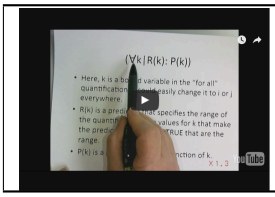
1.4.2 The "for all" quantifier ☛ to edX



☛ Watch Video on edX
☛ Watch Video on YouTube

To avoid having to explicitly list every term in the conjunction, one can instead use the "for all" quantifier:

$$P_0 \wedge P_1 \wedge \cdots \wedge P_{n-1} \quad \text{can be written as (is equivalent to)} \quad (\forall k \mid 0 \le k < n : P_k).$$

This is read as "for all k in the range $0 \le k < n$ the expression P_k holds" or, equivalently, "P_k holds for all k in the range $0 \le k < n$."

More generally, we may have a predicate that is a function of the integer variable k, $P(k)$, and a set of values of k for which this expression evaluates to TRUE. This would be expressed as

$$(\forall k \mid R(k) : P(k)).$$

Here

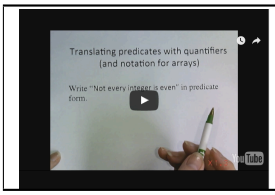
- k is the *bound variable* in the "for all" quantification.
- $R(k)$ is a predicate that is a function of k and specifies the *range* of the quantification. It is the values of k for which $R(k)$ evaluates to TRUE that are in the range.
- $P(k)$ is the predicate that is a function of k.

If $S = \{k_0, k_1, k_2, \ldots\}$ is the set of all integers that satisfies $R(k)$, then

$$(\forall k \mid k \in S : P(k)) = (\forall k \mid R(k) : P(k)) = (P(k_0) \wedge P(k_1) \wedge \cdots).$$

In this course, the set denoted by $R(k)$ can be finite (e.g., $0 \le k < 10$) or countably infinite (e.g., $0 \le k$).

1.4.3 The "there exists" quantifier ☛ to edX



☛ Watch Video on edX
☛ Watch Video on YouTube

To avoid having to explicitly list every term in the disjunction, one can instead use the "there exists" quantifier:

$$P_0 \vee P_1 \vee \cdots \vee P_{n-1} \quad \text{can be written as (is equivalent to)} \quad (\exists k \mid 0 \le k < n : P_k).$$

This is read as "there exists a k in the range $0 \le k < n$ such that the expression P_k holds" or, equivalently, "P_k holds for at least one k in the range $0 \le k < n$."

More generally, in

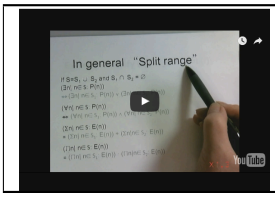
$$(\exists k \mid R(k) : P(k)).$$

- k is the *bound variable* in the "there exists" quantification.
- $R(k)$ is a predicate that is a function of k and specifies the *range* of the quantification. It is the values of k for which $R(k)$ evaluates to TRUE that are in the range.
- $P(k)$ is the predicate that is a function of k.

If $S = \{k_0, k_1, k_2, \ldots\}$ is the set of all integers that satisfy $R(k)$, then

$$(\exists k \mid k \in S : P(k)) = (\exists k \mid R(k) : P(k)) = (P(k_0) \vee P(k_1) \vee \cdots).$$

1.4.4 Splitting the range ☛ to edX



☛ Watch Video on edX
☛ Watch Video on YouTube

Often in our discussions we will want to "split the range" of a quantifier.

Example 1.4 Consider, an array b with elements $b(0)$ through $b(n-1)$. The predicate

$$s = (\sum k \mid 0 \le k < n : b(k))$$

expresses that variable s equals the sum of the elements of b. Given an integer i that satisfies $0 \le i < n$, the given predicate is equivalent to

$$s = (\sum k \mid 0 \le k < i : b(k)) + (\sum k \mid i \le k < n : b(k)).$$

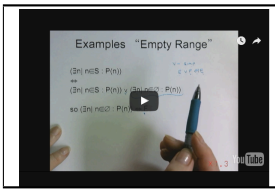
because

$$\begin{aligned}(\sum k \mid 0 \le k < n : b(k)) = \quad & \underbrace{b(0) + \cdots + b(i-1)}_{(\sum k \mid 0 \le k < i : b(k))} \quad + \quad \underbrace{b(i) + \cdots + b(n-1)}_{(\sum k \mid i \le k < n : b(k))} \\ & = (\sum k \mid 0 \le k < i : b(k)) + (\sum k \mid i \le k < n : b(k))\end{aligned}$$

Let us discuss the theory that underlies this. Partition S into the subsets S_0 and S_1. Partitioning means that $S_0 \cup S_1 = S$ and $S_0 \cap S_1 = \varnothing$. Then

$$\begin{aligned}(\forall k \mid k \in S : E(k)) \quad &\Leftrightarrow \quad (\forall k \mid k \in S_0 : E(k)) \ \wedge \ (\forall k \mid k \in S_1 : E(k)) \\ (\exists k \mid k \in S : E(k)) \quad &\Leftrightarrow \quad (\exists k \mid k \in S_0 : E(k)) \ \vee \ (\exists k \mid k \in S_1 : E(k)) \\ (\sum k \mid k \in S : E(k)) \quad &= \quad (\sum k \mid k \in S_0 : E(k)) \ + \ (\sum k \mid k \in S_1 : E(k)) \\ (\prod k \mid k \in S : E(k)) \quad &= \quad (\prod k \mid k \in S_0 : E(k)) \ \times \ (\prod k \mid k \in S_1 : E(k)).\end{aligned}$$

1.4.5 Quantifiers with special ranges ☛ to edX



☛ Watch Video on edX
☛ Watch Video on YouTube

In our proofs of correctness, we will often want to split one component or term from a quantifier.

Example 1.5 Splitting one term from a "for all" quantifier:

$$\begin{aligned} & 1 \le k \wedge (\forall i \mid 1 \le i \le k : a(i) = b(i) + c(i)) \\ = \quad & 1 \le k \wedge (\forall i \mid 1 \le i < k : a(i) = b(i) + c(i)) \wedge \underbrace{(\forall i \mid k \le i \le k : a(i) = b(i) + c(i))}_{a(k) = b(k) + c(k)} \\ = \quad & 1 \le k \wedge (\forall i \mid 1 \le i < k : a(i) = b(i) + c(i)) \wedge a(k) = b(k) + c(k) \end{aligned}$$

Splitting one term from a "there exists" quantifier:

$$\begin{aligned} & 1 \le k \wedge (\exists i \mid 1 \le i \le k : a(i) = b(i) + c(i)) \\ = \quad & 1 \le k \wedge (\exists i \mid 1 \le i < k : a(i) = b(i) + c(i)) \vee \underbrace{(\exists i \mid k \le i \le k : a(i) = b(i) + c(i))}_{a(k) = b(k) + c(k)} \\ = \quad & 1 \le k \wedge (\exists i \mid 1 \le i < k : a(i) = b(i) + c(i)) \vee a(k) = b(k) + c(k) \end{aligned}$$

Splitting one term from a summation:

$$\begin{aligned} & 1 \le k \wedge (\textstyle\sum i \mid 1 \le i \le k : a(i)) \\ = \quad & 1 \le k \wedge (\textstyle\sum i \mid 1 \le i < k : a(i)) + \underbrace{(\textstyle\sum i \mid k \le i \le k : a(i))}_{a(k)} \\ = \quad & 1 \le k \wedge (\textstyle\sum i \mid 1 \le i < k : a(i)) + a(k) \end{aligned}$$

What these examples illustrate is that when the range of a quantifier consists of a single point it simply evaluates to the expression (term) evaluated at that point.

In the last unit you may also have wondered what would happen if one of the two sets S_0 or S_1 is the empty set. Let us motivate this by revisiting Example 1.4:

Example 1.6 Consider, an array b with elements $b(0)$ through $b(n-1)$. The predicate

$$s = (\sum k \mid 0 \leq k < n : b(k))$$

expresses that variable s equals the sum of the elements of b. Given an integer i that satisfies $0 \leq i < n$, the given predicate is equivalent to

$$s = (\sum k \mid 0 \leq k < i : b(k)) + (\sum k \mid i \leq k < n : b(k)).$$

Let us examine the extreme cases:

$i = 0$**:** In this case, we find that

$$s = (\sum k \mid \underbrace{0 \leq k < 0}_{\text{empty!}} : b(k)) + (\sum k \mid 0 \leq k < n : b(k))$$

We notice that in this example, the sum over the empty range better be defined to equal zero!

$i = n$**:** In this case, we find that

$$s = (\sum k \mid 0 \leq k < n : b(k)) + (\sum k \mid \underbrace{n \leq k < n}_{\text{empty!}} : b(k))$$

We notice that, again, the sum over the empty range better be defined to equal zero!

Let us look at this more generally. Split $S = S_0 \cup S_1$ where $S_0 \cap S_1 = \varnothing$. Consider

$$(\sum k \mid k \in S : E(k)) = (\sum k \mid k \in S_0 : E(k)) + (\sum k \mid k \in S_1 : E(k)).$$

If $S_0 = \varnothing$, then $S_1 = S$ and

$$(\sum k \mid k \in S : E(k)) = (\sum k \mid k \in \varnothing : E(k)) + (\sum k \mid k \in S : E(k))$$

implies that

$$(\sum k \mid k \in \varnothing : E(k)) = 0.$$

Via similar reasoning one concludes that

$$\begin{array}{rcl} (\forall k \mid k \in \varnothing : P(k)) & \Leftrightarrow & T \\ (\exists k \mid k \in \varnothing : P(k)) & \Leftrightarrow & F \\ (\sum k \mid k \in \varnothing : E(k)) & = & 0 \\ (\prod k \mid k \in \varnothing : E(k)) & = & 1. \end{array}$$

1.4.6 Practice expressing statements as predicates ☛ to edX

Homework 1.4.6.1 Let us consider a one dimensional array $b(1:n)$ (using Matlab notation), where $1 \leq n$. Let j and k be two integer variables satisfying $1 \leq j \leq k \leq n$. By $b(j:k)$ we mean the subarray of b consisting of $b(j), b(j+1), \ldots b(k)$. The segment $b(j:k)$ is empty (contains no elements) if $j > k$.
Translate the following sentences into predicates.

1. All elements in the subarray $b(j:k)$ are positive.
2. No element in the subarray $b(j:k)$ is positive.
3. It is not the case that all elements in the subarray $b(j:k)$ are positive.
4. All elements in the subarray $b(j:k)$ are not positive.
5. Some element in the subarray $b(j:k)$ is positive.
6. There is an element in the subarray $b(j:k)$ that is positive.
7. At least one element in the subarray $b(j:k)$ is positive.
8. Some element in the subarray $b(j:k)$ is not positive.
9. Not all elements in the subarray $b(j:k)$ are positive.
10. It is not the case that there is an element in the subarray $b(j:k)$ that is positive.

☛ SEE ANSWER
☛ DO EXERCISE ON edX

Homework 1.4.6.2 Translate the following sentence into a predicate: Exactly one element in the subarray $b(j:k)$ is positive.

1. $(\exists i \mid j \leq i \leq k : b(i) > 0 \wedge (\forall p \mid j \leq p \leq k \wedge p \neq i : \neg(b(p) > 0)))$
2. $(\exists i \mid j \leq i \leq k : b(i) > 0) \wedge (\forall p \mid j \leq p \leq k \wedge p \neq i : \neg(b(p) > 0))$

☛ SEE ANSWER
☛ DO EXERCISE ON edX

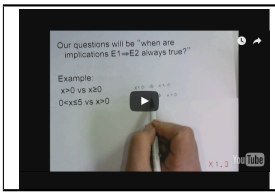
Homework 1.4.6.3 Formalize the following English specifications. Be sure to introduce necessary restrictions.

1. Set s equal to the sum of the elements of $b(j:k)$.
2. Set M equal to the maximum value in $b(j:k)$.
3. Set I equal to the index of a maximum value of $b(j:k)$.
4. Calculate x, the greatest power of 2 that is not greater than positive integer n.
5. Compute c, the number of zeroes in array $b(1:n)$.
6. Consider array of integers $b(1:n)$. Each of its subsegments $b(i:j)$ has a sum $S_{i,j} = (\sum \mid i \leq k \leq j : b(k))$. Compute M equal to the maximum such sum.

☛ SEE ANSWER
☛ DO EXERCISE ON edX

1.5 Weakening/strengthening

1.5.1 Weakening/strengthening laws ☛ to edX



☛ Watch Video on edX
☛ Watch Video on YouTube

When proving correctness of programs, the notion of one predicate being "stronger" or "weaker" than another predicate will play a central role. This will become clear starting in Week 2. In this unit, you will learn what it means for a predicate to be stronger or weaker and you will be equipped with some laws we call "Weakening/strenghtening" laws.

When two predicates E1 and E2 have the property that E1 $\Rightarrow$ E2, the predicate E2 is said to be *weaker* (less restrictive) than predicate E1. Equivalently, E1 is said to be *stronger* (more restrictive) than E2. Notice that this means that any predicate is simultaneously weaker and stronger than itself.

How do we most systematically show that $x \geq 3$ is weaker than $x = 5$ using what we have learned before?

$$
\begin{array}{l}
\quad (x = 5) \Rightarrow (x \geq 3) \\
\Leftrightarrow < \text{algebra} > \\
\quad (x = 5) \Rightarrow (x \geq 6) \vee (x = 5) \vee (x = 4) \vee (x = 3)
\end{array}
$$

Now, at this point you may look at

$$(x = 5) \Rightarrow (x \geq 6) \vee (x = 5) \vee (x = 4) \vee (x = 3)$$

and say "Well, dah! Obviously this is TRUE". However, to show it rigorously, you have to continue the proof:

$$
\begin{array}{l}
\quad (x = 5) \Rightarrow (x \geq 6) \vee (x = 5) \vee (x = 4) \vee (x = 3) \\
\Leftrightarrow < \text{implication; commutivity} > \\
\quad \neg(x = 5) \vee ((x = 5) \vee (x \geq 6) \vee (x = 4) \vee (x = 3)) \quad \Leftrightarrow < \text{associativity; excluded middle; associativity} > \\
\quad T \vee ((x \geq 6) \vee (x = 4) \vee (x = 3)) \\
\Leftrightarrow < \vee\text{-simplification} > \\
\quad T
\end{array}
$$

At the end of the next unit, we will equip you with a few new laws of logic that allow you to essentially say "Well, dah" instead.

1.5.2 Weakening/strengthening exercises ☛ to edX

Homework 1.5.2.1 For each of the following, if applicable, indicate which statement is TRUE (by examination):

1. (a) $0 \leq x \leq 10$ is weaker than $1 \leq x < 5$.
   (b) $0 \leq x \leq 10$ is stronger than $1 \leq x < 5$.
2. (a) $x = 5 \wedge y = 4$ is weaker than $y = 4$.
   (b) $x = 5 \wedge y = 4$ is stronger than $y = 4$.
3. (a) $x \leq 5 \vee y = 3$ is weaker than $x = 5 \wedge y = 4$.
   (b) $x \leq 5 \vee y = 3$ is stronger than $x = 5 \wedge y = 4$.
4. (a) T is weaker than F.
   (b) T is stronger than F.
5. (a) $(\forall i | 5 \leq i \leq 10 : b(i+1) < b(i))$ is weaker than $(\forall i | 7 \leq i \leq 10 : b(i+1) < b(i))$.
   (b) $(\forall i | 5 \leq i \leq 10 : b(i+1) < b(i))$ is stronger than $(\forall i | 7 \leq i \leq 10 : b(i+1) < b(i))$.
6. (a) $x \leq 1$ is weaker than $x \geq 5$.
   (b) $x \leq 1$ is stronger than $x \geq 5$.
7. (a) $x \leq 4$ is weaker than $5 > x$.
   (b) $x \leq 4$ is stronger than $5 > x$.

☛ SEE ANSWER
☛ DO EXERCISE ON edX

There are a few situations that we will encounter later in the course where understanding how specific predicates are stronger than a slightly different predicate will be key to accurate reasoning. Let us first discuss these informally after which we leave the formal proof as an exercise.

- If the expression E1 $\wedge$ E2 is *true* then obviously the expression E1 is *true*. So, E1 $\wedge$ E2 is a stronger predicate (is more restrictive) than E1.
- If the expression E1 is *true* then obviously the expression E1 $\vee$ E3 is *true*. So, E1 is a stronger predicate (is more restrictive) than E1 $\vee$ E3.
- If the expression E1 $\wedge$ E2 is *true* then obviously the expression E1 is *true* and hence E1 $\vee$ E3 is true. So, E1 $\wedge$ E2 is a stronger predicate (is more restrictive) than E1 $\vee$ E3.

While this is obvious, one really should prove it:

Homework 1.5.2.2 Use the Basic Equivalences to prove the following. (Do NOT use the weakening/strengthening laws given in Figure 1.2, which we will discuss later.)

1. E1 $\wedge$ E2 $\Rightarrow$ E1
2. E1 $\Rightarrow$ E1 $\vee$ E3
3. E1 $\wedge$ E2 $\Rightarrow$ E1 $\vee$ E3

☛ SEE ANSWER
☛ DO EXERCISE ON edX

Weakening/ Strengthening:	$((\text{E1} \wedge \text{E2})$	$\Rightarrow$	$\text{E1})$	$\Leftrightarrow$	T
	$(\text{E1}$	$\Rightarrow$	$(\text{E1} \vee \text{E3}))$	$\Leftrightarrow$	T
	$((\text{E1} \wedge \text{E2})$	$\Rightarrow$	$(\text{E1} \vee \text{E3}))$	$\Leftrightarrow$	T

Figure 1.2: Weaking/strengthening laws.

Homework 1.5.2.3 Use the Basic Equivalences and/or the results from Homework 1.5.2.2 to prove that

$$\text{E1} \wedge \text{E2} \Rightarrow (\text{E1} \vee \text{E3}) \wedge \text{E2}.$$

☛ SEE ANSWER
☛ DO EXERCISE ON edX

The insights from these last two homeworks will become powerful tools as we prove programs correct. Together we will call them *Weakening/Strengthening Laws*. They are summarized in Figure 1.2. These form a second set of useful tautologies.

Homework 1.5.2.4 In Figure 1.2 we present three Weakening/Strengening Laws. This exercise shows that if you only decide to remember one, it should be the last one.

1. Show that $(\text{E1} \wedge \text{E2}) \Rightarrow \text{E1}$ is a special case of $(\text{E1} \wedge \text{E2}) \Rightarrow (\text{E1} \vee \text{E3})$.
2. Show that $\text{E1} \Rightarrow (\text{E1} \vee \text{E3})$ is a special case of $(\text{E1} \wedge \text{E2}) \Rightarrow (\text{E1} \vee \text{E3})$.

☛ SEE ANSWER
☛ DO EXERCISE ON edX

What you will find later is that it is $(\text{E1} \wedge \text{E2}) \Rightarrow (\text{E1} \vee \text{E3})$ becomes our tool of choice in many proofs.

Homework 1.5.2.5 For each of the following predicates pairs from Homework 1.5.2.1 use an equivalence style proof, the Basic Logic Equivalences, and the Weakening/strengthening laws to prove which predicate is weaker:

1. $0 \leq x \leq 10$ and $1 \leq x < 5$.
2. $x = 5 \wedge y = 4$ and $y = 4$.
3. $x \leq 5 \vee y = 3$ and $x = 5 \wedge y = 4$.
4. T and F.
5. $(\forall i | 5 \leq i \leq 10 : b(i+1) < b(i))$ and $(\forall i | 7 \leq i \leq 10 : b(i+1) < b(i))$

☛ SEE ANSWER
☛ DO EXERCISE ON edX

1.6 Enrichment

1.6.1 The Humble Programmer – Edsger W. Dijkstra ☛ to edX

Edsger W. Dijkstra was one of the most influential computer scientists. His pioneering and visionary work greatly influenced the material that underlies this course. You should start your journey by reading his ACM Turing Award acceptance speech.

☛ ACM Turing Lecture 1972: "The Humble Programmer" by Edsger W. Dijkstra

A partial tape recording of the lecture can be found on

☛ YouTube

Unfortunately, it seems like some tapes were overwritten with music...

Notice that some find the title of Dijkstra's Turing Award acceptance speech amusing. Here is an interesting quote from Alan Kay:

> "I don't know how many of you have ever met Dijkstra, but you probably know that arrogance in computer science is measured in nano-Dijkstras."

1.6.2 Typesetting proofs with LaTeX ☛ to edX

☛ Watch Video on edX
☛ Watch Video on YouTube

1.6.3 More on logic ☛ to edX

We have tried to give just enough logic background for the course to be self-contained. Here we list some resources if you want to go beyond.

Some places where you can learn more about logic:

FREE!

Dr. Elaine Rich and Prof. Alan Cline created an online introductory course on logic for use at The University of Texas at Austin titled "Fundamentals of Reasoning". They have ported this course to edX "Edge", which is a platform where unofficial courses are offered. They have graciously made it possible for you to try the first few chapters of this course, which should give you a solid background in logic. You can try this course by going to

☛ https://edge.edx.org/courses/course-v1:UT+101+2017/about

(You will have to register for edge.edx.org and sign in.)

Some books to try! (Not free)

- David Gries, The Science of Programming (Monographs in Computer Science), Springer, 1987.
- David Gries and Fred B. Schneider, A Logical Approach to Discrete Math (Texts and Monographs in Computer Science), Springer, 1994.

1.7 Wrapup

1.7.1 Additional exercises ☛ to edX

There are no additional exercises this week. Skip and go on!

1.7.2 Summary ☛ to edX

Boolean operators

- *not* (*negation*, $\neg$);
- *and* (*conjunction*, $\wedge$);
- *or* (*disjunction*, $\vee$);
- *implies* (*implication*, $\Rightarrow$);
- *is equivalent to* (*equivalence*, $\Leftrightarrow$).

For a list of logic symbols that includes LaTeX symbols, you may want to consult Wikipedia's ☛ List of logic symbols entry (https://en.wikipedia.org/wiki/List_of_logic_symbols).

p	q	not $\neg p$	and $p \wedge q$	or $p \vee q$	implies $p \Rightarrow q$	equivalent $p \Leftrightarrow q$
T	T	F	T	T	T	T
T	F	F	F	T	F	F
F	T	T	F	T	T	F
F	F	T	F	F	T	T

Precedence of Boolean operators

- First negate: $\neg p \wedge q$ is the same as $(\neg p) \wedge q$.
- Second, evaluate $\wedge$: $\neg p \wedge q \Rightarrow r$ is the same as $((\neg p) \wedge q) \Rightarrow r$.
- Third, evaluate $\vee$: $\neg p \wedge q \vee r \Rightarrow s$ is the same as $(((\neg p) \wedge q) \vee r) \Rightarrow s$.
- Fourth, evaluate $\Rightarrow$: $t \Leftrightarrow \neg p \wedge q \vee r \Rightarrow s$ is the same as $t \Leftrightarrow ((((\neg p) \wedge q) \vee r) \Rightarrow s)$.
- Last evaluate $\Leftrightarrow$.

Predicates, tautologies, contradications

A **predicate** is a logical statement that may be TRUE or FALSE depending on the values of the variables that appear in the statement.

A **tautology** is a predicate that evaluates to TRUE regardless of the choice of the variables in the predicate.

A **contradiction** is a predicate that evaluates to FALSE regardless of the choice of the variables in the predicate.

Basic Equivalences

The basic equivalences can be found in Figure 1.1 and by visiting the "Laws of Logic" tab in the LAFF-On edX course navigation bar.

The Principle of Mathematical Induction

The **Principle of Mathematical Induction** (weak induction) says that *if* one can show that

- (Base case) a property holds for $k = k_b$; *and*
- (Inductive step) if it holds for $k = K$, where $K \geq k_b$, then it is also holds for $k = K + 1$,

then one can conclude that the property holds for all integers $k \geq k_b$. Often $k_b = 0$ or $k_b = 1$.

Quantifiers

For all ... $(\forall i \mid R(i) : P(i))$ stands for "For all i that satisfy the predicate $R(i)$ the predicate $P(i)$ holds."

There exists ... $(\exists i \mid R(i) : P(i))$ stands for "There exists an i that satisfies the predicate $R(i)$ for which the predicate $P(i)$ holds."

Sum ... $(\sum i \mid R(i) : E(i))$ stands for "Sum expressions $E(i)$ for all i that satisfy the predicate $R(i)$." More traditionally this is denoted by $\sum_{R(i)} E(i)$.

Product ... $(\prod i \mid R(i) : E(i))$ stands for "Multiply expressions $E(i)$ for all i that satisfy the predicate $R(i)$." More traditionally this is denoted by $\prod_{R(i)} E(i)$.

Splitting the range

Partition S into the subsets S_0 and S_1. Partitioning means that $S_0 \cup S_1 = S$ and $S_0 \cap S_1 = \varnothing$. Then

$$
\begin{array}{rcl}
(\forall k \mid k \in S : E(k)) & \Leftrightarrow & (\forall k \mid k \in S_0 : E(k)) \wedge (\forall k \mid k \in S_1 : E(k)) \\
(\exists k \mid k \in S : E(k)) & \Leftrightarrow & (\exists k \mid k \in S_0 : E(k)) \vee (\exists k \mid k \in S_1 : E(k)) \\
(\sum k \mid k \in S : E(k)) & = & (\sum k \mid k \in S_0 : E(k)) + (\sum k \mid k \in S_1 : E(k)) \\
(\prod k \mid k \in S : E(k)) & = & (\prod k \mid k \in S_0 : E(k)) \times (\prod k \mid k \in S_1 : E(k))
\end{array}
$$

One point rule

$$
\begin{array}{rcl}
(\forall k \mid k \in \{i\} : P(k)) & \Leftrightarrow & P(i) \\
(\exists k \mid k \in \{i\} : P(k)) & \Leftrightarrow & P(i) \\
(\sum k \mid k \in \{i\} : E(k)) & = & E(i) \\
(\prod k \mid k \in \{i\} : E(k)) & = & E(i).
\end{array}
$$

Empty range

$$
\begin{array}{rcl}
(\forall k \mid k \in \varnothing : P(k)) & \Leftrightarrow & T \\
(\exists k \mid k \in \varnothing : P(k)) & \Leftrightarrow & F \\
(\sum k \mid k \in \varnothing : E(k)) & = & 0 \\
(\prod k \mid k \in \varnothing : E(k)) & = & 1.
\end{array}
$$

Weakening/strengthening laws

The basic equivalences can be found in Figure 1.2 and by visiting the "Laws of Logic" tab in the LAFF-On edX course navigation bar.