What's in Main

Tobias Nipkow

October 9, 2011

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	$:: 'a$	
$mono$	$:: ('a \Rightarrow 'b) \Rightarrow bool$	
$strict\text{-}mono$	$:: ('a \Rightarrow 'b) \Rightarrow bool$	

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 $>$

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

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

$\{x_1,\ldots,x_n\}$	$\equiv$	$insert\ x_1\ (\ldots\ (insert\ x_n\ \{\})\ldots)$	
$x \notin A$	$\equiv$	$\neg(x \in A)$	
$A \subseteq B$	$\equiv$	$A \le B$	
$A \subset B$	$\equiv$	$A < B$	
$A \supseteq B$	$\equiv$	$B \le A$	
$A \supset B$	$\equiv$	$B < A$	
$\{x.\ P\}$	$\equiv$	$Collect\ (\lambda x.\ P)$	
$\bigcup x{\in}I.\ A$	$\equiv$	$UNION\ I\ (\lambda x.\ A)$	(`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-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-betw	$::$	$('a \Rightarrow 'b) \Rightarrow 'a\ set \Rightarrow 'b\ set \Rightarrow bool$	
fun-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-into :: $'a$ *set* $\Rightarrow$ $('a \Rightarrow 'b) \Rightarrow 'b \Rightarrow 'a$

Syntax

inv $\equiv$ *inv-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

$$
\begin{array}{ll}
\mathit{converse} & :: ('a \times 'b)\ \mathit{set} \Rightarrow ('b \times 'a)\ \mathit{set} \\
\mathit{op}\ O & :: ('a \times 'b)\ \mathit{set} \Rightarrow ('b \times 'c)\ \mathit{set} \Rightarrow ('a \times 'c)\ \mathit{set} \\
\mathit{op}\ `` & :: ('a \times 'b)\ \mathit{set} \Rightarrow 'a\ \mathit{set} \Rightarrow 'b\ \mathit{set} \\
\mathit{inv\text{-}image} & :: ('a \times 'a)\ \mathit{set} \Rightarrow ('b \Rightarrow 'a) \Rightarrow ('b \times 'b)\ \mathit{set} \\
\mathit{Id\text{-}on} & :: 'a\ \mathit{set} \Rightarrow ('a \times 'a)\ \mathit{set} \\
\mathit{Id} & :: ('a \times 'a)\ \mathit{set} \\
\mathit{Domain} & :: ('a \times 'b)\ \mathit{set} \Rightarrow 'a\ \mathit{set} \\
\mathit{Range} & :: ('a \times 'b)\ \mathit{set} \Rightarrow 'b\ \mathit{set} \\
\mathit{Field} & :: ('a \times 'a)\ \mathit{set} \Rightarrow 'a\ \mathit{set} \\
\mathit{refl\text{-}on} & :: 'a\ \mathit{set} \Rightarrow ('a \times 'a)\ \mathit{set} \Rightarrow \mathit{bool} \\
\mathit{refl} & :: ('a \times 'a)\ \mathit{set} \Rightarrow \mathit{bool} \\
\mathit{sym} & :: ('a \times 'a)\ \mathit{set} \Rightarrow \mathit{bool} \\
\mathit{antisym} & :: ('a \times 'a)\ \mathit{set} \Rightarrow \mathit{bool} \\
\mathit{trans} & :: ('a \times 'a)\ \mathit{set} \Rightarrow \mathit{bool} \\
\mathit{irrefl} & :: ('a \times 'a)\ \mathit{set} \Rightarrow \mathit{bool} \\
\mathit{total\text{-}on} & :: 'a\ \mathit{set} \Rightarrow ('a \times 'a)\ \mathit{set} \Rightarrow \mathit{bool} \\
\mathit{total} & :: ('a \times 'a)\ \mathit{set} \Rightarrow \mathit{bool}
\end{array}
$$

Syntax

$$r^{-1} \quad \equiv \quad \mathit{converse}\ r \quad (\texttt{\^{}-1})$$

11 Equiv_Relations

$$
\begin{array}{ll}
\mathit{equiv} & :: 'a\ \mathit{set} \Rightarrow ('a \times 'a)\ \mathit{set} \Rightarrow \mathit{bool} \\
\mathit{op}\ // & :: 'a\ \mathit{set} \Rightarrow ('a \times 'a)\ \mathit{set} \Rightarrow 'a\ \mathit{set}\ \mathit{set} \\
\mathit{congruent} & :: ('a \times 'a)\ \mathit{set} \Rightarrow ('a \Rightarrow 'b) \Rightarrow \mathit{bool} \\
\mathit{congruent2} & :: ('a \times 'a)\ \mathit{set} \Rightarrow ('b \times 'b)\ \mathit{set} \Rightarrow ('a \Rightarrow 'b \Rightarrow 'c) \Rightarrow \mathit{bool}
\end{array}
$$

Syntax

$$
\begin{array}{lcl}
f\ \mathit{respects}\ r & \equiv & \mathit{congruent}\ r\ f \\
f\ \mathit{respects2}\ r & \equiv & \mathit{congruent2}\ r\ r\ f
\end{array}
$$

12 Transitive_Closure

$$
\begin{array}{ll}
\mathit{rtrancl} & :: ('a \times 'a)\ \mathit{set} \Rightarrow ('a \times 'a)\ \mathit{set} \\
\mathit{trancl} & :: ('a \times 'a)\ \mathit{set} \Rightarrow ('a \times 'a)\ \mathit{set} \\
\mathit{reflcl} & :: ('a \times 'a)\ \mathit{set} \Rightarrow ('a \times 'a)\ \mathit{set} \\
\mathit{op}\ \hat{}\hat{} & :: ('a \times 'a)\ \mathit{set} \Rightarrow \mathit{nat} \Rightarrow ('a \times 'a)\ \mathit{set}
\end{array}
$$

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 $nat = 0 \mid Suc\ nat$

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

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

15 Int

Type *int*

$op\ +$	$op\ -$	$uminus$	$op\ *$	$op\ \hat{}$	$op\ div$	$op\ mod$	$op\ dvd$
$op\ \leq$	$op\ <$	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	$:: 'a\ option \Rightarrow 'a\ set$
Option.bind	$:: 'a\ option \Rightarrow ('a \Rightarrow 'b\ option) \Rightarrow 'b\ option$

21 List

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

op @	$:: {}'a\ list \Rightarrow {}'a\ list \Rightarrow {}'a\ list$
butlast	$:: {}'a\ list \Rightarrow {}'a\ list$
concat	$:: {}'a\ list\ list \Rightarrow {}'a\ list$
distinct	$:: {}'a\ list \Rightarrow bool$
drop	$:: nat \Rightarrow {}'a\ list \Rightarrow {}'a\ list$
dropWhile	$:: ({}'a \Rightarrow bool) \Rightarrow {}'a\ list \Rightarrow {}'a\ list$
filter	$:: ({}'a \Rightarrow bool) \Rightarrow {}'a\ list \Rightarrow {}'a\ list$
foldl	$:: ({}'a \Rightarrow {}'b \Rightarrow {}'a) \Rightarrow {}'a \Rightarrow {}'b\ list \Rightarrow {}'a$
foldr	$:: ({}'a \Rightarrow {}'b \Rightarrow {}'b) \Rightarrow {}'a\ list \Rightarrow {}'b \Rightarrow {}'b$
hd	$:: {}'a\ list \Rightarrow {}'a$
last	$:: {}'a\ list \Rightarrow {}'a$
length	$:: {}'a\ list \Rightarrow nat$
lenlex	$:: ({}'a \times {}'a)\ set \Rightarrow ({}'a\ list \times {}'a\ list)\ set$
lex	$:: ({}'a \times {}'a)\ set \Rightarrow ({}'a\ list \times {}'a\ list)\ set$
lexn	$:: ({}'a \times {}'a)\ set \Rightarrow nat \Rightarrow ({}'a\ list \times {}'a\ list)\ set$
lexord	$:: ({}'a \times {}'a)\ set \Rightarrow ({}'a\ list \times {}'a\ list)\ set$
listrel	$:: ({}'a \times {}'a)\ set \Rightarrow ({}'a\ list \times {}'a\ list)\ set$
listrel1	$:: ({}'a \times {}'a)\ set \Rightarrow ({}'a\ list \times {}'a\ list)\ set$
lists	$:: {}'a\ set \Rightarrow {}'a\ list\ set$
listset	$:: {}'a\ set\ list \Rightarrow {}'a\ list\ set$
listsum	$:: {}'a\ list \Rightarrow {}'a$
list-all2	$:: ({}'a \Rightarrow {}'b \Rightarrow bool) \Rightarrow {}'a\ list \Rightarrow {}'b\ list \Rightarrow bool$
list-update	$:: {}'a\ list \Rightarrow nat \Rightarrow {}'a \Rightarrow {}'a\ list$
map	$:: ({}'a \Rightarrow {}'b) \Rightarrow {}'a\ list \Rightarrow {}'b\ list$
measures	$:: ({}'a \Rightarrow nat)\ list \Rightarrow ({}'a \times {}'a)\ set$
op !	$:: {}'a\ list \Rightarrow nat \Rightarrow {}'a$
remdups	$:: {}'a\ list \Rightarrow {}'a\ list$
removeAll	$:: {}'a \Rightarrow {}'a\ list \Rightarrow {}'a\ list$
remove1	$:: {}'a \Rightarrow {}'a\ list \Rightarrow {}'a\ list$
replicate	$:: nat \Rightarrow {}'a \Rightarrow {}'a\ list$
rev	$:: {}'a\ list \Rightarrow {}'a\ list$
rotate	$:: nat \Rightarrow {}'a\ list \Rightarrow {}'a\ list$
rotate1	$:: {}'a\ list \Rightarrow {}'a\ list$
set	$:: {}'a\ list \Rightarrow {}'a\ set$
sort	$:: {}'a\ list \Rightarrow {}'a\ list$
sorted	$:: {}'a\ list \Rightarrow bool$
splice	$:: {}'a\ list \Rightarrow {}'a\ list \Rightarrow {}'a\ list$
sublist	$:: {}'a\ list \Rightarrow (nat \Rightarrow bool) \Rightarrow {}'a\ list$
take	$:: nat \Rightarrow {}'a\ list \Rightarrow {}'a\ list$
takeWhile	$:: ({}'a \Rightarrow bool) \Rightarrow {}'a\ list \Rightarrow {}'a\ list$
tl	$:: {}'a\ list \Rightarrow {}'a\ list$

upt :: $nat \Rightarrow nat \Rightarrow nat\ list$
$upto$ :: $int \Rightarrow int \Rightarrow int\ list$
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.
$'a \rightharpoonup 'b\ =\ 'a \Rightarrow 'b\ option$

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

Syntax

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