CSCE551: Theory of Computation
Lecture 1: Introduction

Max Alekseyev

University of South Carolina

January 15, 2013
Outline

The Course
   Goals
   Evaluation

Mathematical Background
   Sets
   Sequences and Tuples
   Functions and Relations

Graphs
   Undirected Graphs
   Directed Graphs

Strings and Languages
   Concatenation
   Lexicographic Ordering

Boolean Logic
The Course

Instructor: Max Alekseyev (email: maxal@cse.sc.edu)

Title: Theory of Computing


Class webpage: http://cse.sc.edu/~maxal/csce551/

Discussion board: at CSE Dropbox

Gradiance assignments: TBA

Office hours: after each lecture or by appointment
Goals

- Explore fundamental capabilities and limitations of computers
- Understand what is “computing”, what can or cannot be computed
- Determine the “cost” of computing in terms of available resources
- Classify computational problems into “easy” and “hard”
Evaluation

Evaluation of a student’s progress is based on his/her ability to understand and solve problems.
Evaluation

Evaluation of a student’s progress is based on his/her ability to understand and solve problems.

Understanding of a problem is a clear vision of what is given, what is asked, and what would constitute a solution.
Evaluation

Evaluation of a student’s progress is based on his/her ability to understand and solve problems.

Understanding of a problem is a clear vision of what is given, what is asked, and what would constitute a solution.

Solving a problem implies finding a solution as well as convincing arguments (proof) that support this solution.
Evaluation

Evaluation of a student’s progress is based on his/her ability to **understand** and **solve** problems.

Understanding of a problem is a clear vision of what is given, what is asked, and what would constitute a solution.

Solving a problem implies finding a solution as well as convincing arguments (*proof*) that support this solution.

Solving is not possible without understanding. However, understanding without being able to solve is possible.
Sets

A set is a group of objects, called elements (or members) of this set. For example, the students in this room form a set.
Sets

A set is a group of objects, called elements (or members) of this set. For example, the students in this room form a set.

A set can be defined by listing all its elements inside braces, e.g.:

\[ S = \{7, 21, 57\} \]

The order and repetitions of elements in sets do not matter - in particular, \( \{7, 21, 57\} = \{21, 57, 7\} = \{21, 7, 57, 7, 21\} \).
A set is a group of objects, called *elements* (or *members*) of this set. For example, the students in this room form a set.

A set can be defined by listing all its elements inside braces, e.g.:

\[ S = \{7, 21, 57\} \]

The order and repetitions of elements in sets *do not matter* - in particular, \( \{7, 21, 57\} = \{21, 57, 7\} = \{21, 7, 57, 7, 21\} \).

The membership is denoted by \( \in \) symbol. For example, \( 21 \in S \) but \( 10 \notin S \).
Sets

A set is a group of objects, called elements (or members) of this set. For example, the students in this room form a set.

A set can be defined by listing all its elements inside braces, e.g.:

\[ S = \{7, 21, 57\} \]

The order and repetitions of elements in sets do not matter - in particular, \( \{7, 21, 57\} = \{21, 57, 7\} = \{21, 7, 57, 7, 21\} \).

The membership is denoted by \( \in \) symbol. For example, \( 21 \in S \) but \( 10 \notin S \).

For two sets \( A \) and \( B \), we say \( A \) is a subset of \( B \) and write \( A \subseteq B \) if every member of \( A \) is also a member of \( B \). We say that \( A \) is a proper subset of \( B \) and write \( A \subsetneq B \) if \( A \) is a subset of \( B \) and not equal to \( B \).
Sets

A set is a group of objects, called elements (or members) of this set. For example, the students in this room form a set.

A set can be defined by listing all its elements inside braces, e.g.:

\[ S = \{7, 21, 57\} \]

The order and repetitions of elements in sets do not matter - in particular, \( \{7, 21, 57\} = \{21, 57, 7\} = \{21, 7, 57, 7, 21\} \).

The membership is denoted by \( \in \) symbol. For example, \( 21 \in S \) but \( 10 \notin S \).

For two sets \( A \) and \( B \), we say \( A \) is a subset of \( B \) and write \( A \subseteq B \) if every member of \( A \) is also a member of \( B \). We say that \( A \) is a proper subset of \( B \) and write \( A \subsetneq B \) if \( A \) is a subset of \( B \) and not equal to \( B \).

Q: How many distinct subsets are there in the set \( S \)? How many of them are proper?
**Sets**

A set is a group of objects, called *elements* (or *members*) of this set. For example, the students in this room form a set.

A set can be defined by listing all its elements inside braces, e.g.:

\[ S = \{7, 21, 57\} \]

The order and repetitions of elements in sets *do not matter* - in particular, \( \{7, 21, 57\} = \{21, 57, 7\} = \{21, 7, 57, 7, 21\} \).

The membership is denoted by \( \in \) symbol. For example, \( 21 \in S \) but \( 10 \notin S \).

For two sets \( A \) and \( B \), we say \( A \) is a *subset* of \( B \) and write \( A \subseteq B \) if every member of \( A \) is also a member of \( B \). We say that \( A \) is a *proper subset* of \( B \) and write \( A \subset B \) if \( A \) is a subset of \( B \) and not equal to \( B \).

Q: How many distinct subsets are there in the set \( S \)? How many of them are proper?

The set of all subsets of a set \( A \) is called the *power set* of \( A \) and denoted \( 2^A \).
Examples of Sets

The set with no elements is called the *empty set* and denoted $\emptyset$. The empty set is a subset of any other set.
Examples of Sets

The set with no elements is called the *empty set* and denoted $\emptyset$. The empty set is a subset of any other set.

The set of *natural numbers* $\mathcal{N}$ (or $\mathbb{N}$):

$$\mathcal{N} = \{1, 2, 3, \ldots \}.$$
Examples of Sets

The set with no elements is called the *empty set* and denoted $\emptyset$. The empty set is a subset of any other set.

The set of *natural numbers* $\mathbb{N}$ (or $\mathbb{N}$):

$$\mathbb{N} = \{1, 2, 3, \ldots \}.$$ 

The set of *integers* $\mathbb{Z}$ (or $\mathbb{Z}$):

$$\mathbb{Z} = \{ \ldots, -2, -1, 0, 1, 2, \ldots \}.$$ 

It is clear that $\mathbb{N} \subseteq \mathbb{Z}$.
Examples of Sets

The set with no elements is called the empty set and denoted $\emptyset$. The empty set is a subset of any other set.

The set of natural numbers $\mathbb{N}$ (or $\mathbb{N}$):

$$\mathbb{N} = \{1, 2, 3, \ldots \}.$$

The set of integers $\mathbb{Z}$ (or $\mathbb{Z}$):

$$\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots \}.$$

It is clear that $\mathbb{N} \subseteq \mathbb{Z}$.

The set of perfect squares $\{n \mid n = m^2 \text{ for some } m \in \mathbb{N}\}$ is a subset of both $\mathbb{N}$ and $\mathbb{Z}$.
Examples of Sets

The set with no elements is called the *empty set* and denoted $\emptyset$. The empty set is a subset of any other set.

The set of *natural numbers* $\mathbb{N}$ (or $\mathbb{N}$):

$$\mathbb{N} = \{1, 2, 3, \ldots \}.$$  

The set of *integers* $\mathbb{Z}$ (or $\mathbb{Z}$):

$$\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots \}.$$  

It is clear that $\mathbb{N} \subseteq \mathbb{Z}$.

The set of perfect squares $\{n \mid n = m^2 \text{ for some } m \in \mathbb{N}\}$ is a subset of both $\mathbb{N}$ and $\mathbb{Z}$.

The number of elements in a set $A$ is called the *cardinality* (or *size*) of the set and denoted by $|A|$. We have $|\emptyset| = 0$ and $|\mathbb{N}| = |\mathbb{Z}| = \infty$.

Q: For a finite set $A$, what is $|2^A|$?
Set Operations

For given two sets $A$ and $B$, one can define the following set operations:

**Union:** $A \cup B$. Example: $\{1, 2, 3\} \cup \{1, 3, 5\} = \{1, 2, 3, 5\}$.

**Intersection:** $A \cap B$. Example: $\{1, 2, 3\} \cap \{1, 3, 5\} = \{1, 3\}$.

**Difference:** $A \setminus B$. Example: $\{1, 2, 3\} \setminus \{1, 3, 5\} = \{2\}$.

In the case of $B \subset A$, the result of $A \setminus B$ is also called the **complement of $B$ in $A$** and denoted $\overline{B}$. 

The Venn diagram is a convenient way to illustrate the set operations:
Set Operations

For given two sets $A$ and $B$, one can define the following set operations:

**Union:** $A \cup B$. Example: $\{1, 2, 3\} \cup \{1, 3, 5\} = \{1, 2, 3, 5\}$.

**Intersection:** $A \cap B$. Example: $\{1, 2, 3\} \cap \{1, 3, 5\} = \{1, 3\}$.

**Difference:** $A \setminus B$. Example: $\{1, 2, 3\} \setminus \{1, 3, 5\} = \{2\}$.

In the case of $B \subset A$, the result of $A \setminus B$ is also called the complement of $B$ in $A$ and denoted $\bar{B}$.

The Venn diagram is a convenient way to illustrate the set operations:
Sequences and Tuples

A sequence is a list of objects in some order. For example, sequences of the students’ names in alphabetic order such as \((Alice,Bob)\). In contrast to sets, repetitions and order matter in sequences. The sequences \((7,21,57)\) and \((7,7,21,57)\) are not equal.

Finite sequences are called tuples. In particular, a sequence with \(k\) elements is called \(k\)-tuple (as well as pair, triple, quadriple, etc.)
Sequences and Tuples

A sequence is a list of objects in some order. For example, sequences of the students’ names in alphabetic order such as (Alice, Bob). In contrast to sets, repetitions and order matter in sequences. The sequences (7, 21, 57) and (7, 7, 21, 57) are not equal.

Finite sequences are called tuples. In particular, a sequence with \( k \) elements is called \( k \)-tuple (as well as pair, triple, quadriple, etc.)

All \( k \)-tuples \((x_1, x_2, \ldots, x_k)\), where \( x_i \) is taken from the set \( A_i \), form a set \( A_1 \times A_2 \times \cdots \times A_k \), called the Cartesian product or cross product of the sets \( A_1, A_2, \ldots, A_k \). For example, if \( A_1 = \{p, q\} \) and \( A_2 = \{1, 2, 3\} \) then

\[
A_1 \times A_2 = \{(p, 1), (p, 2), (p, 3), (q, 1), (q, 2), (q, 3)\}.
\]

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>(p,1)</td>
<td>(p,2)</td>
<td>(p,3)</td>
</tr>
<tr>
<td>q</td>
<td>(q,1)</td>
<td>(q,2)</td>
<td>(q,3)</td>
</tr>
</tbody>
</table>
Functions

A function (mapping) $f$ sets up a correspondence between the elements of one set $A$ and elements of the other set $B$, written $f : A \rightarrow B$. In particular, if an element $a \in A$ corresponds to an element $b \in B$ under the function $f$, we write $f(a) = b$, where $a$ is called the input (or argument) and $b$ is called the output (or value) of $f$. 
Functions

A function (mapping) $f$ sets up a correspondence between the elements of one set $A$ and elements of the other set $B$, written $f : A \rightarrow B$. In particular, if an element $a \in A$ corresponds to an element $b \in B$ under the function $f$, we write $f(a) = b$, where $a$ is called the input (or argument) and $b$ is called the output (or value) of $f$.

The set $A$ is called the domain of $f$ while the set $B$ is called the range of $f$. 
Functions

A function (mapping) $f$ sets up a correspondence between the elements of one set $A$ and elements of the other set $B$, written $f : A \rightarrow B$. In particular, if an element $a \in A$ corresponds to an element $b \in B$ under the function $f$, we write $f(a) = b$, where $a$ is called the input (or argument) and $b$ is called the output (or value) of $f$.

The set $A$ is called the domain of $f$ while the set $B$ is called the range of $f$.

For a finite set $A$, the function $f : A \rightarrow B$ can be defined by a table of its values of elements of $A$.

For example, for $A = \mathbb{Z}_5 = \{0, 1, 2, 3, 4\}$ (the set of integers modulo 5), a function $f : \mathbb{Z}_5 \rightarrow \mathbb{Z}_5$ can be defined by the following table:

<table>
<thead>
<tr>
<th>$a$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(a)$</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>
Types of Functions

A function $f : A_1 \times A_2 \times \cdots \times A_k \rightarrow B$ is called a $k$-ary function. *Unary functions* correspond to the case $k = 1$, while *binary functions* correspond to the case $k = 2$. A function $f : A_1 \times A_2 \times \cdots \times A_k \rightarrow \{\text{true}, \text{false}\}$ is called a *predicate* or *property*. For example, the property *even* defines evenness of a given integer: even(4) = true and even(5) = false. A property $f : A_1 \times A_2 \times \cdots \times A_k \rightarrow \{\text{true}, \text{false}\}$, whose domain is the set of $k$-tuples, is called a *$k$-ary relation* (on $A$). An example of a binary relation is the *beats* relation between scissors, paper, and stone (see p.9 in Sipser). For binary relation $R$, we often use infix notation writing $xRy$ instead of $R(x,y) = \text{true}$. 
Types of Functions

A function \( f : A_1 \times A_2 \times \cdots \times A_k \rightarrow B \) is called a \( k \)-ary function. \( \textit{Unary functions} \) correspond to the case \( k = 1 \), while \( \textit{binary functions} \) correspond to the case \( k = 2 \).

A function \( f : A \rightarrow \{ \text{TRUE}, \text{FALSE} \} \) is called a \textit{predicate} or \textit{property}. For example, the property \textit{even} defines evenness of a given integer: \( \text{even}(4) = \text{TRUE} \) and \( \text{even}(5) = \text{FALSE} \).
Types of Functions

A function $f : A_1 \times A_2 \times \cdots \times A_k \rightarrow B$ is called a $k$-ary function. **Unary functions** correspond to the case $k = 1$, while **binary functions** correspond to the case $k = 2$.

A function $f : A \rightarrow \{\text{TRUE}, \text{FALSE}\}$ is called a **predicate** or **property**. For example, the property **even** defines evenness of a given integer: even(4) = TRUE and even(5) = FALSE.

A property $f : A \times A \times \cdots \times A \rightarrow \{\text{TRUE}, \text{FALSE}\}$, whose domain is the set of $k$-tuples, is called **$k$-ary relation** (on $A$). An example of a binary relation is the **beats** relation between scissors, paper, and stone (see p.9 in Sipser).
Types of Functions

A function \( f : A_1 \times A_2 \times \cdots \times A_k \longrightarrow B \) is called a \( k \)-ary function. \textit{Unary functions} correspond to the case \( k = 1 \), while \textit{binary functions} correspond to the case \( k = 2 \).

A function \( f : A \longrightarrow \{\text{TRUE, FALSE}\} \) is called a \textit{predicate} or \textit{property}. For example, the property \textit{even} defines evenness of a given integer: \text{even}(4) = \text{TRUE} and \text{even}(5) = \text{FALSE}.

A property \( f : A \times A \times \cdots \times A \longrightarrow \{\text{TRUE, FALSE}\} \), whose domain is the set of \( k \)-tuples, is called \( k \)-ary \textit{relation} (on \( A \)). An example of a binary relation is the \textit{beats} relation between scissors, paper, and stone (see p.9 in Sipser).

For binary relation \( R \), we often use \textit{infix notation} writing \( xRy \) instead of \( R(x, y) = \text{TRUE} \).
A binary relation $R$ on a set $A$ is called an equivalence relation if $R$ is:

- **Reflexive:** for every $x \in A$, $xRx$;
- **Symmetric:** for every $x, y \in A$, $xRy$ implies $yRx$;
- **Transitive:** for every $x, y, z \in A$, $xRy$ and $yRz$ implies $xRz$.

The equality relation is an example of equivalence relation.
A binary relation $R$ on a set $A$ is called an *equivalence relation* if $R$ is:

- **Reflexive:** for every $x \in A$, $xRx$;
- **Symmetric:** for every $x, y \in A$, $xRy$ implies $yRx$;
- **Transitive:** for every $x, y, z \in A$, $xRy$ and $yRz$ implies $xRz$.

The equality relation is an example of equivalence relation.

Q: What other equivalence relations you know?
A binary relation $R$ on a set $A$ is called an *equivalence relation* if $R$ is:

- **Reflexive:** for every $x \in A$, $xRx$;
- **Symmetric:** for every $x, y \in A$, $xRy$ implies $yRx$;
- **Transitive:** for every $x, y, z \in A$, $xRy$ and $yRz$ implies $xRz$.

The equality relation is an example of equivalence relation.

Q: What other equivalence relations do you know?

Q: Is the *less or equal* relation ($\leq$) an equivalence relation?
Undirected Graphs

An (undirected) graph is a pair \((V, E)\) where \(V\) is a set of objects, called nodes or vertices, and \(E\) is a set of 2-element subsets of \(V\), called edges. Graphs have convenient graphical representation: vertices are drawn as dots on a plane, while edges are drawn as lines connecting corresponding pairs of vertices, e.g.:

In unlabeled graph, the vertices have no labels and can be distinguished only from the perspective of edges.

Q: Which vertices in this graph are indistinguishable?
Undirected Graphs

An (undirected) graph is a pair \((V, E)\) where \(V\) is a set of objects, called nodes or vertices, and \(E\) is a set of 2-element subsets of \(V\), called edges.

In labeled graph \(G = (V, E)\) the vertices are labeled (usually with elements of \(V\)). In particular, for this the shown graph, we have:

\[V = \{1, 2, 3, 4, 5, 6\}\]
\[E = \{(1, 2), (2, 4), (2, 3), (3, 5), (4, 5), (3, 6)\}\]
(by convention, we write \((1, 2)\) instead of \(\{1, 2\}\) and assume that \((1, 2) = (2, 1)\) is the same edge)

Two vertices are adjacent if they are connected by an edge (e.g., vertices 3 and 5). A vertex and an edge are incident if the vertex represents an endpoint of the edge.
Undirected Graphs

An (undirected) graph is a pair \((V, E)\) where \(V\) is a set of objects, called nodes or vertices, and \(E\) is a set of 2-element subsets of \(V\), called edges.

A path in a graph is a sequence of vertices, where every two consecutive vertices are adjacent (e.g., vertices \((1, 2, 4, 5, 3, 2)\) form a path). A graph is connected if there is a path between any two vertices; and disconnected otherwise.

A cycle is a path where starting and ending vertices coincide and viewed as a single vertex (so that a cycle has neither starting nor ending vertex). A path/cycle is called simple if it does not repeat any vertices.
An (undirected) graph is a pair \((V, E)\) where \(V\) is a set of objects, called nodes or vertices, and \(E\) is a set of 2-element subsets of \(V\), called edges.

A graph having no cycles is called acyclic graph (or forest). An acyclic graph consists of one or more connected components, called trees. In other words, a tree is a connected graph without cycles.
Directed Graphs

A directed graph (digraph) is a pair \((V, E)\) where \(V\) is a set of objects, called *nodes* or *vertices*, and \(E\) is a set of pairs of elements of \(V\), called *directed edges* or *arcs*.

Differences from undirected graph:

- Vertices have incoming degree (*indegree*) as well as outgoing degree (*outdegree*).
- Paths are directed; a directed path comes into a vertex through an incoming edge and leaves it through outgoing edge.
- In a *strongly connected* (component of) graph, there is a directed path connecting any vertex with any other vertex.

Q: Find strongly connected component in the shown graph.
Strings and Languages

- An *alphabet* is a nonempty finite set whose elements are called *symbols*. E.g.: $\Sigma_1 = \{0, 1\}$ or $abc = \{a, b, c, d, \ldots, x, y, z\}$ etc.

- A *string* (over some fixed alphabet) is a finite sequence of symbols. E.g.: 01001 is a sting over $\Sigma_1$ while “samplestring” is a string over $abc$.

- The length of a string $w$ equals the number of symbols in $w$, denoted $|w|$. A string with 0 symbols, called the *empty string*, is denoted $\varepsilon$. So, by definition, $|\varepsilon| = 0$.

- A string $z$ is a *substring* of a string $w$ if $z$ appears consecutively within $w$. E.g.: “sam”, “sample”, and ”ring” are substrings of “samplestring”. *Prefix* and *suffix* are particular kinds of substrings.

Q: If $z$ is a substring of $w$, what can we say about $|z|$ and $|w|$?
Concatenation of strings $x$ and $y$, denoted $xy$, is the string obtained by appending $y$ to the end of $x$. In particular, if $x = x_1x_2 \ldots x_m$ and $y = y_1y_2 \ldots y_n$, then $xy = x_1x_2 \ldots x_my_1y_2 \ldots y_n$.

Q: Is it true that $z$ is a substring of $w$ if and only if there exist strings $x, y$ such that $w = xzy$?
Strings and Languages

Concatenation of strings $x$ and $y$, denoted $xy$, is the string obtained by appending $y$ to the end of $x$. In particular, if $x = x_1x_2 \ldots x_m$ and $y = y_1y_2 \ldots y_n$, then $xy = x_1x_2 \ldots x_my_1y_2 \ldots y_n$.

Q: Is it true that $z$ is a substring of $w$ if and only if there exist strings $x, y$ such that $w = xzy$?

To denote $k$-fold concatenation of a string $x$ with itself we use the superscript notation:

$$
\underbrace{xx \cdots x}^k = x^k.
$$
Lexicographic Ordering

The symbols of an alphabet can be ordered in some way. However, often there exists natural ordering such as $0 < 1$ in $\Sigma_1$ or $a < b < c < \cdots < z$ in $abc$.

The strings over an ordered alphabet can be ordered *lexicographically* such as in dictionaries, except that shorter prefixes of any string precede it.

A string $x = x_1x_2 \ldots x_m$ is le $y = y_1y_2 \ldots y_n$, then $x$ is lexicographically smaller then $y$ is one of the following conditions holds:

- $x$ is a prefix of $y$;
- there exists an integer $k$ such that $1 \leq k \leq \min\{m, n\}$ and for $j = 1, 2, \ldots, k - 1$, $x_j = y_j$ but $x_k < y_k$.

Q: Reformulate this definition in terms of substrings.
Boolean Operations

Boolean values are TRUE = 1 and FALSE = 0.

Boolean operations: conjunction (AND, $\land$), disjunction (OR, $\lor$), exclusive or (XOR, addition modulo 2, $\oplus$), equality (equivalence, iff, “if and only if”, $\leftrightarrow$)

<table>
<thead>
<tr>
<th>$\land$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\lor$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\oplus$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\leftrightarrow$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Boolean Operations

Boolean values are \texttt{TRUE} = 1 and \texttt{FALSE} = 0.

Boolean operations: \textit{conjunction} (AND, $\land$), \textit{disjunction} (OR, $\lor$), \textit{exclusive or} (XOR, addition modulo 2, $\oplus$), \textit{equality} (equivalence, iff, “if and only if”, $\iff$)

\[
\begin{array}{c|cc}
\land & 0 & 1 \\
0 & 0 & 0 \\
1 & 0 & 1 \\
\end{array} \quad \begin{array}{c|cc}
\lor & 0 & 1 \\
0 & 0 & 1 \\
1 & 1 & 1 \\
\end{array} \quad \begin{array}{c|cc}
\oplus & 0 & 1 \\
0 & 0 & 1 \\
1 & 1 & 0 \\
\end{array} \quad \begin{array}{c|cc}
\iff & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 1 \\
\end{array}
\]

Other Boolean operations: \textit{negation} (NOT, $\neg$), \textit{implication} $\rightarrow$

Q: In what respect negation and implication differ from the operations mentioned above?
Boolean Operations

Boolean values are \textbf{TRUE} = 1 and \textbf{FALSE} = 0.

Boolean operations: \textit{conjunction} (AND, \(\land\)), \textit{disjunction} (OR, \(\lor\)), \textit{exclusive or} (XOR, addition modulo 2, \(\oplus\)), \textit{equality} (equivalence, \(\iff\), “if and only if”, \(\leftrightarrow\))

\[
\begin{array}{c|cc}
\land & 0 & 1 \\
0 & 0 & 0 \\
1 & 0 & 1 \\
\end{array}
\begin{array}{c|cc}
\lor & 0 & 1 \\
0 & 0 & 1 \\
1 & 1 & 1 \\
\end{array}
\begin{array}{c|cc}
\oplus & 0 & 1 \\
0 & 0 & 1 \\
1 & 1 & 0 \\
\end{array}
\begin{array}{c|cc}
\leftrightarrow & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 1 \\
\end{array}
\]

Other Boolean operations: \textit{negation} (NOT, \(\neg\)), \textit{implication} \(\rightarrow\)

Q: In what respect negation and implication differ from the operations mentioned above?

\(\neg 0 = 1, \neg 1 = 0;\)

\(0 \rightarrow 0 = 1, 0 \rightarrow 1 = 1, \text{(FALSE implies whatever!)}\)

\(1 \rightarrow 0 = 0, 1 \rightarrow 1 = 1, \text{(TRUE can imply only TRUE)}\).

Arguments of boolean operations are called \textit{operands}. 
Boolean Operations

Both \{\land, \neg\} and \{\lor, \neg\} are complete set of Boolean operations, i.e., \textit{any} binary Boolean operations can be expressed in terms of the elements of either of these sets.

Boolean operations \land and \lor satisfy the \textit{distributive law} similarly for arithmetic operations \times and +:

- For any three numbers \(a, b, c\), we have
  \[
  a \times (b + c) = (a \times b) + (a \times c)
  \]

- Similarly, for any three Boolean values \(P, Q, R\), we have
  \[
  P \land (Q \lor R) = (P \land Q) \lor (P \land R)
  \]
  as well as
  \[
  P \lor (Q \land R) = (P \lor Q) \land (P \lor R).
  \]