Everything on Logic

Verbose	Shorthand	Meaning
not	!	negation
and	&&	conjunction
or	||	disjunction
implies	=>	implication
else	,	alternative
iff	<=>	bi-implication

Implication

P => Q
P implies Q

p	q	p implies q
true	true	true
true	false	false
false	true	true
false	false	true

Else

The else operator is used with the implication operator
F implies G else H
(F and G) or ((not F) and H)

Bi-implication

The <=> is two-way implication.
P iff Q or P <=> Q → (P implies Q) and (Q implies P)
Either both are true or both false

Nested Implication

C1 => F1,
C2 => F2,
C3 => F3

OR

C1 implies F1
else C2 implies F2
else C3 implies F3

Some Shorthands

{F G H} is equivalent to F and G and H
a != b equivalent to not (a=b), or, a not= b

Equivalent Logical Expressions

p **and** q	q **and** p
p **and** p	p
not not p	p
not (p **and** q)	(**not** p) **or** (**not** q)
not (p **or** q)	(**not** p) **and** (**not** q)
p **implies** q	(**not** p) **or** q
p **implies** q	(**not** q) **implies** (**not** p)

Quantified Expressions

Q x:e | P where q is a quantifier

all x:e | P	P holds for every x in e
some x:e | P	P holds for at least one x in e
no x:e | P	P holds for no x in e
lone x:e | P	P holds for at most one x in e
one x:e | P	P holds for exactly one x in e

Several variables can be bound to the same quantifier; one x:e1, y:e2 | P

Variables with same type can share declaration; all x,y:e | P

Let Expressions

When an expression appears repeatedly, or is a subexpression of a large complicated expression you can factor it out via a let expression

Let x:e | A

Means the same as the expression A, with each occurence of x replaced by the expression e