i-set0-mult-div-self-i-set0*:

$I \in i\text{-set0-mult } k \implies I \otimes k \in i\text{-set0}$

*<proof>*

**lemma** *i-set0-mod-0-mult-in-i-set0-mult*:

$\llbracket I \in i\text{-set0}; m \bmod k = 0 \rrbracket \implies I \otimes m \in i\text{-set0-mult } k$

*<proof>*

**lemma** *i-set0-self-mult-in-i-set0-mult*:

$I \in i\text{-set0} \implies I \otimes k \in i\text{-set0-mult } k$

*<proof>*

**lemma** *i-set0-mult-mod-gr0-div-not-in-i-set0*:

$\llbracket 0 < k; 0 < d; 0 < k \bmod d \rrbracket \implies \exists I \in i\text{-set0-mult } k. I \otimes d \notin i\text{-set0}$

*<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*]:  $0 \equiv \text{False}$

**definition** *One-bool-def* [*simp*]:  $1 \equiv \text{True}$

**definition** *add-bool-def*:  $a + b \equiv a \vee b$



**definition** *mult-bool-def*:  $a * b \equiv a \wedge b$

**instance**  $\langle proof \rangle$

**end**

**value**  $False < True$

**value**  $True < True$

**value**  $True \leq True$

**lemmas** *bool-op-rel-defs* =

*add-bool-def*

*mult-bool-def*

*less-bool-def*

*le-bool-def*

**instance** *bool* :: *wellorder*

$\langle proof \rangle$

**instance** *bool* :: *comm-semiring-1*

$\langle proof \rangle$

### 3.1 Basic definitions

**lemma** *UNIV-nat*:  $\mathbf{N} = (UNIV::nat\ set)$

$\langle proof \rangle$

Universal temporal operator: Always/Globally

**definition** *iAll* ::  $iT \Rightarrow (Time \Rightarrow bool) \Rightarrow bool$  — Always

**where**  $iAll\ I\ P \equiv \forall t \in I. P\ t$

Existential temporal operator: Eventually/Finally

**definition** *iEx* ::  $iT \Rightarrow (Time \Rightarrow bool) \Rightarrow bool$  — Eventually

**where**  $iEx\ I\ P \equiv \exists t \in I. P\ t$

**syntax**

$-iAll :: Time \Rightarrow iT \Rightarrow (Time \Rightarrow bool) \Rightarrow bool\ ((\exists \square - \cdot / -) [0, 0, 10] 10)$

$-iEx :: Time \Rightarrow iT \Rightarrow (Time \Rightarrow bool) \Rightarrow bool\ ((\exists \diamond - \cdot / -) [0, 0, 10] 10)$

**translations**

$\square\ t\ I. P \equiv CONST\ iAll\ I\ (\lambda t. P)$

$\diamond\ t\ I. P \equiv CONST\ iEx\ I\ (\lambda t. P)$

Future temporal operator: Next

**definition** *iNext* ::  $Time \Rightarrow iT \Rightarrow (Time \Rightarrow bool) \Rightarrow bool$  — Next

**where**  $iNext\ t0\ I\ P \equiv P\ (inext\ t0\ I)$

Past temporal operator: Last/Previous

**definition** *iLast* ::  $Time \Rightarrow iT \Rightarrow (Time \Rightarrow bool) \Rightarrow bool$  — Last

**where**  $iLast\ t0\ I\ P \equiv P\ (iprev\ t0\ I)$

**syntax**

$-iNext :: Time \Rightarrow Time \Rightarrow iT \Rightarrow (Time \Rightarrow bool) \Rightarrow bool\ ((\exists \circ - - - ./ -)\ [0, 0, 10]\ 10)$

$-iLast :: Time \Rightarrow Time \Rightarrow iT \Rightarrow (Time \Rightarrow bool) \Rightarrow bool\ ((\exists \ominus - - - ./ -)\ [0, 0, 10]\ 10)$

**translations**

$\circ\ t\ t0\ I.\ P \equiv CONST\ iNext\ t0\ I\ (\lambda t.\ P)$

$\ominus\ t\ t0\ I.\ P \equiv CONST\ iLast\ t0\ I\ (\lambda t.\ P)$

**lemma**  $\circ\ t\ 10\ [0\dots].\ (t + 10 > 10)$

$\langle proof \rangle$

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**  $iNextWeak :: Time \Rightarrow iT \Rightarrow (Time \Rightarrow bool) \Rightarrow bool$  — Weak Next  
**where**  $iNextWeak\ t0\ I\ P \equiv (\Box\ t\ \{inext\ t0\ I\}\ \downarrow >\ t0.\ P\ t)$

**definition**  $iNextStrong :: Time \Rightarrow iT \Rightarrow (Time \Rightarrow bool) \Rightarrow bool$  — Strong Next  
**where**  $iNextStrong\ t0\ I\ P \equiv (\Diamond\ t\ \{inext\ t0\ I\}\ \downarrow >\ t0.\ P\ t)$

**definition**  $iLastWeak :: Time \Rightarrow iT \Rightarrow (Time \Rightarrow bool) \Rightarrow bool$  — Weak Last  
**where**  $iLastWeak\ t0\ I\ P \equiv (\Box\ t\ \{iprev\ t0\ I\}\ \downarrow <\ t0.\ P\ t)$

**definition**  $iLastStrong :: Time \Rightarrow iT \Rightarrow (Time \Rightarrow bool) \Rightarrow bool$  — Strong Last  
**where**  $iLastStrong\ t0\ I\ P \equiv (\Diamond\ t\ \{iprev\ t0\ I\}\ \downarrow <\ t0.\ P\ t)$

**syntax**

$-iNextWeak :: Time \Rightarrow Time \Rightarrow iT \Rightarrow (Time \Rightarrow bool) \Rightarrow bool\ ((\exists \circ_W - - - ./ -)\ [0, 0, 10]\ 10)$

$-iNextStrong :: Time \Rightarrow Time \Rightarrow iT \Rightarrow (Time \Rightarrow bool) \Rightarrow bool\ ((\exists \circ_S - - - ./ -)\ [0, 0, 10]\ 10)$

$-iLastWeak :: Time \Rightarrow Time \Rightarrow iT \Rightarrow (Time \Rightarrow bool) \Rightarrow bool\ ((\exists \ominus_W - - - ./ -)\ [0, 0, 10]\ 10)$

$-iLastStrong :: Time \Rightarrow Time \Rightarrow iT \Rightarrow (Time \Rightarrow bool) \Rightarrow bool\ ((\exists \ominus_S - - - ./ -)\ [0, 0, 10]\ 10)$

**translations**

$\circ_W\ t\ t0\ I.\ P \equiv CONST\ iNextWeak\ t0\ I\ (\lambda t.\ P)$

$\circ_S\ t\ t0\ I.\ P \equiv CONST\ iNextStrong\ t0\ I\ (\lambda t.\ P)$

$\ominus_W\ t\ t0\ I.\ P \equiv CONST\ iLastWeak\ t0\ I\ (\lambda t.\ P)$

$\ominus_S\ t\ t0\ I.\ P \equiv CONST\ iLastStrong\ t0\ I\ (\lambda t.\ P)$

Some examples for Next and Last operator

**lemma**  $\circ\ t\ 5\ [0\dots,10].\ ([0::int,10,20,30,40,50,60,70,80,90] ! t < 80)$

$\langle proof \rangle$

**lemma**  $\ominus t 5 [0\dots,10]. ([0::int,10,20,30,40,50,60,70,80,90] ! t < 80)$   
 ⟨proof⟩

Temporal Until operator

**definition**  $iUntil :: iT \Rightarrow (Time \Rightarrow bool) \Rightarrow (Time \Rightarrow bool) \Rightarrow bool$  — Until  
**where**  $iUntil I P Q \equiv \diamond t I. Q t \wedge (\Box t' (I \downarrow < t). P t')$

Temporal Since operator (past operator corresponding to Until)

**definition**  $iSince :: iT \Rightarrow (Time \Rightarrow bool) \Rightarrow (Time \Rightarrow bool) \Rightarrow bool$  — Since  
**where**  $iSince I P Q \equiv \diamond t I. Q t \wedge (\Box t' (I \downarrow > t). P t')$

**syntax**

$-iUntil :: Time \Rightarrow Time \Rightarrow iT \Rightarrow (Time \Rightarrow bool) \Rightarrow (Time \Rightarrow bool) \Rightarrow bool$   
 $((./ - (3U - -)./ -) [10, 0, 0, 0, 10] 10)$   
 $-iSince :: Time \Rightarrow Time \Rightarrow iT \Rightarrow (Time \Rightarrow bool) \Rightarrow (Time \Rightarrow bool) \Rightarrow bool$   
 $((./ - (3S - -)./ -) [10, 0, 0, 0, 10] 10)$

**translations**

$P. t U t' I. Q \equiv CONST iUntil I (\lambda t. P) (\lambda t'. Q)$   
 $P. t S t' I. Q \equiv CONST iSince I (\lambda t. P) (\lambda t'. Q)$

**definition**  $iWeakUntil :: iT \Rightarrow (Time \Rightarrow bool) \Rightarrow (Time \Rightarrow bool) \Rightarrow bool$  —  
 Weak Until/Wating-for/Unless  
**where**  $iWeakUntil I P Q \equiv (\Box t I. P t) \vee (\diamond t I. Q t \wedge (\Box t' (I \downarrow < t). P t'))$

**definition**  $iWeakSince :: iT \Rightarrow (Time \Rightarrow bool) \Rightarrow (Time \Rightarrow bool) \Rightarrow bool$  —  
 Weak Since/Back-to  
**where**  $iWeakSince I P Q \equiv (\Box t I. P t) \vee (\diamond t I. Q t \wedge (\Box t' (I \downarrow > t). P t'))$

**syntax**

$-iWeakUntil :: Time \Rightarrow Time \Rightarrow iT \Rightarrow (Time \Rightarrow bool) \Rightarrow (Time \Rightarrow bool) \Rightarrow bool$   
 $((./ - (3W - -)./ -) [10, 0, 0, 0, 10] 10)$   
 $-iWeakSince :: Time \Rightarrow Time \Rightarrow iT \Rightarrow (Time \Rightarrow bool) \Rightarrow (Time \Rightarrow bool) \Rightarrow bool$   
 $((./ - (3B - -)./ -) [10, 0, 0, 0, 10] 10)$

**translations**

$P. t W t' I. Q \equiv CONST iWeakUntil I (\lambda t. P) (\lambda t'. Q)$   
 $P. t B t' I. Q \equiv CONST iWeakSince I (\lambda t. P) (\lambda t'. Q)$

**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  
 ((./ - (3R - -)./ -) [10, 0, 0, 0, 10] 10)  
 -iTrigger :: Time  $\Rightarrow$  Time  $\Rightarrow$  iT  $\Rightarrow$  (Time  $\Rightarrow$  bool)  $\Rightarrow$  (Time  $\Rightarrow$  bool)  $\Rightarrow$  bool  
 ((./ - (3T - -)./ -) [10, 0, 0, 0, 10] 10)

**translations**

P. t R t' I. Q  $\equiv$  CONST iRelease I ( $\lambda t. P$ ) ( $\lambda t'. Q$ )  
 P. t T t' I. Q  $\equiv$  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

$\langle ML \rangle$

## 3.2 Basic lemmata for temporal operators

### 3.2.1 Intro/elim rules

**lemma**

iexI[*intro*]:  $\llbracket P t; t \in I \rrbracket \Longrightarrow \diamond t I. P t$  **and**  
 rev-iexI[*intro?*]:  $\llbracket t \in I; P t \rrbracket \Longrightarrow \diamond t I. P t$  **and**  
 iexE[*elim!*]:  $\llbracket \diamond t I. P t; \bigwedge t. \llbracket t \in I; P t \rrbracket \Longrightarrow Q \rrbracket \Longrightarrow Q$   
 $\langle proof \rangle$

**lemma**

iallI[*intro!*]:  $(\bigwedge t. t \in I \Longrightarrow P t) \Longrightarrow \square t I. P t$  **and**  
 ispec[*dest?*]:  $\llbracket \square t I. P t; t \in I \rrbracket \Longrightarrow P t$  **and**  
 iallE[*elim*]:  $\llbracket \square t I. P t; P t \Longrightarrow Q; t \notin I \Longrightarrow Q \rrbracket \Longrightarrow Q$   
 $\langle proof \rangle$

**lemma**

iuntilI[*intro*]:  
 $\llbracket Q t; (\bigwedge t'. t' \in I \downarrow < t \Longrightarrow P t'); t \in I \rrbracket \Longrightarrow P t'. t' \mathcal{U} t I. Q t$  **and**  
 rev-iuntilI[*intro?*]:  
 $\llbracket t \in I; Q t; (\bigwedge t'. t' \in I \downarrow < t \Longrightarrow P t') \rrbracket \Longrightarrow P t'. t' \mathcal{U} t I. Q t$   
 $\langle proof \rangle$

**lemma**

iuntilE[*elim*]:  
 $\llbracket P' t'. t' \mathcal{U} t I. P t; \bigwedge t. \llbracket t \in I; P t \rrbracket \Longrightarrow Q \rrbracket \Longrightarrow Q$   
 $\langle proof \rangle$

**lemma***isinceI[intro]:* $\llbracket Q\ t; (\bigwedge t'. t' \in I \downarrow > t \implies P\ t'); t \in I \rrbracket \implies P\ t'. t' \mathcal{S}\ t\ I.\ Q\ t$  **and***rev-isinceI[intro?]:* $\llbracket t \in I; Q\ t; (\bigwedge t'. t' \in I \downarrow > t \implies P\ t') \rrbracket \implies P\ t'. t' \mathcal{S}\ t\ I.\ Q\ t$ *<proof>***lemma***isinceE[elim]:* $\llbracket P'\ t'. t' \mathcal{S}\ t\ I.\ P\ t; \bigwedge t. \llbracket t \in I; P\ t \rrbracket \implies Q \rrbracket \implies Q$ *<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** *isex-triv[simp]:*  $(\Diamond\ t\ I.\ P) = ((\exists t. t \in I) \wedge P)$ *<proof>***lemma** *isex-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** *isex-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** *isex-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** *isex-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** *isex-commute:* $(\Diamond\ t1\ I1.\ \Diamond\ t2\ I2.\ P\ t1\ t2) =$  $(\Diamond\ t2\ I2.\ \Diamond\ t1\ I1.\ P\ t1\ t2)$ *<proof>***lemma** *iall-conjL1:* $I2 \neq \{\} \implies$  $(\Box\ t1\ I1.\ (P1\ t1 \wedge (\Box\ t2\ I2.\ P2\ t1\ t2))) =$  $(\Box\ t1\ I1.\ \Box\ t2\ I2.\ P1\ t1 \wedge P2\ t1\ t2)$

$\langle proof \rangle$

**lemma** *iall-conjR1*:

$$\begin{aligned} I2 \neq \{\} &\implies \\ (\Box t1 I1. ((\Box t2 I2. P2 t1 t2) \wedge P1 t1)) &= \\ (\Box t1 I1. \Box t2 I2. P2 t1 t2 \wedge P1 t1) & \\ \langle proof \rangle & \end{aligned}$$

**lemma** *iall-conjL2*:

$$\begin{aligned} \Box t1 I1. I2 t1 \neq \{\} &\implies \\ (\Box t1 I1. (P1 t1 \wedge (\Box t2 (I2 t1). P2 t1 t2))) &= \\ (\Box t1 I1. \Box t2 (I2 t1). P1 t1 \wedge P2 t1 t2) & \\ \langle proof \rangle & \end{aligned}$$

**lemma** *iall-conjR2*:

$$\begin{aligned} \Box t1 I1. I2 t1 \neq \{\} &\implies \\ (\Box t1 I1. ((\Box t2 (I2 t1). P2 t1 t2) \wedge P1 t1)) &= \\ (\Box t1 I1. \Box t2 (I2 t1). P2 t1 t2 \wedge P1 t1) & \\ \langle proof \rangle & \end{aligned}$$

**lemma** *iall-commute*:

$$\begin{aligned} (\Box t1 I1. \Box t2 I2. P t1 t2) &= \\ (\Box t2 I2. \Box t1 I1. P t1 t2) & \\ \langle proof \rangle & \end{aligned}$$

**lemma** *iall-conj-distrib*:

$$\begin{aligned} (\Box t I. P t \wedge Q t) &= ((\Box t I. P t) \wedge (\Box t I. Q t)) \\ \langle proof \rangle & \end{aligned}$$

**lemma** *iox-disj-distrib*:

$$\begin{aligned} (\Diamond t I. P t \vee Q t) &= ((\Diamond t I. P t) \vee (\Diamond t I. Q t)) \\ \langle proof \rangle & \end{aligned}$$

**lemma** *iall-conj-distrib2*:

$$\begin{aligned} (\Box t1 I1. \Box t2 (I2 t1). P t1 t2 \wedge Q t1 t2) &= \\ ((\Box t1 I1. \Box t2 (I2 t1). P t1 t2) \wedge (\Box t1 I1. \Box t2 (I2 t1). Q t1 t2)) & \\ \langle proof \rangle & \end{aligned}$$

**lemma** *iox-disj-distrib2*:

$$\begin{aligned} (\Diamond t1 I1. \Diamond t2 (I2 t1). P t1 t2 \vee Q t1 t2) &= \\ ((\Diamond t1 I1. \Diamond t2 (I2 t1). P t1 t2) \vee (\Diamond t1 I1. \Diamond t2 (I2 t1). Q t1 t2)) & \\ \langle proof \rangle & \end{aligned}$$

**lemma** *iUntil-disj-distrib*:

$$\begin{aligned} (P t1. t1 \mathcal{U} t2 I. (Q1 t2 \vee Q2 t2)) &= ((P t1. t1 \mathcal{U} t2 I. Q1 t2) \vee (P t1. t1 \mathcal{U} \\ t2 I. Q2 t2)) & \\ \langle proof \rangle & \end{aligned}$$

**lemma** *iSince-disj-distrib*:

$$(P \ t1. \ t1 \ S \ t2 \ I. \ (Q1 \ t2 \ \vee \ Q2 \ t2)) = ((P \ t1. \ t1 \ S \ t2 \ I. \ Q1 \ t2) \ \vee \ (P \ t1. \ t1 \ S \ t2 \ I. \ Q2 \ t2))$$

*<proof>*

**lemma**

$$iNext\text{-iff}: (\bigcirc \ t \ t0 \ I. \ P \ t) = (\Box \ t \ [\dots 0] \oplus (inext \ t0 \ I). \ P \ t) \ \mathbf{and}$$

$$iLast\text{-iff}: (\ominus \ t \ t0 \ I. \ P \ t) = (\Box \ t \ [\dots 0] \oplus (iprev \ t0 \ I). \ P \ t)$$

*<proof>*

**lemma**

$$iNext\text{-iEx-iff}: (\bigcirc \ t \ t0 \ I. \ P \ t) = (\Diamond \ t \ [\dots 0] \oplus (inext \ t0 \ I). \ P \ t) \ \mathbf{and}$$

$$iLast\text{-iEx-iff}: (\ominus \ t \ t0 \ I. \ P \ t) = (\Diamond \ t \ [\dots 0] \oplus (iprev \ t0 \ I). \ P \ t)$$

*<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* =

$$iNext\text{-iff} \ iLast\text{-iff}$$

*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*:  $(I = \{\}) = (\Box t I. False)$  *<proof>*

**lemma** *not-empty-iff-iEx-True*:  $(I \neq \{\}) = (\Diamond t I. True)$  *<proof>*

**lemma**

*iNext-True*:  $\bigcirc t t0 I. True$  **and**  
*iNextWeak-True*:  $(\bigcirc_W t t0 I. True)$  **and**  
*iNext-False*:  $\neg (\bigcirc t t0 I. False)$  **and**  
*iNextStrong-False*:  $\neg (\bigcirc_S t t0 I. False)$

*<proof>*

**lemma**

*iNextStrong-True*:  $(\bigcirc_S t t0 I. True) = (I \downarrow > t0 \neq \{\} \wedge t0 \in I)$  **and**  
*iNextWeak-False*:  $(\neg (\bigcirc_W t t0 I. False)) = (I \downarrow > t0 \neq \{\} \wedge t0 \in I)$

*<proof>*

**lemma**

*iLast-True*:  $\ominus t t0 I. True$  **and**



*iLastWeak-True*:  $(\ominus_W t t0 I. True)$  **and**  
*iLast-False*:  $\neg (\ominus t t0 I. False)$  **and**  
*iLastStrong-False*:  $\neg (\ominus_S t t0 I. False)$   
 ⟨proof⟩

**lemma**

*iLastStrong-True*:  $(\ominus_S t t0 I. True) = (I \downarrow < t0 \neq \{\} \wedge t0 \in I)$  **and**  
*iLastWeak-False*:  $(\neg (\ominus_W t t0 I. False)) = (I \downarrow < t0 \neq \{\} \wedge t0 \in I)$   
 ⟨proof⟩

**lemma** *iUntil-True-left[simp]*:  $(True. t' U t I. Q t) = (\diamond t I. Q t)$   
 ⟨proof⟩

**lemma** *iUntil-True[simp]*:  $(P t'. t' U t I. True) = (I \neq \{\})$   
 ⟨proof⟩

**lemma** *iUntil-False-left[simp]*:  $(False. t' U t I. Q t) = (I \neq \{\} \wedge Q (iMin I))$   
 ⟨proof⟩

**lemma** *iUntil-False[simp]*:  $\neg (P t'. t' U t I. False)$   
 ⟨proof⟩

**lemma** *iSince-True-left[simp]*:  $(True. t' S t I. Q t) = (\diamond t I. Q t)$   
 ⟨proof⟩

**lemma** *iSince-True-if*:

$(P t'. t' S t I. True) =$   
 (if finite  $I$  then  $I \neq \{\}$  else  $\diamond t1 I. \square t2 (I \downarrow > t1). P t2)$   
 ⟨proof⟩

**corollary** *iSince-True-finite[simp]*: finite  $I \implies (P t'. t' S t I. True) = (I \neq \{\})$   
 ⟨proof⟩

**lemma** *iSince-False-left[simp]*:  $(False. t' S t I. Q t) = (finite I \wedge I \neq \{\} \wedge Q (Max I))$   
 ⟨proof⟩

**lemma** *iSince-False[simp]*:  $\neg (P t'. t' S t I. False)$   
 ⟨proof⟩

**lemma** *iWeakUntil-True-left[simp]*:  $True. t' W t I. Q t$   
 ⟨proof⟩

**lemma** *iWeakUntil-True[simp]*:  $P t'. t' W t I. True$   
 ⟨proof⟩

**lemma** *iWeakUntil-False-left[simp]*:  $(False. t' W t I. Q t) = (I = \{\} \vee Q (iMin I))$   
 ⟨proof⟩

**lemma** *iWeakUntil-False[simp]*:  $(P\ t'.\ t'\ \mathcal{W}\ t\ I.\ \text{False}) = (\Box\ t\ I.\ P\ t)$   
 $\langle\text{proof}\rangle$

**lemma** *iWeakSince-True-left[simp]*:  $\text{True}.\ t'\ \mathcal{B}\ t\ I.\ Q\ t$   
 $\langle\text{proof}\rangle$

**lemma** *iWeakSince-True-disj*:  
 $(P\ t'.\ t'\ \mathcal{B}\ t\ I.\ \text{True}) =$   
 $(I = \{\}) \vee (\Diamond\ t1\ I.\ \Box\ t2\ (I\ \Downarrow\ t1).\ P\ t2))$   
 $\langle\text{proof}\rangle$

**lemma** *iWeakSince-True-finite[simp]*:  $\text{finite}\ I \implies (P\ t'.\ t'\ \mathcal{B}\ t\ I.\ \text{True})$   
 $\langle\text{proof}\rangle$

**lemma** *iWeakSince-False-left[simp]*:  $(\text{False}.\ t'\ \mathcal{B}\ t\ I.\ Q\ t) = (I = \{\}) \vee (\text{finite}\ I \wedge$   
 $Q\ (\text{Max}\ I))$   
 $\langle\text{proof}\rangle$

**lemma** *iWeakSince-False[simp]*:  $(P\ t'.\ t'\ \mathcal{B}\ t\ I.\ \text{False}) = (\Box\ t\ I.\ P\ t)$   
 $\langle\text{proof}\rangle$

**lemma** *iRelease-True-left[simp]*:  $(\text{True}.\ t'\ \mathcal{R}\ t\ I.\ Q\ t) = (I = \{\}) \vee Q\ (\text{iMin}\ I)$   
 $\langle\text{proof}\rangle$

**lemma** *iRelease-True[simp]*:  $P\ t'.\ t'\ \mathcal{R}\ t\ I.\ \text{True}$   
 $\langle\text{proof}\rangle$

**lemma** *iRelease-False-left[simp]*:  $(\text{False}.\ t'\ \mathcal{R}\ t\ I.\ Q\ t) = (\Box\ t\ I.\ Q\ t)$   
 $\langle\text{proof}\rangle$

**lemma** *iRelease-False[simp]*:  $(P\ t'.\ t'\ \mathcal{R}\ t\ I.\ \text{False}) = (I = \{\})$   
 $\langle\text{proof}\rangle$

**lemma** *iTrigger-True-left[simp]*:  $(\text{True}.\ t'\ \mathcal{T}\ t\ I.\ Q\ t) = (I = \{\}) \vee (\Diamond\ t1\ I.\ \Box\ t2$   
 $(I\ \Downarrow\ t1).\ Q\ t2))$   
 $\langle\text{proof}\rangle$

**lemma** *iTrigger-True[simp]*:  $P\ t'.\ t'\ \mathcal{T}\ t\ I.\ \text{True}$   
 $\langle\text{proof}\rangle$

**lemma** *iTrigger-False-left[simp]*:  $(\text{False}.\ t'\ \mathcal{T}\ t\ I.\ Q\ t) = (\Box\ t\ I.\ Q\ t)$   
 $\langle\text{proof}\rangle$

**lemma** *iTrigger-False[simp]*:  $(P\ t'.\ t'\ \mathcal{T}\ t\ I.\ \text{False}) = (I = \{\})$   
 $\langle\text{proof}\rangle$

**lemma**  
*iUntil-TrueTrue[simp]*:  $(\text{True}.\ t'\ \mathcal{U}\ t\ I.\ \text{True}) = (I \neq \{\})$  **and**

*iSince-TrueTrue[simp]*:  $(\text{True}. t' \mathcal{S} t I. \text{True}) = (I \neq \{\})$  **and**  
*iWeakUntil-TrueTrue[simp]*:  $\text{True}. t' \mathcal{W} t I. \text{True}$  **and**  
*iWeakSince-TrueTrue[simp]*:  $\text{True}. t' \mathcal{B} t I. \text{True}$  **and**  
*iRelease-TrueTrue[simp]*:  $\text{True}. t' \mathcal{R} t I. \text{True}$  **and**  
*iTrigger-TrueTrue[simp]*:  $\text{True}. t' \mathcal{T} t I. \text{True}$   
 ⟨proof⟩

### 3.2.3 Empty sets and singletons

**lemma** *iAll-empty[simp]*:  $\square t \{\}$ .  $P t$  ⟨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]*:  $(\square t' \{t\}. P t') = P t$  ⟨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]*:  $(P t0. t0 \mathcal{W} t1 \{t\}. Q t1) = (P t \vee Q t)$   
 ⟨proof⟩

**lemma** *iWeakSince-singleton[simp]*:  $(P t0. t0 \mathcal{B} t1 \{t\}. Q t1) = (P t \vee Q t)$   
 ⟨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*:  $(\text{True}. t' \mathcal{U} t I. P t) = (\Diamond t I. P t)$  *<proof>*

**lemma** *iSince-iEx-conv*:  $(\text{True}. t' \mathcal{S} t I. P t) = (\Diamond t I. P t)$  *<proof>*

**lemma** *iRelease-iAll-conv*:  $(\text{False}. t' \mathcal{R} t I. P t) = (\Box t I. P t)$   
*<proof>*

**lemma** *iTrigger-iAll-conv*:  $(\text{False}. t' \mathcal{T} t I. P t) = (\Box t I. P t)$   
*<proof>*

**lemma** *iWeakUntil-iUntil-conv*:  $(P t'. t' \mathcal{W} t I. Q t) = ((P t'. t' \mathcal{U} t I. Q t) \vee (\Box t I. P t))$   
*<proof>*

**lemma** *iWeakSince-iSince-conv*:  $(P t'. t' \mathcal{B} t I. Q t) = ((P t'. t' \mathcal{S} t I. Q t) \vee (\Box t I. P t))$   
*<proof>*

**lemma** *iUntil-iWeakUntil-conv*:  $(P t'. t' \mathcal{U} t I. Q t) = ((P t'. t' \mathcal{W} t I. Q t) \wedge (\Diamond t I. Q t))$   
*<proof>*

**lemma** *iSince-iWeakSince-conv*:  $(P t'. t' \mathcal{S} t I. Q t) = ((P t'. t' \mathcal{B} t I. Q t) \wedge (\Diamond t I. Q t))$   
*<proof>*

**lemma** *iRelease-iWeakUntil-conv*:  $(P t'. t' \mathcal{R} t I. Q t) = (Q t'. t' \mathcal{W} t I. (Q t \wedge P t))$   
*<proof>*

**lemma** *iRelease-iUntil-conv*:  $(P t'. t' \mathcal{R} t I. Q t) = ((\Box t I. Q t) \vee (Q t'. t' \mathcal{U} t I. (Q t \wedge P t)))$   
*<proof>*

**lemma** *iTrigger-iWeakSince-conv*:  $(P t'. t' \mathcal{T} t I. Q t) = (Q t'. t' \mathcal{B} t I. (Q t \wedge P t))$   
*<proof>*

**lemma** *iTrigger-iSince-conv*:  $(P\ t'.\ t'\ \mathcal{T}\ t\ I.\ Q\ t) = ((\Box\ t\ I.\ Q\ t) \vee (Q\ t'.\ t'\ \mathcal{S}\ t\ I.\ (Q\ t \wedge P\ t)))$   
 ⟨proof⟩

**lemma** *iRelease-not-iUntil-conv*:  $(P\ t'.\ t'\ \mathcal{R}\ t\ I.\ Q\ t) = (\neg(\neg P\ t'.\ t'\ \mathcal{U}\ t\ I.\ \neg Q\ t))$   
 ⟨proof⟩

**lemma** *iUntil-not-iRelease-conv*:  $(P\ t'.\ t'\ \mathcal{U}\ t\ I.\ Q\ t) = (\neg(\neg P\ t'.\ t'\ \mathcal{R}\ t\ I.\ \neg Q\ t))$   
 ⟨proof⟩

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*:  $\text{finite } I \implies (P\ t'.\ t'\ \mathcal{T}\ t\ I.\ Q\ t) = (\neg(\neg P\ t'.\ t'\ \mathcal{S}\ t\ I.\ \neg Q\ t))$   
 ⟨proof⟩

**lemma** *iSince-not-iTrigger-conv*:  $\text{finite } I \implies (P\ t'.\ t'\ \mathcal{S}\ t\ I.\ Q\ t) = (\neg(\neg P\ t'.\ t'\ \mathcal{T}\ t\ I.\ \neg Q\ t))$   
 ⟨proof⟩

**lemma** *not-iUntil*:  
 $(\neg(P\ t1.\ t1\ \mathcal{U}\ t2\ I.\ Q\ t2)) =$   
 $(\Box\ t\ I.\ (Q\ t \longrightarrow (\Diamond\ t'\ (I\ \downarrow < t).\ \neg P\ t')))$   
 ⟨proof⟩

**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*:

$$(P \ t1. \ t1 \ \mathcal{W} \ t2 \ I. \ Q \ t2) = \\ (\neg (\neg Q \ t1. \ t1 \ \mathcal{U} \ t2 \ I. (\neg P \ t2 \wedge \neg Q \ t2))) \\ \langle \text{proof} \rangle$$

Negation and temporal operators

**lemma**

$$\begin{aligned} \text{not-iNext[simp]}: (\neg (\bigcirc \ t \ t0 \ I. \ P \ t)) &= (\bigcirc \ t \ t0 \ I. \ \neg \ P \ t) \ \mathbf{and} \\ \text{not-iNextWeak[simp]}: (\neg (\bigcirc_W \ t \ t0 \ I. \ P \ t)) &= (\bigcirc_S \ t \ t0 \ I. \ \neg \ P \ t) \ \mathbf{and} \\ \text{not-iNextStrong[simp]}: (\neg (\bigcirc_S \ t \ t0 \ I. \ P \ t)) &= (\bigcirc_W \ t \ t0 \ I. \ \neg \ P \ t) \ \mathbf{and} \\ \text{not-iLast[simp]}: (\neg (\ominus \ t \ t0 \ I. \ P \ t)) &= (\ominus \ t \ t0 \ I. \ \neg \ P \ t) \ \mathbf{and} \\ \text{not-iLastWeak[simp]}: (\neg (\ominus_W \ t \ t0 \ I. \ P \ t)) &= (\ominus_S \ t \ t0 \ I. \ \neg \ P \ t) \ \mathbf{and} \\ \text{not-iLastStrong[simp]}: (\neg (\ominus_S \ t \ t0 \ I. \ P \ t)) &= (\ominus_W \ t \ t0 \ I. \ \neg \ P \ t) \\ \langle \text{proof} \rangle \end{aligned}$$

**lemma** *not-iWeakUntil*:

$$\begin{aligned} (\neg (P \ t1. \ t1 \ \mathcal{W} \ t2 \ I. \ Q \ t2)) &= \\ ((\Box \ t \ I. (Q \ t \longrightarrow (\Diamond \ t' (I \ \downarrow < \ t). \ \neg \ P \ t'))) \wedge (\Diamond \ t \ I. \ \neg \ P \ t)) \\ \langle \text{proof} \rangle \end{aligned}$$

**lemma** *not-iWeakSince*:

$$\begin{aligned} (\neg (P \ t1. \ t1 \ \mathcal{B} \ t2 \ I. \ Q \ t2)) &= \\ ((\Box \ t \ I. (Q \ t \longrightarrow (\Diamond \ t' (I \ \downarrow > \ t). \ \neg \ P \ t'))) \wedge (\Diamond \ t \ I. \ \neg \ P \ t)) \\ \langle \text{proof} \rangle \end{aligned}$$

**lemma** *not-iRelease*:

$$\begin{aligned} (\neg (P \ t'. \ t' \ \mathcal{R} \ t \ I. \ Q \ t)) &= \\ ((\Diamond \ t \ I. \ \neg \ Q \ t) \wedge (\Box \ t \ I. \ P \ t \longrightarrow (\Diamond \ t \ I \ \downarrow \leq \ t. \ \neg \ Q \ t))) \\ \langle \text{proof} \rangle \end{aligned}$$

**lemma** *not-iRelease-iUntil-conv*:

$$(\neg (P \ t'. \ t' \ \mathcal{R} \ t \ I. \ Q \ t)) = (\neg P \ t'. \ t' \ \mathcal{U} \ t \ I. \ \neg \ Q \ t) \\ \langle \text{proof} \rangle$$

**lemma** *not-iTrigger*:

$$\begin{aligned} (\neg (P \ t'. \ t' \ \mathcal{T} \ t \ I. \ Q \ t)) &= \\ ((\Diamond \ t \ I. \ \neg \ Q \ t) \wedge (\Box \ t \ I. \ \neg \ P \ t \vee (\Diamond \ t \ I \ \downarrow \geq \ t. \ \neg \ Q \ t))) \\ \langle \text{proof} \rangle \end{aligned}$$

**lemma** *not-iTrigger-iSince-conv*:

$$\text{finite } I \implies (\neg (P \ t'. \ t' \ \mathcal{T} \ t \ I. \ Q \ t)) = (\neg P \ t'. \ t' \ \mathcal{S} \ t \ I. \ \neg \ Q \ t) \\ \langle \text{proof} \rangle$$

### 3.2.5 Some implication results

**lemma** *all-imp-iall*:  $\forall x. P \ x \implies \Box \ t \ I. \ P \ t$   $\langle \text{proof} \rangle$

**lemma** *bex-imp-lex*:  $\Diamond \ t \ I. \ P \ t \implies \exists x. P \ x$   $\langle \text{proof} \rangle$

**lemma** *iAll-imp-iEx*:  $I \neq \{\}$   $\implies \Box \ t \ I. \ P \ t \implies \Diamond \ t \ I. \ P \ t$   $\langle \text{proof} \rangle$

**lemma** *i-set-iAll-imp-iEx*:  $I \in \text{i-set} \implies \Box \ t \ I. \ P \ t \implies \Diamond \ t \ I. \ P \ t$

*<proof>*

**lemmas**  $iT\text{-}iAll\text{-}imp\text{-}iEx = iT\text{-}not\text{-}empty[THEN\ iAll\text{-}imp\text{-}iEx]$

**lemma**  $iUntil\text{-}imp\text{-}iEx: P\ t1.\ t1\ \mathcal{U}\ t2\ I.\ Q\ t2 \implies \diamond\ t\ I.\ Q\ t$   
*<proof>*

**lemma**  $iSince\text{-}imp\text{-}iEx: P\ t1.\ t1\ \mathcal{S}\ t2\ I.\ Q\ t2 \implies \diamond\ t\ I.\ Q\ t$   
*<proof>*

**lemma**  $iAll\text{-}subset\text{-}imp\text{-}iAll: \llbracket \square\ t\ B.\ P\ t; A \subseteq B \rrbracket \implies \square\ t\ A.\ P\ t$   
*<proof>*

**lemma**  $iEx\text{-}subset\text{-}imp\text{-}iEx: \llbracket \diamond\ t\ A.\ P\ t; A \subseteq B \rrbracket \implies \diamond\ t\ B.\ P\ t$   
*<proof>*

**lemma**  $iAll\text{-}mp: \llbracket \square\ t\ I.\ P\ t \longrightarrow Q\ t; \square\ t\ I.\ P\ t \rrbracket \implies \square\ t\ I.\ Q\ t$  *<proof>*

**lemma**  $iEx\text{-}mp: \llbracket \square\ t\ I.\ P\ t \longrightarrow Q\ t; \diamond\ t\ I.\ P\ t \rrbracket \implies \diamond\ t\ I.\ Q\ t$  *<proof>*

**lemma**  $iUntil\text{-}imp:$

$\llbracket P1\ t1.\ t1\ \mathcal{U}\ t2\ I.\ Q\ t2; \square\ t\ I.\ P1\ t \longrightarrow P2\ t \rrbracket \implies P2\ t1.\ t1\ \mathcal{U}\ t2\ I.\ Q\ t2$   
*<proof>*

**lemma**  $iSince\text{-}imp:$

$\llbracket P1\ t1.\ t1\ \mathcal{S}\ t2\ I.\ Q\ t2; \square\ t\ I.\ P1\ t \longrightarrow P2\ t \rrbracket \implies P2\ t1.\ t1\ \mathcal{S}\ t2\ I.\ Q\ t2$   
*<proof>*

**lemma**  $iWeakUntil\text{-}imp:$

$\llbracket P1\ t1.\ t1\ \mathcal{W}\ t2\ I.\ Q\ t2; \square\ t\ I.\ P1\ t \longrightarrow P2\ t \rrbracket \implies P2\ t1.\ t1\ \mathcal{W}\ t2\ I.\ Q\ t2$   
*<proof>*

**lemma**  $iWeakSince\text{-}imp:$

$\llbracket P1\ t1.\ t1\ \mathcal{B}\ t2\ I.\ Q\ t2; \square\ t\ I.\ P1\ t \longrightarrow P2\ t \rrbracket \implies P2\ t1.\ t1\ \mathcal{B}\ t2\ I.\ Q\ t2$   
*<proof>*

**lemma**  $iRelease\text{-}imp:$

$\llbracket P1\ t1.\ t1\ \mathcal{R}\ t2\ I.\ Q\ t2; \square\ t\ I.\ P1\ t \longrightarrow P2\ t \rrbracket \implies P2\ t1.\ t1\ \mathcal{R}\ t2\ I.\ Q\ t2$   
*<proof>*

**lemma**  $iTrigger\text{-}imp:$

$\llbracket P1\ t1.\ t1\ \mathcal{T}\ t2\ I.\ Q\ t2; \square\ t\ I.\ P1\ t \longrightarrow P2\ t \rrbracket \implies P2\ t1.\ t1\ \mathcal{T}\ t2\ I.\ Q\ t2$   
*<proof>*

**lemma**  $iMin\text{-}imp\text{-}iUntil:$

$\llbracket I \neq \{\}; Q\ (iMin\ I) \rrbracket \implies P\ t'.\ t'\ \mathcal{U}\ t\ I.\ Q\ t$   
*<proof>*

**lemma** *Max-imp-iSince*:

$$\llbracket \text{finite } I; I \neq \{\}; Q (\text{Max } I) \rrbracket \implies P t'. t' \mathcal{S} t I. Q t$$

*<proof>*

### 3.2.6 Congruence rules for temporal operators' predicates

**lemma** *iAll-cong*:  $\Box t I. f t = g t \implies (\Box t I. P (f t) t) = (\Box t I. P (g t) t)$

*<proof>*

**lemma** *iEx-cong*:  $\Box t I. f t = g t \implies (\Diamond t I. P (f t) t) = (\Diamond t I. P (g t) t)$

*<proof>*

**lemma** *iUntil-cong1*:

$$\Box t I. f t = g t \implies (P (f t1) t1. t1 \mathcal{U} t2 I. Q t2) = (P (g t1) t1. t1 \mathcal{U} t2 I. Q t2)$$

*<proof>*

**lemma** *iUntil-cong2*:

$$\Box t I. f t = g t \implies (P t1. t1 \mathcal{U} t2 I. Q (f t2) t2) = (P t1. t1 \mathcal{U} t2 I. Q (g t2) t2)$$

*<proof>*

**lemma** *iSince-cong1*:

$$\Box t I. f t = g t \implies (P (f t1) t1. t1 \mathcal{S} t2 I. Q t2) = (P (g t1) t1. t1 \mathcal{S} t2 I. Q t2)$$

*<proof>*

**lemma** *iSince-cong2*:

$$\Box t I. f t = g t \implies (P t1. t1 \mathcal{S} t2 I. Q (f t2) t2) = (P t1. t1 \mathcal{S} t2 I. Q (g t2) t2)$$

*<proof>*

**lemma** *bex-subst*:

$$\forall x \in A. P x \longrightarrow (Q x = Q' x) \implies (\exists x \in A. P x \wedge Q x) = (\exists x \in A. P x \wedge Q' x)$$

*<proof>*

**lemma** *iEx-subst*:

$$\Box t I. P t \longrightarrow (Q t = Q' t) \implies (\Diamond t I. P t \wedge Q t) = (\Diamond t I. P t \wedge Q' t)$$

*<proof>*

### 3.2.7 Temporal operators with set unions/intersections and subsets

**lemma** *iAll-subset*:  $\llbracket A \subseteq B; \Box t B. P t \rrbracket \implies \Box t A. P t$

*<proof>*

**lemma** *iEx-subset*:  $\llbracket A \subseteq B; \Diamond t A. P t \rrbracket \implies \Diamond t B. P t$



$\langle \text{proof} \rangle$

**lemma** *iUntil-append*:

$\llbracket \text{finite } A; \text{Max } A \leq \text{iMin } B \rrbracket \implies$   
 $P \ t1. \ t1 \ \mathcal{U} \ t2 \ A. \ Q \ t2 \implies P \ t1. \ t1 \ \mathcal{U} \ t2 \ (A \cup B). \ Q \ t2$   
 $\langle \text{proof} \rangle$

**lemma** *iSince-prepend*:

$\llbracket \text{finite } A; \text{Max } A \leq \text{iMin } B \rrbracket \implies$   
 $P \ t1. \ t1 \ \mathcal{S} \ t2 \ B. \ Q \ t2 \implies P \ t1. \ t1 \ \mathcal{S} \ t2 \ (A \cup B). \ Q \ t2$   
 $\langle \text{proof} \rangle$

**lemma**

*iAll-union*:  $\llbracket \Box \ t \ A. \ P \ t; \Box \ t \ B. \ P \ t \rrbracket \implies \Box \ t \ (A \cup B). \ P \ t$  **and**  
*iAll-union-conv*:  $(\Box \ t \ A \cup B. \ P \ t) = ((\Box \ t \ A. \ P \ t) \wedge (\Box \ t \ B. \ P \ t))$   
 $\langle \text{proof} \rangle$

**lemma**

*iEx-union*:  $(\Diamond \ t \ A. \ P \ t) \vee (\Diamond \ t \ B. \ P \ t) \implies \Diamond \ t \ (A \cup B). \ P \ t$  **and**  
*iEx-union-conv*:  $(\Diamond \ t \ (A \cup B). \ P \ t) = ((\Diamond \ t \ A. \ P \ t) \vee (\Diamond \ t \ B. \ P \ t))$   
 $\langle \text{proof} \rangle$

**lemma** *iAll-inter*:  $(\Box \ t \ A. \ P \ t) \vee (\Box \ t \ B. \ P \ t) \implies \Box \ t \ (A \cap B). \ P \ t$   $\langle \text{proof} \rangle$

**lemma** *not-iEx-inter*:

$\exists A \ B \ P. (\Diamond \ t \ A. \ P \ t) \wedge (\Diamond \ t \ B. \ P \ t) \wedge \neg (\Diamond \ t \ (A \cap B). \ P \ t)$   
 $\langle \text{proof} \rangle$

**lemma**

*iAll-insert*:  $\llbracket P \ t; \Box \ t \ I. \ P \ t \rrbracket \implies \Box \ t' \ (\text{insert } t \ I). \ P \ t'$  **and**  
*iAll-insert-conv*:  $(\Box \ t' \ (\text{insert } t \ I). \ P \ t') = (P \ t \wedge (\Box \ t' \ I. \ P \ t'))$   
 $\langle \text{proof} \rangle$

**lemma**

*iEx-insert*:  $\llbracket P \ t \vee (\Diamond \ t \ I. \ P \ t) \rrbracket \implies \Diamond \ t' \ (\text{insert } t \ I). \ P \ t'$  **and**  
*iEx-insert-conv*:  $(\Diamond \ t' \ (\text{insert } t \ I). \ P \ t') = (P \ t \vee (\Diamond \ t' \ I. \ P \ t'))$   
 $\langle \text{proof} \rangle$

### 3.3 Further results for temporal operators

**lemma** *Collect-minI-iEx*:  $\Diamond \ t \ I. \ P \ t \implies \Diamond \ t \ I. \ P \ t \wedge (\Box \ t' \ (I \downarrow < t). \neg P \ t')$   
 $\langle \text{proof} \rangle$

**lemma** *iUntil-disj-conv1*:

$I \neq \{\}$   $\implies$   
 $(P \ t'. \ t' \ \mathcal{U} \ t \ I. \ Q \ t) = (Q \ (iMin \ I) \vee (P \ t'. \ t' \ \mathcal{U} \ t \ I. \ Q \ t \wedge iMin \ I < t))$   
 $\langle \text{proof} \rangle$

**lemma** *iSince-disj-conv1*:

$\llbracket \text{finite } I; I \neq \{\} \rrbracket \implies$   
 $(P t'. t' \mathcal{S} t I. Q t) = (Q (\text{Max } I) \vee (P t'. t' \mathcal{S} t I. Q t \wedge t < \text{Max } I))$   
 <proof>

**lemma** *iUntil-next*:

$I \neq \{\} \implies$   
 $(P t'. t' \mathcal{U} t I. Q t) =$   
 $(Q (\text{iMin } I) \vee (P (\text{iMin } I) \wedge (P t'. t' \mathcal{U} t (I \downarrow > (\text{iMin } I)). Q t)))$   
 <proof>

**lemma** *iSince-prev*:  $\llbracket \text{finite } I; I \neq \{\} \rrbracket \implies$

$(P t'. t' \mathcal{S} t I. Q t) =$   
 $(Q (\text{Max } I) \vee (P (\text{Max } I) \wedge (P t'. t' \mathcal{S} t (I \downarrow < \text{Max } I). Q t)))$   
 <proof>

**lemma** *iNext-induct-rule*:

$\llbracket P (\text{iMin } I); \square t I. (P t \longrightarrow (\bigcirc t' t I. P t')) \rrbracket \implies P t$   
 <proof>

**lemma** *iNext-induct*:

$\llbracket P (\text{iMin } I); \square t I. (P t \longrightarrow (\bigcirc t' t I. P t')) \rrbracket \implies \square t I. P t$   
 <proof>

**lemma** *iLast-induct-rule*:

$\llbracket P (\text{Max } I); \square t I. (P t \longrightarrow (\ominus t' t I. P t')) \rrbracket; \text{finite } I; t \in I \implies P t$   
 <proof>

**lemma** *iLast-induct*:

$\llbracket P (\text{Max } I); \square t I. (P t \longrightarrow (\ominus t' t I. P t')) \rrbracket; \text{finite } I \implies \square t I. P t$   
 <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*:  $\text{finite } I \implies$

$((P t1 \wedge \neg Q t1). t1 \mathcal{S} t2 I. Q t2) = (P t1. t1 \mathcal{S} t2 I. Q t2)$   
 <proof>

**lemma** *iWeakSince-conj-not*:  $\text{finite } I \implies$

$((P t1 \wedge \neg Q t1). t1 \mathcal{B} t2 I. Q t2) = (P t1. t1 \mathcal{B} t2 I. Q t2)$   
 <proof>

**lemma** *iNextStrong-imp-iNextWeak*:  $(\circ_S t t0 I. P t) \longrightarrow (\circ_W t t0 I. P t)$   
 ⟨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*  $I$ ;  $t0 \in I$  ]]  $\implies$   
 $((\circ_W t t0 I. P t) = (\circ t t0 I. P t)) \wedge ((\circ_S t t0 I. P t) = (\circ t t0 I. P t))$   
 ⟨proof⟩

**lemma** *infin-imp-iNextWeak-eq-iNext*: [[ *infinite*  $I$ ;  $t0 \in I$  ]]  $\implies (\circ_W t t0 I. P t)$   
 $= (\circ t t0 I. P t)$

⟨proof⟩

**lemma** *infin-imp-iNextStrong-eq-iNext*: [[ *infinite*  $I$ ;  $t0 \in I$  ]]  $\implies (\circ_S t t0 I. P t)$   
 $= (\circ t t0 I. P t)$

⟨proof⟩

**lemma** *infin-imp-iNextStrong-eq-iNextWeak*: [[ *infinite*  $I$ ;  $t0 \in I$  ]]  $\implies (\circ_S t t0 I.$   
 $P t) = (\circ_W t t0 I. P t)$

⟨proof⟩

**lemma**

*not-in-iNext-eq*:  $t0 \notin I \implies (\circ t t0 I. P t) = (P t0)$  **and**

*not-in-iLast-eq*:  $t0 \notin I \implies (\ominus t t0 I. P t) = (P t0)$

⟨proof⟩

**lemma**

*not-in-iNextWeak-eq*:  $t0 \notin I \implies (\circ_W t t0 I. P t)$  **and**

*not-in-iLastWeak-eq*:  $t0 \notin I \implies (\ominus_W t t0 I. P t)$

⟨proof⟩

**lemma**

*not-in-iNextStrong-eq*:  $t0 \notin I \implies \neg (\circ_S t t0 I. P t)$  **and**

*not-in-iLastStrong-eq*:  $t0 \notin I \implies \neg (\ominus_S t t0 I. P t)$

⟨proof⟩

**lemma**

*iNext-UNIV*:  $(\circ t t0 UNIV. P t) = P (Suc t0)$  **and**

*iNextWeak-UNIV*:  $(\circ_W t t0 UNIV. P t) = P (Suc t0)$  **and**

*iNextStrong-UNIV*:  $(\circ_S t t0 UNIV. P t) = P (Suc t0)$

⟨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*:  $(\circ t (I \rightarrow n) I. P t) = P (I \rightarrow \text{Suc } n)$   
 $\langle \text{proof} \rangle$

**lemma** *iprev-nth-iLast-Suc*:  $(\ominus t (I \leftarrow n) I. P t) = P (I \leftarrow \text{Suc } n)$   
 $\langle \text{proof} \rangle$

### 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*:  $(\Box t I \oplus k. P t) = (\Box t I. P (t + k))$   
 $\langle \text{proof} \rangle$

**lemma** *iT-Mult-iAll-conv*:  $(\Box t I \otimes k. P t) = (\Box t I. P (t * k))$   
 $\langle \text{proof} \rangle$

**lemma** *iT-Plus-neg-iAll-conv*:  $(\Box t I \oplus - k. P t) = (\Box t (I \downarrow \geq k). P (t - k))$   
 $\langle \text{proof} \rangle$

**lemma** *iT-Minus-iAll-conv*:  $(\Box t k \ominus I. P t) = (\Box t (I \downarrow \leq k). P (k - t))$   
 $\langle \text{proof} \rangle$

**lemma** *iT-Div-iAll-conv*:  $(\Box t I \oslash k. P t) = (\Box t I. P (t \text{ div } k))$   
 $\langle \text{proof} \rangle$

**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*:  $(\Diamond t I \oslash k. P t) = (\Diamond t I. P (t \text{ div } k))$   
 $\langle \text{proof} \rangle$

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*:  $(P\ t1.\ t1\ \mathcal{U}\ t2\ (I\ \odot\ k).\ Q\ t2) = (P\ (t1\ \text{div}\ k).\ t1\ \mathcal{U}\ t2\ I.\ Q\ (t2\ \text{div}\ k))$   
 ⟨proof⟩

Until and Since operators can be converted into each other through subtraction of intervals from constants

**lemma** *iUntil-iSince-conv*:  
 $\llbracket\ \text{finite}\ I;\ \text{Max}\ I\ \leq\ k\ \rrbracket\ \Longrightarrow$   
 $(P\ t1.\ t1\ \mathcal{U}\ t2\ I.\ Q\ t2) = (P\ (k\ -\ t1).\ t1\ \mathcal{S}\ t2\ (k\ \ominus\ I).\ Q\ (k\ -\ t2))$   
 ⟨proof⟩

**lemma** *iSince-iUntil-conv*:  
 $\llbracket\ \text{finite}\ I;\ \text{Max}\ I\ \leq\ k\ \rrbracket\ \Longrightarrow$   
 $(P\ t1.\ t1\ \mathcal{S}\ t2\ I.\ Q\ t2) = (P\ (k\ -\ t1).\ t1\ \mathcal{U}\ t2\ (k\ \ominus\ I).\ Q\ (k\ -\ t2))$   
 ⟨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*:

$0 < k \implies (P \ t1. \ t1 \ \mathcal{S} \ t2 \ (I \circledast k). \ Q \ t2) = (P \ (t1 \ \text{div} \ k). \ t1 \ \mathcal{S} \ t2 \ I. \ Q \ (t2 \ \text{div} \ k))$   
 ⟨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*:  $(P \ t1. \ t1 \ \mathcal{W} \ t2 \ (I \circledast k). \ Q \ t2) = (P \ (t1 \ \text{div} \ k). \ t1 \ \mathcal{W} \ t2 \ I. \ Q \ (t2 \ \text{div} \ k))$   
 ⟨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*:  $0 < k \implies (P \ t1. \ t1 \ \mathcal{B} \ t2 \ (I \otimes k). \ Q \ t2) = (P \ (t1 * k). \ t1 \ \mathcal{B} \ t2 \ I. \ Q \ (t2 * k))$   
 ⟨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*:  
 $0 < k \implies (P \ t1. \ t1 \ \mathcal{B} \ t2 \ (I \circledast k). \ Q \ t2) = (P \ (t1 \ \text{div} \ k). \ t1 \ \mathcal{B} \ t2 \ I. \ Q \ (t2 \ \text{div} \ k))$   
 ⟨proof⟩

Release and Trigger operators

**lemma** *iT-Plus-iRelease-conv*:  $(P \ t1. \ t1 \ \mathcal{R} \ t2 \ (I \oplus k). \ Q \ t2) = (P \ (t1 + k). \ t1$

$\mathcal{R} \ t2 \ I. \ Q \ (t2 + k)$   
 $\langle proof \rangle$

**lemma** *iT-Mult-iRelease-conv*:  $(P \ t1. \ t1 \ \mathcal{R} \ t2 \ (I \otimes k). \ Q \ t2) = (P \ (t1 * k). \ t1 \ \mathcal{R} \ t2 \ I. \ Q \ (t2 * k))$   
 $\langle proof \rangle$

**lemma** *iT-Plus-neg-iRelease-conv*:  $(P \ t1. \ t1 \ \mathcal{R} \ t2 \ (I \oplus - k). \ Q \ t2) = (P \ (t1 - k). \ t1 \ \mathcal{R} \ t2 \ (I \downarrow \geq k). \ Q \ (t2 - k))$   
 $\langle proof \rangle$

**lemma** *iT-Minus-iRelease-conv*:  $(P \ t1. \ t1 \ \mathcal{R} \ t2 \ (k \ominus I). \ Q \ t2) = (P \ (k - t1). \ t1 \ \mathcal{T} \ t2 \ (I \downarrow \leq k). \ Q \ (k - t2))$   
 $\langle proof \rangle$

**lemma** *iT-Div-iRelease-conv*:  $(P \ t1. \ t1 \ \mathcal{R} \ t2 \ (I \circ k). \ Q \ t2) = (P \ (t1 \ \text{div} \ k). \ t1 \ \mathcal{R} \ t2 \ I. \ Q \ (t2 \ \text{div} \ k))$   
 $\langle proof \rangle$

**lemma** *iT-Plus-iTrigger-conv*:  $(P \ t1. \ t1 \ \mathcal{T} \ t2 \ (I \oplus k). \ Q \ t2) = (P \ (t1 + k). \ t1 \ \mathcal{T} \ t2 \ I. \ Q \ (t2 + k))$   
 $\langle proof \rangle$

**lemma** *iT-Mult-iTrigger-conv*:  $0 < k \implies (P \ t1. \ t1 \ \mathcal{T} \ t2 \ (I \otimes k). \ Q \ t2) = (P \ (t1 * k). \ t1 \ \mathcal{T} \ t2 \ I. \ Q \ (t2 * k))$   
 $\langle proof \rangle$

**lemma** *iT-Plus-neg-iTrigger-conv*:  $(P \ t1. \ t1 \ \mathcal{T} \ t2 \ (I \oplus - k). \ Q \ t2) = (P \ (t1 - k). \ t1 \ \mathcal{T} \ t2 \ (I \downarrow \geq k). \ Q \ (t2 - k))$   
 $\langle proof \rangle$

**lemma** *iT-Minus-iTrigger-conv*:  
 $(P \ t1. \ t1 \ \mathcal{T} \ t2 \ (k \ominus I). \ Q \ t2) = (P \ (k - t1). \ t1 \ \mathcal{R} \ t2 \ (I \downarrow \leq k). \ Q \ (k - t2))$   
 $\langle proof \rangle$

**lemma** *iT-Div-iTrigger-conv*:  
 $0 < k \implies (P \ t1. \ t1 \ \mathcal{T} \ t2 \ (I \circ k). \ Q \ t2) = (P \ (t1 \ \text{div} \ k). \ t1 \ \mathcal{T} \ t2 \ I. \ Q \ (t2 \ \text{div} \ k))$   
 $\langle proof \rangle$

**end**