What's in Main

Tobias Nipkow

December 12, 2016

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. For infix operators and their precedences see the final section. The sophisticated class structure is only hinted at. For details see http://isabelle.in.tum.de/library/HOL.

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

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

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

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*.

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

Set

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

$\{a_1,\ldots,a_n\}$	$\equiv$	$insert\ a_1\ (\ldots\ (insert\ a_n\ \{\})\ldots)$	
$a \notin A$	$\equiv$	$\neg(x \in A)$	
$A \subseteq B$	$\equiv$	$A \leq B$	
$A \subset B$	$\equiv$	$A < B$	
$A \supseteq B$	$\equiv$	$B \leq A$	
$A \supset B$	$\equiv$	$B < A$	
$\{x.\ P\}$	$\equiv$	$Collect\ (\lambda x.\ P)$	
$\{t \mid x_1\ \ldots\ x_n.\ P\}$	$\equiv$	$\{v.\ \exists x_1\ \ldots\ x_n.\ v = t \wedge P\}$	
$\bigcup x{\in}I.\ A$	$\equiv$	$UNION\ I\ (\lambda x.\ A)$	(UN)
$\bigcup x.\ A$	$\equiv$	$UNION\ UNIV\ (\lambda x.\ A)$	
$\bigcap x{\in}I.\ A$	$\equiv$	$INTER\ I\ (\lambda x.\ A)$	(INT)
$\bigcap x.\ A$	$\equiv$	$INTER\ UNIV\ (\lambda x.\ A)$	
$\forall x{\in}A.\ P$	$\equiv$	$Ball\ A\ (\lambda x.\ P)$	
$\exists x{\in}A.\ P$	$\equiv$	$Bex\ A\ (\lambda x.\ P)$	
$range\ f$	$\equiv$	$f\ `\ UNIV$	

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

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

Hilbert_Choice

Hilbert's selection (ε) operator: $\mathit{SOME}\ x.\ P$.

$\mathit{inv\_into} :: 'a\ \mathit{set} \Rightarrow ('a \Rightarrow 'b) \Rightarrow 'b \Rightarrow 'a$

Syntax

$\mathit{inv} \quad \equiv \quad \mathit{inv\_into}\ \mathit{UNIV}$

Fixed Points

Theory: *Inductive*.

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

Sum_Type

Type constructor $+$.

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

Product_Type

Types *unit* and $\times$.

$()$	$:: \mathit{unit}$
Pair	$:: 'a \Rightarrow 'b \Rightarrow 'a \times 'b$
fst	$:: 'a \times 'b \Rightarrow 'a$
snd	$:: 'a \times 'b \Rightarrow 'b$
$\mathit{case\_prod}$	$:: ('a \Rightarrow 'b \Rightarrow 'c) \Rightarrow 'a \times 'b \Rightarrow 'c$
curry	$:: ('a \times 'b \Rightarrow 'c) \Rightarrow 'a \Rightarrow 'b \Rightarrow 'c$
Sigma	$:: 'a\ \mathit{set} \Rightarrow ('a \Rightarrow 'b\ \mathit{set}) \Rightarrow ('a \times 'b)\ \mathit{set}$

Syntax

$(a,\ b)$	$\equiv$	$\mathit{Pair}\ a\ b$
$\lambda(x,\ y).\ t$	$\equiv$	$\mathit{case\_prod}\ (\lambda x\ y.\ t)$
$A \times B$	$\equiv$	$\mathit{Sigma}\ A\ (\lambda\_.\ B)$

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.

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

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

Type synonym $'a\ rel = ('a \times 'a)\ set$

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*

Transitive_Closure

$$\begin{array}{lll}
rtrancl & :: & ('a \times 'a)\ set \Rightarrow ('a \times 'a)\ set \\
trancl & :: & ('a \times 'a)\ set \Rightarrow ('a \times 'a)\ set \\
reflcl & :: & ('a \times 'a)\ set \Rightarrow ('a \times 'a)\ set \\
acyclic & :: & ('a \times 'a)\ set \Rightarrow bool \\
op\ \widehat{\ }\widehat{\ } & :: & ('a \times 'a)\ set \Rightarrow nat \Rightarrow ('a \times 'a)\ set
\end{array}$$

Syntax

$$\begin{array}{llll}
r^* & \equiv & rtrancl\ r & (\texttt{\^{}*}) \\
r^+ & \equiv & trancl\ r & (\texttt{\^{}+}) \\
r^= & \equiv & reflcl\ r & (\texttt{\^{}=})
\end{array}$$

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:

$$\begin{array}{llll}
0 & :: & 'a & \\
1 & :: & 'a & \\
op + & :: & 'a \Rightarrow 'a \Rightarrow 'a & \\
op - & :: & 'a \Rightarrow 'a \Rightarrow 'a & \\
uminus & :: & 'a \Rightarrow 'a & (\texttt{-}) \\
op * & :: & 'a \Rightarrow 'a \Rightarrow 'a & \\
inverse & :: & 'a \Rightarrow 'a & \\
op\ div & :: & 'a \Rightarrow 'a \Rightarrow 'a & \\
abs & :: & 'a \Rightarrow 'a & \\
sgn & :: & 'a \Rightarrow 'a & \\
op\ dvd & :: & 'a \Rightarrow 'a \Rightarrow bool & \\
op\ div & :: & 'a \Rightarrow 'a \Rightarrow 'a & \\
op\ mod & :: & 'a \Rightarrow 'a \Rightarrow 'a &
\end{array}$$

Syntax

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

Nat

datatype $nat = 0 \mid Suc\ nat$

$$\begin{array}{lllllll}
op + & op - & op * & op\ \widehat{\ } & op\ div & op\ mod & op\ dvd \\
op \leq & op < & min & max & Min & Max &
\end{array}$$

$$\begin{array}{lll}
of_nat & :: & nat \Rightarrow 'a \\
op\ \widehat{\ }\widehat{\ } & :: & ('a \Rightarrow 'a) \Rightarrow nat \Rightarrow 'a \Rightarrow 'a
\end{array}$$

Int

Type *int*

$op\ +$	$op\ -$	$uminus$	$op\ *$	$op\ \widehat{}$	$op\ div$	$op\ mod$	$op\ dvd$
$op\ \leq$	$op\ <$	min	max	Min	Max		
abs	sgn						

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

Syntax

$int \quad \equiv \quad of\_nat$

Finite_Set

$finite$	$:: 'a\ set \Rightarrow bool$
$card$	$:: 'a\ set \Rightarrow nat$
$Finite\_Set.fold$	$:: ('a \Rightarrow 'b \Rightarrow 'b) \Rightarrow 'b \Rightarrow 'a\ set \Rightarrow 'b$

Groups_Big

sum	$:: ('a \Rightarrow 'b) \Rightarrow 'a\ set \Rightarrow 'b$
$prod$	$:: ('a \Rightarrow 'b) \Rightarrow 'a\ set \Rightarrow 'b$

Syntax

$\sum A$	$\equiv$	$sum\ (\lambda x.\ x)\ A$	(`SUM`)
$\sum x \in A.\ t$	$\equiv$	$sum\ (\lambda x.\ t)\ A$	
$\sum x \vert P.\ t$	$\equiv$	$\sum x\ \vert\ P.\ t$	
Similarly for $\prod$ instead of $\sum$			(`PROD`)

Wellfounded

wf	$:: ('a \times 'a)\ set \Rightarrow bool$
$Wellfounded.acc$	$:: ('a \times 'a)\ set \Rightarrow 'a\ set$
$measure$	$:: ('a \Rightarrow nat) \Rightarrow ('a \times 'a)\ set$
$op\ <*lex*>$	$:: ('a \times 'a)\ set \Rightarrow ('b \times 'b)\ set \Rightarrow (('a \times 'b) \times 'a \times 'b)\ set$
$op\ <*mlex*>$	$:: ('a \Rightarrow nat) \Rightarrow ('a \times 'a)\ set \Rightarrow ('a \times 'a)\ set$
$less\_than$	$:: (nat \times nat)\ set$
$pred\_nat$	$:: (nat \times nat)\ set$

Set_Interval

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

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 $\bigcup$		
$\sum x = a..b.\ t$	$\equiv$	$sum\ (\lambda x.\ t)\ \{a..b\}$
$\sum x = a..<b.\ t$	$\equiv$	$sum\ (\lambda x.\ t)\ \{a..<b\}$
$\sum x \le b.\ t$	$\equiv$	$sum\ (\lambda x.\ t)\ \{..b\}$
$\sum x < b.\ t$	$\equiv$	$sum\ (\lambda x.\ t)\ \{..<b\}$
Similarly for $\prod$ instead of $\sum$		

Power

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

Option

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

the	$::$	$'a\ option \Rightarrow 'a$
map_option	$::$	$('a \Rightarrow 'b) \Rightarrow 'a\ option \Rightarrow 'b\ option$
set_option	$::$	$'a\ option \Rightarrow 'a\ set$
$Option.bind$	$::$	$'a\ option \Rightarrow ('a \Rightarrow 'b\ option) \Rightarrow 'b\ option$

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	$:: ('a \Rightarrow bool) \Rightarrow 'a\ list \Rightarrow 'a\ list$
find	$:: ('a \Rightarrow bool) \Rightarrow 'a\ list \Rightarrow 'a\ option$
fold	$:: ('a \Rightarrow 'b \Rightarrow 'b) \Rightarrow 'a\ list \Rightarrow 'b \Rightarrow 'b$
foldr	$:: ('a \Rightarrow 'b \Rightarrow 'b) \Rightarrow 'a\ list \Rightarrow 'b \Rightarrow 'b$
foldl	$:: ('a \Rightarrow 'b \Rightarrow 'a) \Rightarrow 'a \Rightarrow 'b\ list \Rightarrow 'a$
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	$:: ('a \times 'b)\ set \Rightarrow ('a\ list \times 'b\ list)\ set$
listrel1	$:: ('a \times 'a)\ set \Rightarrow ('a\ list \times 'a\ list)\ set$
lists	$:: 'a\ set \Rightarrow 'a\ list\ set$
listset	$:: 'a\ set\ list \Rightarrow 'a\ list\ set$
sum_list	$:: 'a\ list \Rightarrow 'a$
prod_list	$:: 'a\ list \Rightarrow 'a$
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	$:: 'a\ list \Rightarrow 'a\ list$

$sorted$	::	$'a\ list \Rightarrow bool$
$splice$	::	$'a\ list \Rightarrow 'a\ list \Rightarrow 'a\ list$
$sublist$	::	$'a\ list \Rightarrow nat\ set \Rightarrow 'a\ list$
$take$	::	$nat \Rightarrow 'a\ list \Rightarrow 'a\ list$
$takeWhile$	::	$('a \Rightarrow bool) \Rightarrow 'a\ list \Rightarrow 'a\ list$
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

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

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.

Map

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

$Map.empty$	::	$'a \Rightarrow 'b\ option$
$op\ ++$	::	$('a \Rightarrow 'b\ option) \Rightarrow ('a \Rightarrow 'b\ option) \Rightarrow 'a \Rightarrow 'b\ option$
$op\ \circ_m$	::	$('a \Rightarrow 'b\ option) \Rightarrow ('c \Rightarrow 'a\ option) \Rightarrow 'c \Rightarrow 'b\ option$
$op\ \|`$	::	$('a \Rightarrow 'b\ option) \Rightarrow 'a\ set \Rightarrow 'a \Rightarrow 'b\ option$
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_of	::	$('a \times 'b)\ list \Rightarrow 'a \Rightarrow 'b\ option$
map_upds	::	$('a \Rightarrow 'b\ option) \Rightarrow 'a\ list \Rightarrow 'b\ list \Rightarrow 'a \Rightarrow 'b\ option$

Syntax

$Map.empty$	$\equiv$	$Map.empty$
$m(x \mapsto y)$	$\equiv$	$m(x{:=}Some\ y)$
$m(x_1{\mapsto}y_1,\ldots,x_n{\mapsto}y_n)$	$\equiv$	$m(x_1{\mapsto}y_1)\ldots(x_n{\mapsto}y_n)$
$[x_1{\mapsto}y_1,\ldots,x_n{\mapsto}y_n]$	$\equiv$	$Map.empty(x_1{\mapsto}y_1,\ldots,x_n{\mapsto}y_n)$
$m(xs\ [\mapsto]\ ys)$	$\equiv$	$map\_upds\ m\ xs\ ys$

Infix operators in Main

	Operator	precedence	associativity
Meta-logic	$\Longrightarrow$	1	right
	$\equiv$	2	
Logic	$\wedge$	35	right
	$\vee$	30	right
	$\longrightarrow$, $\longleftrightarrow$	25	right
	$=$, $\neq$	50	left
Orderings	$\leq$, $<$, $\geq$, $>$	50	
Sets	$\subseteq$, $\subset$, $\supseteq$, $\supset$	50	
	$\in$, $\notin$	50	
	$\cap$	70	left
	$\cup$	65	left
Functions and Relations	$\circ$	55	left
	$`$	90	right
	O	75	right
	$``$	90	right
	$\frown\frown$	80	right
Numbers	$+$, $-$	65	left
	$*$, $/$	70	left
	div, mod	70	left
	$\hat{}$	80	right
	dvd	50	
Lists	$\#$, @	65	right
	!	100	left