What's in Main

Tobias Nipkow

June 21, 2010

Abstract

This document lists the main types, functions and syntax provided by theory *Main*. It is meant as a quick overview of what is available. The sophisticated class structure is only hinted at. For details see http://isabelle.in.tum.de/dist/library/HOL/.

1 HOL

The basic logic: $x = y$, *True*, *False*, $\neg P$, $P \wedge Q$, $P \vee Q$, $P \longrightarrow Q$, $\forall x.\ P$, $\exists x.\ P$, $\exists!x.\ P$, $THE\ x.\ P$.

$undefined :: 'a$
$default :: 'a$

Syntax

$x \neq y$	$\equiv$	$\neg (x = y)$	(~=)
$P \longleftrightarrow Q$	$\equiv$	$P = Q$	
$if\ x\ then\ y\ else\ z$	$\equiv$	$If\ x\ y\ z$	
$let\ x = e_1\ in\ e_2$	$\equiv$	$Let\ e_1\ (\lambda x.\ e_2)$	

2 Orderings

A collection of classes defining basic orderings: preorder, partial order, linear order, dense linear order and wellorder.

$op \leq :: 'a \Rightarrow 'a \Rightarrow bool$	(<=)
$op < :: 'a \Rightarrow 'a \Rightarrow bool$	
$Least :: ('a \Rightarrow bool) \Rightarrow 'a$	
$min :: 'a \Rightarrow 'a \Rightarrow 'a$	
$max :: 'a \Rightarrow 'a \Rightarrow 'a$	
$top :: 'a$	

bot	:: $'a$	
mono	:: $('a \Rightarrow 'b) \Rightarrow bool$	
strict-mono	:: $('a \Rightarrow 'b) \Rightarrow bool$	

Syntax

$x \geq y$	$\equiv$	$y \leq x$	`(>=)`
$x > y$	$\equiv$	$y < x$	
$\forall x \leq y.\ P$	$\equiv$	$\forall x.\ x \leq y \longrightarrow P$	
$\exists x \leq y.\ P$	$\equiv$	$\exists x.\ x \leq y \wedge P$	

Similarly for $<$, $\geq$ and $>$

$LEAST\ x.\ P$	$\equiv$	$Least\ (\lambda x.\ P)$

3 Lattices

Classes semilattice, lattice, distributive lattice and complete lattice (the latter in theory *Set*).

inf	:: $'a \Rightarrow 'a \Rightarrow 'a$
sup	:: $'a \Rightarrow 'a \Rightarrow 'a$
Inf	:: $'a\ set \Rightarrow 'a$
Sup	:: $'a\ set \Rightarrow 'a$

Syntax

Available by loading theory *Lattice-Syntax* in directory *Library*.

$x \sqsubseteq y$	$\equiv$	$x \leq y$
$x \sqsubset y$	$\equiv$	$x < y$
$x \sqcap y$	$\equiv$	$inf\ x\ y$
$x \sqcup y$	$\equiv$	$sup\ x\ y$
$\bigsqcap A$	$\equiv$	$Sup\ A$
$\bigsqcup A$	$\equiv$	$Inf\ A$
$\top$	$\equiv$	top
$\bot$	$\equiv$	bot

4 Set

Sets are predicates: $'a\ set\ =\ 'a \Rightarrow bool$

$\{\}$	:: $'a\ set$	
insert	:: $'a \Rightarrow 'a\ set \Rightarrow 'a\ set$	
Collect	:: $('a \Rightarrow bool) \Rightarrow 'a\ set$	
op $\in$	:: $'a \Rightarrow 'a\ set \Rightarrow bool$	`(:)`
op $\cup$	:: $'a\ set \Rightarrow 'a\ set \Rightarrow 'a\ set$	`(Un)`

$op \cap$	$:: {'a}\ set \Rightarrow {'a}\ set \Rightarrow {'a}\ set$	(`Int`)
$UNION$	$:: {'a}\ set \Rightarrow ({'a} \Rightarrow {'b}\ set) \Rightarrow {'b}\ set$	
$INTER$	$:: {'a}\ set \Rightarrow ({'a} \Rightarrow {'b}\ set) \Rightarrow {'b}\ set$	
$Union$	$:: {'a}\ set\ set \Rightarrow {'a}\ set$	
$Inter$	$:: {'a}\ set\ set \Rightarrow {'a}\ set$	
Pow	$:: {'a}\ set \Rightarrow {'a}\ set\ set$	
$UNIV$	$:: {'a}\ set$	
op ‘	$:: ({'a} \Rightarrow {'b}) \Rightarrow {'a}\ set \Rightarrow {'b}\ set$	
$Ball$	$:: {'a}\ set \Rightarrow ({'a} \Rightarrow bool) \Rightarrow bool$	
Bex	$:: {'a}\ set \Rightarrow ({'a} \Rightarrow bool) \Rightarrow bool$	

Syntax

$\{x_1,\ldots,x_n\}$	$\equiv$	$insert\ x_1\ (\ldots\ (insert\ x_n\ \{\})\ldots)$	
$x \notin A$	$\equiv$	$\neg(x \in A)$	
$A \subseteq B$	$\equiv$	$A \leq B$	
$A \subset B$	$\equiv$	$A < B$	
$A \supseteq B$	$\equiv$	$B \leq A$	
$A \supset B$	$\equiv$	$B < A$	
$\{x.\ P\}$	$\equiv$	$Collect\ (\lambda x.\ P)$	
$\bigcup x{\in}I.\ A$	$\equiv$	$UNION\ I\ (\lambda x.\ A)$	(`UN`)
$\bigcup x.\ A$	$\equiv$	$UNION\ UNIV\ (\lambda x.\ A)$	
$\bigcap x{\in}I.\ A$	$\equiv$	$INTER\ I\ (\lambda x.\ A)$	(`INT`)
$\bigcap x.\ A$	$\equiv$	$INTER\ UNIV\ (\lambda x.\ A)$	
$\forall x{\in}A.\ P$	$\equiv$	$Ball\ A\ (\lambda x.\ P)$	
$\exists x{\in}A.\ P$	$\equiv$	$Bex\ A\ (\lambda x.\ P)$	
$range\ f$	$\equiv$	f ‘ $UNIV$	

5 Fun

id	$:: {'a} \Rightarrow {'a}$	
$op \circ$	$:: ({'a} \Rightarrow {'b}) \Rightarrow ({'c} \Rightarrow {'a}) \Rightarrow {'c} \Rightarrow {'b}$	(`o`)
$inj\text{-}on$	$:: ({'a} \Rightarrow {'b}) \Rightarrow {'a}\ set \Rightarrow bool$	
inj	$:: ({'a} \Rightarrow {'b}) \Rightarrow bool$	
$surj$	$:: ({'a} \Rightarrow {'b}) \Rightarrow bool$	
bij	$:: ({'a} \Rightarrow {'b}) \Rightarrow bool$	
$bij\text{-}betw$	$:: ({'a} \Rightarrow {'b}) \Rightarrow {'a}\ set \Rightarrow {'b}\ set \Rightarrow bool$	
$fun\text{-}upd$	$:: ({'a} \Rightarrow {'b}) \Rightarrow {'a} \Rightarrow {'b} \Rightarrow {'a} \Rightarrow {'b}$	

Syntax

$f(x := y)$	$\equiv$	$fun\text{-}upd\ f\ x\ y$
$f(x_1{:=}y_1,\ldots,x_n{:=}y_n)$	$\equiv$	$f(x_1{:=}y_1)\ldots(x_n{:=}y_n)$

6 Hilbert_Choice

Hilbert's selection (ε) operator: *SOME* x. P.

$inv\text{-}into :: 'a\ set \Rightarrow ('a \Rightarrow 'b) \Rightarrow 'b \Rightarrow 'a$

Syntax

$inv \quad \equiv \quad inv\text{-}into\ UNIV$

7 Fixed Points

Theory: *Inductive*.

Least and greatest fixed points in a complete lattice $'a$:
$lfp :: ('a \Rightarrow 'a) \Rightarrow 'a$
$gfp :: ('a \Rightarrow 'a) \Rightarrow 'a$
Note that in particular sets ($'a \Rightarrow bool$) are complete lattices.

8 Sum_Type

Type constructor $+$.

Inl	$:: 'a \Rightarrow 'a + 'b$
Inr	$:: 'a \Rightarrow 'b + 'a$
op <+>	$:: 'a\ set \Rightarrow 'b\ set \Rightarrow ('a + 'b)\ set$

9 Product_Type

Types *unit* and $\times$.

$()$	$:: unit$
$Pair$	$:: 'a \Rightarrow 'b \Rightarrow 'a \times 'b$
fst	$:: 'a \times 'b \Rightarrow 'a$
snd	$:: 'a \times 'b \Rightarrow 'b$
$split$	$:: ('a \Rightarrow 'b \Rightarrow 'c) \Rightarrow 'a \times 'b \Rightarrow 'c$
$curry$	$:: ('a \times 'b \Rightarrow 'c) \Rightarrow 'a \Rightarrow 'b \Rightarrow 'c$
$Sigma$	$:: 'a\ set \Rightarrow ('a \Rightarrow 'b\ set) \Rightarrow ('a \times 'b)\ set$

Syntax

(a, b)	$\equiv$	$Pair\ a\ b$	
$\lambda(x, y).\ t$	$\equiv$	$split\ (\lambda x\ y.\ t)$	
$A \times B$	$\equiv$	$Sigma\ A\ (\lambda\_.\ B)$	(<*>)

Pairs may be nested. Nesting to the right is printed as a tuple, e.g. (a, b, c) is really $(a, (b, c))$. Pattern matching with pairs and tuples extends to all binders, e.g. $\forall (x, y)\in A.\ P$, $\{(x, y).\ P\}$, etc.

10 Relation

converse	$::\ ('a \times 'b)\ set \Rightarrow ('b \times 'a)\ set$
op O	$::\ ('a \times 'b)\ set \Rightarrow ('b \times 'c)\ set \Rightarrow ('a \times 'c)\ set$
op ``	$::\ ('a \times 'b)\ set \Rightarrow 'a\ set \Rightarrow 'b\ set$
inv-image	$::\ ('a \times 'a)\ set \Rightarrow ('b \Rightarrow 'a) \Rightarrow ('b \times 'b)\ set$
Id-on	$::\ 'a\ set \Rightarrow ('a \times 'a)\ set$
Id	$::\ ('a \times 'a)\ set$
Domain	$::\ ('a \times 'b)\ set \Rightarrow 'a\ set$
Range	$::\ ('a \times 'b)\ set \Rightarrow 'b\ set$
Field	$::\ ('a \times 'a)\ set \Rightarrow 'a\ set$
refl-on	$::\ 'a\ set \Rightarrow ('a \times 'a)\ set \Rightarrow bool$
refl	$::\ ('a \times 'a)\ set \Rightarrow bool$
sym	$::\ ('a \times 'a)\ set \Rightarrow bool$
antisym	$::\ ('a \times 'a)\ set \Rightarrow bool$
trans	$::\ ('a \times 'a)\ set \Rightarrow bool$
irrefl	$::\ ('a \times 'a)\ set \Rightarrow bool$
total-on	$::\ 'a\ set \Rightarrow ('a \times 'a)\ set \Rightarrow bool$
total	$::\ ('a \times 'a)\ set \Rightarrow bool$

Syntax

r^{-1} $\equiv$ *converse r* (`^-1`)

11 Equiv_Relations

equiv	$::\ 'a\ set \Rightarrow ('a \times 'a)\ set \Rightarrow bool$
op //	$::\ 'a\ set \Rightarrow ('a \times 'a)\ set \Rightarrow 'a\ set\ set$
congruent	$::\ ('a \times 'a)\ set \Rightarrow ('a \Rightarrow 'b) \Rightarrow bool$
congruent2	$::\ ('a \times 'a)\ set \Rightarrow ('b \times 'b)\ set \Rightarrow ('a \Rightarrow 'b \Rightarrow 'c) \Rightarrow bool$

Syntax

f respects r $\equiv$ *congruent r f*
f respects2 r $\equiv$ *congruent2 r r f*

12 Transitive_Closure

rtrancl	$::\ ('a \times 'a)\ set \Rightarrow ('a \times 'a)\ set$
trancl	$::\ ('a \times 'a)\ set \Rightarrow ('a \times 'a)\ set$
reflcl	$::\ ('a \times 'a)\ set \Rightarrow ('a \times 'a)\ set$
*op ^^*	$::\ ('a \times 'a)\ set \Rightarrow nat \Rightarrow ('a \times 'a)\ set$

Syntax

r^*	$\equiv$	*rtrancl r*	`(^*)`
r^+	$\equiv$	*trancl r*	`(^+)`
$r^=$	$\equiv$	*reflcl r*	`(^=)`

13 Algebra

Theories *Groups*, *Rings*, *Fields* and *Divides* define a large collection of classes describing common algebraic structures from semigroups up to fields. Everything is done in terms of overloaded operators:

0	$:: 'a$	
1	$:: 'a$	
op $+$	$:: 'a \Rightarrow 'a \Rightarrow 'a$	
op $-$	$:: 'a \Rightarrow 'a \Rightarrow 'a$	
uminus	$:: 'a \Rightarrow 'a$	`(-)`
op $*$	$:: 'a \Rightarrow 'a \Rightarrow 'a$	
inverse	$:: 'a \Rightarrow 'a$	
op $/$	$:: 'a \Rightarrow 'a \Rightarrow 'a$	
abs	$:: 'a \Rightarrow 'a$	
sgn	$:: 'a \Rightarrow 'a$	
op dvd	$:: 'a \Rightarrow 'a \Rightarrow bool$	
op div	$:: 'a \Rightarrow 'a \Rightarrow 'a$	
op mod	$:: 'a \Rightarrow 'a \Rightarrow 'a$	

Syntax

$|x| \quad \equiv \quad abs\ x$

14 Nat

datatype *nat* $=$ *0* $|$ *Suc nat*

op $+$	*op* $-$	*op* $*$	*op div*	*op mod*	*op dvd*
op $\leq$	*op* $<$	*min*	*max*	*Min*	*Max*

of-nat $:: nat \Rightarrow 'a$
op ^^ $:: ('a \Rightarrow 'a) \Rightarrow nat \Rightarrow 'a \Rightarrow 'a$

15 Int

Type *int*

$op +$	$op -$	$uminus$	$op *$	op ^	$op\ div$	$op\ mod$	$op\ dvd$
$op \le$	$op <$	min	max	Min	Max		
abs	sgn						

nat :: $int \Rightarrow nat$
$of\text{-}int$ :: $int \Rightarrow 'a$
$\mathbb{Z}$ :: $'a\ set$ (`Ints`)

Syntax

$int \equiv of\text{-}nat$

16 Finite_Set

$finite$ :: $'a\ set \Rightarrow bool$
$card$ :: $'a\ set \Rightarrow nat$
$fold$ :: $('a \Rightarrow 'b \Rightarrow 'b) \Rightarrow 'b \Rightarrow 'a\ set \Rightarrow 'b$
$fold\text{-}image$ :: $('b \Rightarrow 'b \Rightarrow 'b) \Rightarrow ('a \Rightarrow 'b) \Rightarrow 'b \Rightarrow 'a\ set \Rightarrow 'b$
$setsum$ :: $('a \Rightarrow 'b) \Rightarrow 'a\ set \Rightarrow 'b$
$setprod$ :: $('a \Rightarrow 'b) \Rightarrow 'a\ set \Rightarrow 'b$

Syntax

$\sum A \equiv setsum\ (\lambda x.\ x)\ A$ (`SUM`)
$\sum x{\in}A.\ t \equiv setsum\ (\lambda x.\ t)\ A$
$\sum x|P.\ t \equiv \sum x \mid P.\ t$
Similarly for $\prod$ instead of $\sum$ (`PROD`)

17 Wellfounded

wf :: $('a \times 'a)\ set \Rightarrow bool$
$acyclic$ :: $('a \times 'a)\ set \Rightarrow bool$
acc :: $('a \times 'a)\ set \Rightarrow 'a\ set$
$measure$ :: $('a \Rightarrow nat) \Rightarrow ('a \times 'a)\ set$
$op <*lex*>$ :: $('a \times 'a)\ set \Rightarrow ('b \times 'b)\ set \Rightarrow (('a \times 'b) \times 'a \times 'b)\ set$
$op <*mlex*>$ :: $('a \Rightarrow nat) \Rightarrow ('a \times 'a)\ set \Rightarrow ('a \times 'a)\ set$
$less\text{-}than$ :: $(nat \times nat)\ set$
$pred\text{-}nat$ :: $(nat \times nat)\ set$

18 SetInterval

$lessThan$ :: $'a \Rightarrow 'a\ set$
$atMost$ :: $'a \Rightarrow 'a\ set$
$greaterThan$ :: $'a \Rightarrow 'a\ set$

atLeast	$:: 'a \Rightarrow 'a\ set$
greaterThanLessThan	$:: 'a \Rightarrow 'a \Rightarrow 'a\ set$
atLeastLessThan	$:: 'a \Rightarrow 'a \Rightarrow 'a\ set$
greaterThanAtMost	$:: 'a \Rightarrow 'a \Rightarrow 'a\ set$
atLeastAtMost	$:: 'a \Rightarrow 'a \Rightarrow 'a\ set$

Syntax

$\{..<y\}$	$\equiv$	$lessThan\ y$
$\{..y\}$	$\equiv$	$atMost\ y$
$\{x<..\}$	$\equiv$	$greaterThan\ x$
$\{x..\}$	$\equiv$	$atLeast\ x$
$\{x<..<y\}$	$\equiv$	$greaterThanLessThan\ x\ y$
$\{x..<y\}$	$\equiv$	$atLeastLessThan\ x\ y$
$\{x<..y\}$	$\equiv$	$greaterThanAtMost\ x\ y$
$\{x..y\}$	$\equiv$	$atLeastAtMost\ x\ y$
$\bigcup\ i{\le}n.\ A$	$\equiv$	$\bigcup\ i \in \{..n\}.\ A$
$\bigcup\ i{<}n.\ A$	$\equiv$	$\bigcup\ i \in \{..<n\}.\ A$
Similarly for $\bigcap$ instead of $\bigcup$		
$\sum x = a..b.\ t$	$\equiv$	$setsum\ (\lambda x.\ t)\ \{a..b\}$
$\sum x = a..<b.\ t$	$\equiv$	$setsum\ (\lambda x.\ t)\ \{a..<b\}$
$\sum x{\le}b.\ t$	$\equiv$	$setsum\ (\lambda x.\ t)\ \{..b\}$
$\sum x{<}b.\ t$	$\equiv$	$setsum\ (\lambda x.\ t)\ \{..<b\}$
Similarly for $\prod$ instead of $\sum$		

19 Power

$op\ \hat{}\ :: 'a \Rightarrow nat \Rightarrow 'a$

20 Option

datatype $'a\ option = None \mid Some\ 'a$

the	$:: 'a\ option \Rightarrow 'a$
Option.map	$:: ('a \Rightarrow 'b) \Rightarrow 'a\ option \Rightarrow 'b\ option$
Option.set	$:: 'a\ option \Rightarrow 'a\ set$

21 List

datatype $'a\ list = [] \mid op\ \#\ 'a\ ('a\ list)$

$op\ @ :: 'a\ list \Rightarrow 'a\ list \Rightarrow 'a\ list$

$$
\begin{array}{ll}
\mathit{butlast} & :: {}'a\ \mathit{list} \Rightarrow {}'a\ \mathit{list} \\
\mathit{concat} & :: {}'a\ \mathit{list}\ \mathit{list} \Rightarrow {}'a\ \mathit{list} \\
\mathit{distinct} & :: {}'a\ \mathit{list} \Rightarrow \mathit{bool} \\
\mathit{drop} & :: \mathit{nat} \Rightarrow {}'a\ \mathit{list} \Rightarrow {}'a\ \mathit{list} \\
\mathit{dropWhile} & :: ({}'a \Rightarrow \mathit{bool}) \Rightarrow {}'a\ \mathit{list} \Rightarrow {}'a\ \mathit{list} \\
\mathit{filter} & :: ({}'a \Rightarrow \mathit{bool}) \Rightarrow {}'a\ \mathit{list} \Rightarrow {}'a\ \mathit{list} \\
\mathit{foldl} & :: ({}'a \Rightarrow {}'b \Rightarrow {}'a) \Rightarrow {}'a \Rightarrow {}'b\ \mathit{list} \Rightarrow {}'a \\
\mathit{foldr} & :: ({}'a \Rightarrow {}'b \Rightarrow {}'b) \Rightarrow {}'a\ \mathit{list} \Rightarrow {}'b \Rightarrow {}'b \\
\mathit{hd} & :: {}'a\ \mathit{list} \Rightarrow {}'a \\
\mathit{last} & :: {}'a\ \mathit{list} \Rightarrow {}'a \\
\mathit{length} & :: {}'a\ \mathit{list} \Rightarrow \mathit{nat} \\
\mathit{lenlex} & :: ({}'a \times {}'a)\ \mathit{set} \Rightarrow ({}'a\ \mathit{list} \times {}'a\ \mathit{list})\ \mathit{set} \\
\mathit{lex} & :: ({}'a \times {}'a)\ \mathit{set} \Rightarrow ({}'a\ \mathit{list} \times {}'a\ \mathit{list})\ \mathit{set} \\
\mathit{lexn} & :: ({}'a \times {}'a)\ \mathit{set} \Rightarrow \mathit{nat} \Rightarrow ({}'a\ \mathit{list} \times {}'a\ \mathit{list})\ \mathit{set} \\
\mathit{lexord} & :: ({}'a \times {}'a)\ \mathit{set} \Rightarrow ({}'a\ \mathit{list} \times {}'a\ \mathit{list})\ \mathit{set} \\
\mathit{listrel} & :: ({}'a \times {}'a)\ \mathit{set} \Rightarrow ({}'a\ \mathit{list} \times {}'a\ \mathit{list})\ \mathit{set} \\
\mathit{lists} & :: {}'a\ \mathit{set} \Rightarrow {}'a\ \mathit{list}\ \mathit{set} \\
\mathit{listset} & :: {}'a\ \mathit{set}\ \mathit{list} \Rightarrow {}'a\ \mathit{list}\ \mathit{set} \\
\mathit{listsum} & :: {}'a\ \mathit{list} \Rightarrow {}'a \\
\mathit{list\text{-}all2} & :: ({}'a \Rightarrow {}'b \Rightarrow \mathit{bool}) \Rightarrow {}'a\ \mathit{list} \Rightarrow {}'b\ \mathit{list} \Rightarrow \mathit{bool} \\
\mathit{list\text{-}update} & :: {}'a\ \mathit{list} \Rightarrow \mathit{nat} \Rightarrow {}'a \Rightarrow {}'a\ \mathit{list} \\
\mathit{map} & :: ({}'a \Rightarrow {}'b) \Rightarrow {}'a\ \mathit{list} \Rightarrow {}'b\ \mathit{list} \\
\mathit{measures} & :: ({}'a \Rightarrow \mathit{nat})\ \mathit{list} \Rightarrow ({}'a \times {}'a)\ \mathit{set} \\
\mathit{op}\ ! & :: {}'a\ \mathit{list} \Rightarrow \mathit{nat} \Rightarrow {}'a \\
\mathit{remdups} & :: {}'a\ \mathit{list} \Rightarrow {}'a\ \mathit{list} \\
\mathit{removeAll} & :: {}'a \Rightarrow {}'a\ \mathit{list} \Rightarrow {}'a\ \mathit{list} \\
\mathit{remove1} & :: {}'a \Rightarrow {}'a\ \mathit{list} \Rightarrow {}'a\ \mathit{list} \\
\mathit{replicate} & :: \mathit{nat} \Rightarrow {}'a \Rightarrow {}'a\ \mathit{list} \\
\mathit{rev} & :: {}'a\ \mathit{list} \Rightarrow {}'a\ \mathit{list} \\
\mathit{rotate} & :: \mathit{nat} \Rightarrow {}'a\ \mathit{list} \Rightarrow {}'a\ \mathit{list} \\
\mathit{rotate1} & :: {}'a\ \mathit{list} \Rightarrow {}'a\ \mathit{list} \\
\mathit{set} & :: {}'a\ \mathit{list} \Rightarrow {}'a\ \mathit{set} \\
\mathit{sort} & :: {}'a\ \mathit{list} \Rightarrow {}'a\ \mathit{list} \\
\mathit{sorted} & :: {}'a\ \mathit{list} \Rightarrow \mathit{bool} \\
\mathit{splice} & :: {}'a\ \mathit{list} \Rightarrow {}'a\ \mathit{list} \Rightarrow {}'a\ \mathit{list} \\
\mathit{sublist} & :: {}'a\ \mathit{list} \Rightarrow (\mathit{nat} \Rightarrow \mathit{bool}) \Rightarrow {}'a\ \mathit{list} \\
\mathit{take} & :: \mathit{nat} \Rightarrow {}'a\ \mathit{list} \Rightarrow {}'a\ \mathit{list} \\
\mathit{takeWhile} & :: ({}'a \Rightarrow \mathit{bool}) \Rightarrow {}'a\ \mathit{list} \Rightarrow {}'a\ \mathit{list} \\
\mathit{tl} & :: {}'a\ \mathit{list} \Rightarrow {}'a\ \mathit{list} \\
\mathit{upt} & :: \mathit{nat} \Rightarrow \mathit{nat} \Rightarrow \mathit{nat}\ \mathit{list} \\
\mathit{upto} & :: \mathit{int} \Rightarrow \mathit{int} \Rightarrow \mathit{int}\ \mathit{list}
\end{array}
$$

$zip :: 'a\ list \Rightarrow 'b\ list \Rightarrow ('a \times 'b)\ list$

Syntax

$[x_1,\ldots,x_n]$	$\equiv$	$x_1 \# \ldots \# x_n \# []$
$[m..<n]$	$\equiv$	$upt\ m\ n$
$[i..j]$	$\equiv$	$upto\ i\ j$
$[e.\ x \leftarrow xs]$	$\equiv$	$map\ (\lambda x.\ e)\ xs$
$[x{\leftarrow}xs\ .\ b]$	$\equiv$	$filter\ (\lambda x.\ b)\ xs$
$xs[n := x]$	$\equiv$	$list\text{-}update\ xs\ n\ x$
$\sum x{\leftarrow}xs.\ e$	$\equiv$	$listsum\ (map\ (\lambda x.\ e)\ xs)$

List comprehension: $[e.\ q_1,\ \ldots,\ q_n]$ where each qualifier q_i is either a generator $pat \leftarrow e$ or a guard, i.e. boolean expression.

22 Map

Maps model partial functions and are often used as finite tables. However, the domain of a map may be infinite.

$'a \rightharpoonup 'b\ =\ 'a \Rightarrow 'b\ option$

$Map.empty$	$:: 'a \rightharpoonup 'b$
$op\ ++$	$:: ('a \rightharpoonup 'b) \Rightarrow ('a \rightharpoonup 'b) \Rightarrow 'a \rightharpoonup 'b$
$op\ \circ_m$	$:: ('a \rightharpoonup 'b) \Rightarrow ('c \rightharpoonup 'a) \Rightarrow 'c \rightharpoonup 'b$
$op\ \lvert`$	$:: ('a \rightharpoonup 'b) \Rightarrow 'a\ set \Rightarrow 'a \rightharpoonup 'b$
dom	$:: ('a \rightharpoonup 'b) \Rightarrow 'a\ set$
ran	$:: ('a \rightharpoonup 'b) \Rightarrow 'b\ set$
$op\ \subseteq_m$	$:: ('a \rightharpoonup 'b) \Rightarrow ('a \rightharpoonup 'b) \Rightarrow bool$
$map\text{-}of$	$:: ('a \times 'b)\ list \Rightarrow 'a \rightharpoonup 'b$
$map\text{-}upds$	$:: ('a \rightharpoonup 'b) \Rightarrow 'a\ list \Rightarrow 'b\ list \Rightarrow 'a \rightharpoonup 'b$

Syntax

$Map.empty$	$\equiv$	$\lambda x.\ None$
$m(x \mapsto y)$	$\equiv$	$m(x{:=}Some\ y)$
$m(x_1{\mapsto}y_1,\ldots,x_n{\mapsto}y_n)$	$\equiv$	$m(x_1{\mapsto}y_1)\ldots(x_n{\mapsto}y_n)$
$[x_1{\mapsto}y_1,\ldots,x_n{\mapsto}y_n]$	$\equiv$	$Map.empty(x_1{\mapsto}y_1,\ldots,x_n{\mapsto}y_n)$
$m(xs\ [\mapsto]\ ys)$	$\equiv$	$map\text{-}upds\ m\ xs\ ys$