Theory of Automata

Formal Languages
Why

Quite often, we've found, teaching theory, Undergrads are bored, puzzled and weary.
``Sterile proofs are from Hell--
Teach us HTML!''
They don't get it--in turn, we're left teary.

But good theorems don't formally hide
The human emotion supplied
In their proofs' demonstration.
Scientific elation
Belongs not in formaldehyde.

Rigor without mortis, our goal,
Is for sure a negotiable shoal--
We need not run aground.
In our teaching, we've found
A proof's spirit, in verse, is made whole.

Formal methods are meant to elide
Ideas we aspire to confide.
So to ``lemma," ``parameter,"
Add ``iambic pentameter."
This should help students' ennui subside.

Pedagogy's best shown by example.
Ensuite, you are offered a sample
In the hope you're not wincing.
Find our thesis convincing--
We expect explanations are ample.

Martin Cohn
Harry Mairson
Computer Science Department
Brandeis University
Overview

• Theory of computation:
  – Intensively studied in first half of 20th century.
  – Started by mathematicians and logicians.
  – Later, became an independent academic discipline and was separated from mathematics.
  – Some pioneers:
    • Alonzo Church: Lambda calculus
    • Kurt Godel (1906-1978): Incompleteness theorems, Godel prize (5000)
    • Alan Turing (1912-1954): Turing machine
    • Stephen Kleene (1909-1994): Kleene star, regular expression
    • John von Neumann (1903-1957): game theory
    • Claude Shannon (1916-2001): father of information theory
    • Noam Chomsky (1928-): CNF, Chomsky hierarchy

• Central areas: Automata, Computability, Complexity

• Computability: What are the fundamental capabilities and limitations of computers?
  – Classify problems as solvable and unsolvable

• Complexity: What makes some problems computationally hard and others easy?
  – Classify solvable problems as easy ones and hard ones

• Both deal with formal models of computation: Turing machines, recursive functions, lambda calculus, and production systems
Overview

- **Automata theory**: study of **abstract machines** and problems they are able to solve.
  - closely related to formal language theory as the automata are often classified by the class of formal languages they are able to recognize.
  - An abstract machine, also called an abstract computer, is a theoretical model of a computer hardware or software system used in automata theory.
  - A typical abstract machine consists of a definition in terms of input, output, and the set of allowable operations used to turn the former into the latter. E.g., FSM, PDA, Turing machine.

- **Formal languages**: A set of strings over a given alphabet.
  - In contrast to natural language
  - Often defined by formal grammar, which is a set of formation rules that describe which strings formed from the alphabet of a formal language are *syntactically* valid.
  - Used for the precise definition of data formats and the syntax of program languages.
  - Play a crucial role in the development of compilers.
  - Used in logic / foundations of mathematics to represent the syntax of formal theories
  - Regular languages, context-free languages, context-sensitive languages, recursive languages (decidable), recursively enumerable languages (semi-decidable)
  - Formal language theory uses separate formalisms, automata, to describe their recognizers.
Grammars, Languages, and Machines

Grammar

Generates

Language

Recognizes or Accepts

Machine
Languages and Machines

- SD Languages (Recursively enumerable)
- D Languages (Recursive)
- Context-Free Languages
- Regular Languages
- FSMs
- PDAs
- Turing Machines
Recursive languages (decidable) are not included.
### The Chomsky Hierarchy

<table>
<thead>
<tr>
<th>Grammar</th>
<th>Languages</th>
<th>Automaton</th>
<th>Production rules (constraints)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type-0</td>
<td>Recursively enumerable</td>
<td>Turing machine</td>
<td>$\alpha \rightarrow \beta$ (no restrictions)</td>
</tr>
<tr>
<td>Type-1</td>
<td>Context-sensitive</td>
<td>Linear-bounded non-deterministic Turing machine</td>
<td>$\alpha A\beta \rightarrow \alpha \gamma \beta$</td>
</tr>
<tr>
<td>Type-2</td>
<td>Context-free</td>
<td>Non-deterministic pushdown automaton</td>
<td>$A \rightarrow \gamma$</td>
</tr>
<tr>
<td>Type-3</td>
<td>Regular</td>
<td>Finite state automaton</td>
<td>$A \rightarrow a$, $A \rightarrow aB$</td>
</tr>
</tbody>
</table>

- Summarizes each of Chomsky's four types of grammars, the class of language it generates, the type of automaton that recognizes it, and the form its rules must have.
- There are other categories of formal languages not included, e.g., recursive
Noam Chomsky

- 1928 –
  - Professor emeritus at MIT
  - Father of modern linguistics
  - most cited scholar
  - world’s top public intellectual
  - Still holds office
  - Chomsky normal form (CNF)
    - context-free grammar

- Controversial political critic
- Often receives undercover police protection
Mathematical Background

Appendix A
Boolean Logic Wffs

A wff is any string that is formed according to the following rules:

- A propositional symbol is a wff.
- If $P$ is a wff, then $\neg P$ is a wff.
- If $P$ and $Q$ are wffs, then so are: $P \lor Q$, $P \land Q$, $P \rightarrow Q$, and $P \leftrightarrow Q$
- If $P$ is a wff, then $(P)$ is a wff.

Note: A proposition is a statement that has a truth value.
Truth Tables Define Operators

<table>
<thead>
<tr>
<th>$P$</th>
<th>$Q$</th>
<th>$\neg P$</th>
<th>$P \lor Q$</th>
<th>$P \land Q$</th>
<th>$P \rightarrow Q$</th>
<th>$P \leftrightarrow Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>
When Wffs are True

• A Boolean wff is **valid** or is a **tautology** iff it is true for all assignments of truth values to the variables it contains.

• A Boolean wff is **satisfiable** iff it is true for at least one assignment of truth values to the variables it contains.

• A Boolean wff is **unsatisfiable** iff it is false for all assignments of truth values to the variables it contains.

• Two wffs $P$ and $Q$ are **equivalent**, written $P \equiv Q$, iff they have the same truth values regardless of the truth values of the variables they contain.
Using Truth Tables

$P \lor \neg P$ is a tautology:

<table>
<thead>
<tr>
<th>$P$</th>
<th>$\neg P$</th>
<th>$P \lor \neg P$</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>
Properties of Boolean Operators

• \( \lor \) and \( \land \) are commutative and associative.

• \( \leftrightarrow \) is commutative but not associative.

• \( \lor \) and \( \land \) are idempotent:

\[ (P \lor P) \equiv P \]

• \( \lor \) and \( \land \) distribute over each other:
  • \( P \land (Q \lor R) \equiv (P \land Q) \lor (P \land R) \)
  • \( P \lor (Q \land R) \equiv (P \lor Q) \land (P \lor R) \)
More Properties

- **Absorption laws:**
  - \( P \land (P \lor Q) \equiv P \)
  - \( P \lor (P \land Q) \equiv P \)

- **Double negation:** \( \neg\neg P \equiv P \)

- **de Morgan’s Laws:**
  - \( \neg(P \land Q) \equiv (\neg P \lor \neg Q) \)
  - \( \neg(P \lor Q) \equiv (\neg P \land \neg Q) \)
Entailment

A set $A$ of wffs *logically implies* or *entails* a conclusion $Q$ iff, whenever all of the wffs in $A$ are true, $Q$ is also true.

Example:

$$A \land B \land C \quad \text{entail} \quad A \rightarrow D$$
Inference Rules

• An inference rule is **sound** iff, whenever it is applied to a set $A$ of axioms, any conclusion that it produces is entailed by $A$. An entire proof is sound iff it consists of a sequence of inference steps each of which was constructed using a sound inference rule.

• A set of inference rules $R$ is **complete** iff, given any set $A$ of axioms, all statements that are entailed by $A$ can be proved by applying the rules in $R$. 
Some Sound Inference Rules

- **Modus ponens**: From \((P \rightarrow Q)\) and \(P\), conclude \(Q\).
- **Modus tollens**: From \((P \rightarrow Q)\) and \(\neg Q\), conclude \(\neg P\).
- **Or introduction**: From \(P\), conclude \((P \lor Q)\).
- **And introduction**: From \(P\) and \(Q\), conclude \((P \land Q)\).
- **And elimination**: From \((P \land Q)\), conclude \(P\) or conclude \(Q\).
First-Order Logic

A well-formed formula (wff) in first-order logic is an expression that can be formed by:

- If $P$ is an $n$-ary predicate and each of the expressions $x_1, x_2, \ldots, x_n$ is a term, then an expression of the form $P(x_1, x_2, \ldots, x_n)$ is a wff. If any variable occurs in such a wff, then that variable is free.
- If $P$ is a wff, then $\neg P$ is a wff.
- If $P$ and $Q$ are wffs, then so are $P \lor Q, P \land Q, P \rightarrow Q,$ and $P \leftrightarrow Q.$
- If $P$ is a wff, then $(P)$ is a wff.
- If $P$ is a wff, then $\forall x \ (P)$ and $\exists x \ (P)$ are wffs. Any free instance of $x$ in $P$ is bound by the quantifier and is then no longer free.
Sentences

A wff with no free variables is called a **sentence** or a **statement**.

1.  $\text{Bear}(\text{Smoky})$
2.  $\forall x (\text{Bear}(x) \rightarrow \text{Animal}(x))$
3.  $\forall x (\text{Animal}(x) \rightarrow \text{Bear}(x))$
4.  $\forall x (\text{Animal}(x) \rightarrow \exists y (\text{Mother-of}(y, x)))$
5.  $\forall x ((\text{Animal}(x) \land \neg \text{Dead}(x)) \rightarrow \text{Alive}(x))$

A **ground instance** is a sentence that contains no variables.
Interpretations

• An interpretation for a sentence $w$ is a pair $(D, I)$, where $D$ is a universe of objects. $I$ assigns meaning to the symbols of $w$: it assigns values, drawn from $D$, to the constants in $w$ and it assigns functions and predicates (whose domains and ranges are subsets of $D$) to the function and predicate symbols of $w$.

• A sentence $w$ is valid iff it is true in all interpretations.

• A sentence $w$ is satisfiable iff there exists some interpretation in which $w$ is true.

• A sentence $w$ is unsatisfiable iff $\neg w$ is valid.
Examples

- \( \forall x ((P(x) \land Q(\text{Smoky})) \rightarrow P(x)) \)

- \( \neg (\forall x (P(x) \lor \neg (P(x)))) \)

- \( \forall x (P(x, x)) \)
Additional Sound Inference Rules

- **Quantifier exchange**:  
  - From $\neg \exists x \ (P)$, conclude $\forall x \ (\neg P)$  
  - From $\forall x \ (\neg P)$, conclude $\neg \exists x \ (P)$  
  - From $\neg \forall x \ (P)$, conclude $\exists x \ (\neg P)$  
  - From $\exists x \ (\neg P)$, conclude $\neg \forall x \ (P)$

- **Universal instantiation**: For any constant $C$, from $\forall x \ (P(x))$, conclude $P(C)$.

- **Existential generalization**: For any constant $C$, from $P(C)$ conclude $\exists x \ (P(x))$. 
A Simple Proof

Assume the following three axioms:

[1] \( \forall x (P(x) \land Q(x) \rightarrow R(x)) \)
[2] \( P(X_1) \)
[3] \( Q(X_1) \)

We prove \( R(X_1) \) as follows:

[4] \( P(X_1) \land Q(X_1) \rightarrow R(X_1) \) (Universal instantiation, [1])
[5] \( P(X_1) \land Q(X_1) \) (And introduction, [2], [3])
[6] \( R(X_1) \) (Modus ponens, [5], [4])
Theory

- A first order theory is a set of axioms and the set of all theorems that can be proved, using a set of **sound and complete** inference rules, from those axioms.
- A theory is consistent iff there is no sentence $P$ such that both $P$ and $\neg P$ are theorems.
  - inconsistent: contains such a contradiction.
- Let $w$ be a world plus an interpretation (that maps logical objects to objects in the world). We say a theory is sound w.r.t. $w$ iff every theorem in the theory corresponds to a fact that is true in $w$.

We say a theory is complete w.r.t. $w$ iff every fact that is true in $w$ corresponds to a theorem in the theory.
Gödel’s Theorems

• Completeness Theorem: there exists some set of inference rules $R$ such that, given any set of axioms $A$ and a sentence $c$, there is a proof of $c$, starting with $A$ and applying the rules in $R$, iff $c$ is entailed by $A$.

• Incompleteness Theorem: any theory that is derived from a decidable set of axioms and that characterizes the standard behavior of the constants 0 and 1, plus the functions $\text{plus}$ and $\text{times}$ on the natural numbers, cannot be both consistent and complete.
Kurt Gödel

• 1906 – 1978. One of the greatest logicians of all time

• Gödel and Einstein … were known to take long walks together to and from the Institute for Advanced Study. … toward the end of his life Einstein confided that his "own work no longer meant much, that he came to the Institute merely…to have the privilege of walking home with Gödel. "

• 1947, Einstein … accompanied Gödel to his U.S. citizenship exam, where they acted as witnesses. Gödel had confided in them that he had discovered an inconsistency in the U.S. Constitution, one that would allow the U.S. to become a dictatorship

• In later life, suffered periods of mental instability… fear of being poisoned; wouldn't eat unless his wife tasted his food for him. Late in 1977, wife was hospitalized for six months. In her absence, he refused to eat, eventually starving himself to death. He weighed 65 pounds when he died.
Extensions

• Propositional logic

• First order logic: use variables
  – Range over individuals

• Second order logic
  – Additional variables that range over sets of individuals
  – E.g., $\forall P \forall x (x \in P \lor x \notin P)$

• Higher order logic
  – A predicate can take one or more other predicates as arguments
Sets

- $S_1 = \{13, 11, 8, 23\}$
- $S_2 = \{8, 23, 11, 13\}$
- $S_3 = \{8, 8, 23, 23, 11, 11, 13, 13\}$
- $S_4 = \{\text{apple, pear, banana, grape}\}$
- $S_5 = \{\text{January, February, March, April, May, June, July, August, September, October, November, December}\}$
- $S_6 = \{x : x \in S_5 \text{ and } x \text{ has 31 days}\}$
- $S_7 = \{\text{January, March, May, July, August, October, December}\}$
Sets

• $S_8 = \{i : \exists x \in \mathbb{N} (i = 2x)\}$

• $S_9 = \{0, 2, 4, 6, 8, \ldots\}$

• $S_{10} = \text{the even natural numbers}$

• $S_{11} = \text{the syntactically valid C programs}$

• $S_{12} = \{x : x \in S_{11} \text{ and } x \text{ never gets into an infinite loop}\}$

• $S_{13} = \{\text{finite length strings of } a' \text{'s and } b' \text{'s}\}$
Sets

- $\mathbb{N} = \text{the nonnegative integers}$ (also called the natural numbers)

- $\mathbb{Z} = \text{the integers}$ (… -3, -2, -1, 0, 1, 2, 3, …)
Defining a Set Using Programs

• Write a program that enumerates the elements of \( S \).

• Write a program that decides \( S \) by implementing the characteristic function of \( S \). Such a program returns \( True \) if run on some element that is in \( S \) and \( False \) if run on an element that is not in \( S \).
  – A characteristic function can be used to determine whether or not a given element is in \( S \).

• Formally, a function \( f \) with domain \( Domain \) is a characteristic function of a set \( S \) iff \( f(x) = True \) if \( x \) is an element of \( S \) and \( False \) otherwise.
  – Note, \( f \) must be total
Cardinality

The cardinality of every set we will consider is:

- a natural number (if S is finite),

- Empty set \( \emptyset \)
  - so \( |\emptyset| = 0 \)
  - \(|\{\emptyset\}| = ?\)

- countably infinite: S has the same number of elements as there are natural numbers

- uncountably infinite: S has more elements than there are natural numbers
Relating Sets to Each Other

(a) $B \subseteq A$

(b) $A \cap B$

(c) $A \cap B$

(d) $A \subseteq B$

(e) $A$

(f) $A \cup B$
Sets of Sets

• The **power set** of $A$ is the set of all subsets of $A$.

Let $A = \{1, 2, 3\}$. Then:

$$P(A) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$$

Cardinality of power set?
Ordered pair

An *ordered pair* is a sequence of two objects, written:

\[(x, y)\]

Order matters. So (unless \(x\) and \(y\) are equal):

\[(x, y) \neq (y, x)\]
Cartesian Product

The **Cartesian product** of two sets $A$ and $B$ is the set of all ordered pairs $(a, b)$ such that $a \in A$ and $b \in B$. We write it as:

$$A \times B$$

If $A$ and $B$ are finite, the cardinality of their Cartesian product is:

$$|A \times B| = |A| \cdot |B|$$
Cartesian Product

Let $A$ be: $\{\text{Dave, Sara, Billy}\}$
Let $B$ be: $\{\text{cake, pie, ice cream}\}$

\[
A \times B = \{(\text{Dave, cake}), (\text{Dave, pie}), \\
(\text{Dave, ice cream}), (\text{Sara, cake}), \\
(\text{Sara, pie}), (\text{Sara, ice cream}), \\
(\text{Billy, cake}), (\text{Billy, pie}), (\text{Billy, ice cream})\}
\]

\[
B \times A = \{(\text{cake, Dave}), (\text{pie, Dave}), \\
(\text{ice cream, Dave}), \\
(\text{cake, Sara}), (\text{pie, Sara}), (\text{ice cream, Sara}), \\
(\text{cake, Billy}), (\text{pie, Billy}), (\text{ice cream, Billy})\}
\]
Binary Relations

A binary relation over two sets $A$ and $B$ is a subset of:

$$A \times B$$

Example:

$$Dessert = \{ (Dave, \text{cake}), (Dave, \text{ice cream}), (Sara, \text{pie}), (Sara, \text{ice cream}) \}$$

$$Dessert^{-1} = \{ (\text{cake}, \text{Dave}), (\text{ice cream}, \text{Dave}), (\text{pie}, \text{Sara}), (\text{ice cream}, \text{Sara}) \}$$

Binary relations are particularly useful and are often written as: $x \, R \, y$
Relations

3-ary (ternary) relation: $A \times A \times A$

\{(Sara, Dave, Billy), (Beth, Mark, Cathy), (Cathy, Billy, Pete)\}

An $n$-ary relation over sets $A_1, A_2, \ldots A_n$ is a subset of:

$$A_1 \times A_2 \times \ldots \times A_n$$

- A relation may be equal to $\{\}$
  - Dave, Sue, and Billy all hate dessert
- no constraints on how many times a particular element may occur in a relation
The composition of $R_1 \subseteq A \times B$ and $R_2 \subseteq B \times C$, written $R_2 \circ R_1$, is:

$$R_2 \circ R_1 = \{(a, c) : \exists b \ ((a, b) \in R_1 \land ((b, c) \in R_2)\}$$

- $R_2 \circ R_1 \subseteq A \times C$
- $R_1 \circ R_2$ is fine, as long as consistent

$Dessert = \{(Dave, cake), (Dave, ice cream), (Sara, pie), (Sara, ice cream)\}$

$Fatgrams = \{(cake, 30), (pie, 25), (ice cream, 15)\}$

$Fatgrams \circ Dessert = \{(Dave, 30), (Dave, 15), (Sara, 25), (Sara, 15)\}$
Representing Relations

Ways to represent a binary relation \( R \):

- List the elements of \( R \)
  
  Mother-of = \{ (Doreen, Ann), (Ann, Catherine), (Catherine, Allison) \}

- Write a procedure that defines \( R \) either by:
  - Enumerating it
  - Deciding it

- Encode \( R \) as an adjacency matrix

- Encode \( R \) as a directed graph
Representing a Binary Relation as an Adjacency Matrix

<table>
<thead>
<tr>
<th></th>
<th>Doreen</th>
<th>Ann</th>
<th>Catherine</th>
<th>Allison</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doreen</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ann</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Catherine</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Allison</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
Representing Binary Relations as Graphs

(a) Doreen \rightarrow Ann \rightarrow Catherine \rightarrow Allison

(b) \( x \rightarrow y \)

(c) ice cream \rightarrow 15

pie \rightarrow 25

cake \rightarrow 30

streudel
Properties of Relations

$R \subseteq A \times A$ is reflexive iff, $\forall x \in A \ ((x, x) \in R)$

Examples:

$\leq$ defined on the integers. For every integer $x$, $x \leq x$
Properties of Relations

$R \subseteq A \times A$ is **symmetric** iff $\forall x, y ((x, y) \in R \rightarrow (y, x) \in R)$

Examples:
- $=$ is symmetric
- $\leq$ is not symmetric

(a)

(b)
Properties of Relations

$R \subseteq A \times A$ is \textit{antisymmetric} iff

$$\forall x, y ((x, y) \in R \land x \neq y \rightarrow (y, x) \notin R)$$

Examples:

$<$

$\leq$ (why?)

$R \subseteq A \times A$ is \textit{transitive} iff

$$\forall x, y, z (((x, y) \in R \land (y, z) \in R \rightarrow (x, z) \in R)$$

Examples:

$\land$

$\leq=$
Equivalence Relation

A relation $R \subseteq A \times A$ is an equivalence relation iff it is:

- reflexive,
- symmetric, and
- transitive.

Examples:

- Equality
- Lives-at-Same-Address-As
- Same-Length-As
Equivalence Classes

An equivalence relation $R$ on a set $S$ carves $S$ up into a set of clusters or islands, which we call **equivalence classes**. This set of equivalence classes has the following key property:

$$\forall s, t \in S \ ((s \in \text{class}_i \land (s, t) \in R) \rightarrow t \in \text{class}_i)$$

If $R$ is an equivalence relation on a nonempty set $A$, then the set of equivalence classes of $R$ is a partition $\Pi$ of $A$. Because $\Pi$ is a partition iff:

(a) no element of $\Pi$ is empty;
(b) all members of $\Pi$ are disjoint; and
(c) the union of all the elements of $\Pi$ equals $A$.

Example: Same-gender (R) is an equivalence relation, being reflexive, symmetric, and transitive. It carves all students (S) into two equivalence classes, $[m]$ and $[f]$, which form a partition.
To describe equivalence classes, we'll use the notation \([x]\) to mean the equivalence class to which \(x\) belongs.

In general, there are maybe lots of different ways to describe the same equivalence class.

For example, \([1]\), \([2]\), \([3]\) all describe the same equivalence class \(S\) that contains 1, 2, and 3.
Partial Order

A *partial order* is a relation that is:

- reflexive
- antisymmetric
- transitive
- Subset-of is a partial order defined on the set of all sets
- Proper-Subset-of?

Sometimes, the term “partial order” refers to ”partially ordered set,” which is a set plus a partial order relation.
Subset-of is a Partial Order

- To make the graph relatively easy to read, we follow the convention that we do not write in the links that are required by reflexivity and transitivity.
A concept is a set of entities in the world.
- Some concepts are more general than others.
- We say a concept \( x \) is subsumed by a concept \( y \) (\( x \leq y \)) iff every instance of \( x \) is also an instance of \( y \).
- Concept subsumption is a partial order.
A Subsumption Lattice

\[ \forall x (P(x)) \]

\[ P(X_1) \quad P(X_2) \]

\[ P(X_2) \lor Q(X_2) \]

\[ \forall x (R(x) \land S(x)) \]

\[ \forall x (R(x) \land T(x)) \]

\[ \forall x (R(x)) \]

\[ \text{True} \]

\[ \text{False} \]
Total Orders

A total order $R \subseteq A \times A$ is a partial order that has the additional property that:

$$\forall x, y \in A \ ((x, y) \in R \lor (y, x) \in R).$$

Example: $\leq$

Sometimes, the term “total order” refers to "totally ordered set,” which is a set plus a total order relation.
Functions

A function $f$ from a set $A$ to a set $B$ is a binary relation, subset of $A \times B$, such that:

$$\forall x \in A \ (((((x, y) \in f \land (x, z) \in f) \rightarrow y = z) \land \exists y \in B \ ((x, y) \in f))$$

- each element in $A$ relates to exactly one element in $B$.
  - one to one, many to one, but not one to many
- total

$Dessert = \{ \ (Dave, \ cake), \n  \ (Dave, \ ice \ cream), \n  \ (Sara, \ pie), \n  \ (Sara, \ ice \ cream) \}$ is not a function.

$\text{succ}(n) = n + 1$ is a function.
Notations of Functions

• $f : A \rightarrow B$
  – $A$: domain
  – $B$: codomain or range
  – $f$ is a function from $A$ to $B$

• Can define a function in two parts
  – the first specifies the domain and range
  – the second defines how elements are related

  $\text{succ}: \mathbb{Z} \rightarrow \mathbb{Z},$
  $\text{succ}(n) = n + 1.$

• succ is a unary function
• $+: (\mathbb{Z} \times \mathbb{Z}) \rightarrow \mathbb{Z}$ is a binary function
Properties of Functions

- $f : A \rightarrow B$ is a **total function** on $A$ iff it is a function that is defined on all elements of $A$.

- $f : A \rightarrow B$ is a **partial function** on $A$ iff it is a function that is defined on a subset of $A$.

- Partial functions generalizes total functions
- Partial functions are often used when the “defined domain”, is not known (e.g. many functions in computability theory)
Properties of Functions

- $f : A \rightarrow B$ is **one-to-one** iff no two elements of $A$ map to the same element of $B$.
  - injective
  - $f$ is a injection
- $f : A \rightarrow B$ is **onto** iff every element of $B$ is the value of some element of $A$.
  - surjective
  - $f$ is a surjection
  - the elements of $B$ are covered
- $f : A \rightarrow B$ is bijective if it is both injective and surjective
  - $f$ is a bijection

- the inverse of a bijection is also a function
- if $f \subseteq A \times B$, then $f^{-1} \subseteq B \times A = \{(b, a) : (a,b) \in f\}$
Properties of Functions

1
A → X
B → Y
C → Z

2
A → X
B → Y
C → Z

3
A → X
B → Y
C → Z

4
A → X
B → Y
C → Z
Q

5
A → X
B → Y
C → Z

6
A → X
B → Y
C → Z
Properties of Binary Functions

A binary function \( \# \) is **commutative** iff:
\[
\forall x, y \in A \ (x \# y = y \# x).
\]

\[
i + j = j + i. \quad \text{(integer addition)}
\]

\[
A \cap B = B \cap A. \quad \text{(set intersection)}
\]

\[
P \land Q \equiv Q \land P. \quad \text{(Boolean and)}
\]

A binary function \( \# \) is **associative** iff:
\[
\forall x, y, z \in A \ ((x \# y) \# z = x \# (y \# z)).
\]

\[
(i + j) + k = i + (j + k). \quad \text{(integer addition)}
\]

\[
(A \cap B) \cap C = A \cap (B \cap C). \quad \text{(set intersection)}
\]

\[
(P \land Q) \land R \equiv P \land (Q \land R). \quad \text{(Boolean and)}
\]

\[
(s \| t) \| w = s \| (t \| w). \quad \text{(string concatenation)}
\]
Properties of Binary Functions

A binary function \( \# \) is **idempotent** iff \( \forall x \in A \ (x \# x = x) \).

\[
\begin{align*}
\text{min}(i, i) &= i. & \text{(integer minimum)} \\
A \cap A &= A. & \text{(set intersection)} \\
P \land P &= P & \text{(Boolean and)}
\end{align*}
\]

The **distributivity** property: A function \( \# \) distributes over another function \( % \) iff:

\[
\forall x, y, z \in A \ (x \# (y \% z) = (x \# y) \% (x \# z)).
\]

\[
\begin{align*}
i \cdot (j + k) &= (i \cdot j) + (i \cdot k). & \text{(integer multiplication over addition)} \\
A \cup (B \cap C) &= (A \cup B) \cap (A \cup C). & \text{(set union over intersection)} \\
P \land (Q \lor R) &\equiv (P \land Q) \lor (P \land Q). & \text{(Boolean and over or)}
\end{align*}
\]
Properties of Binary Functions

Absorption laws also relate two binary functions to each other:

A function # absorbs another function % iff:

\[ \forall x, y \in A \ (x \# (x \% y) = x). \]

\[ A \cap (A \cup B) = A. \quad \text{(Set intersection absorbs union.)} \]

\[ P \lor (P \land Q) \equiv P. \quad \text{(Boolean or absorbs and.)} \]

\[ P \land (P \lor Q) \equiv P. \quad \text{(Boolean and absorbs or.)} \]
Identities

An element $a$ is an identity for the function $#$ iff:

$$\forall x \in A \ ((x \# a = x) \land (a \# x = x))$$

- $i \cdot 1 = i$ (1 is an identity for integer multiplication.)
- $i + 0 = i$ (0 is an identity for integer addition.)
- $A \cup \emptyset = A$ ($\emptyset$ is an identity for set union.)
- $P \lor False \equiv P$ ($False$ is an identity for Boolean or.)
- $s || "" = s$ ("" is an identity for string concatenation.)
Zeros

An element \( a \) is a zero for the function \( \# \) iff:

\[
\forall x \in A ((x \# a = a) \land (a \# x = a)).
\]

\[i \cdot 0 = 0\]  
\((0 \text{ is a zero for integer multiplication.})\)

\[A \cap \emptyset = \emptyset\]  
\((\emptyset \text{ is a zero for set intersection.})\)

\[P \land \text{False} \equiv \text{False}\]  
\((\text{False is a zero for Boolean and.})\)
Self Inverses

A unary function $ is a **self inverse** iff:

$$\forall x \ (\$(\$(x)) = x).$$

- $$-(-(i)) = i.$$  
  (Multiplying by -1 is a self inverse for integers.)

- $$1/(1/i) = i \text{ if } i \neq 0.$$  
  (Dividing into 1 is a self inverse for integers.)

- $$\neg\neg A = A.$$  
  (Complement is a self inverse for sets.)

- $$\neg(- \ P) = P.$$  
  (Negation is a self inverse for Booleans.)

- $$(s^R)^R = s.$$  
  (Reversal is a self inverse for strings.)
Properties of Functions on Sets

Commutativity: \[ A \cup B = B \cup A. \] \[ A \cap B = B \cap A. \]

Associativity: \[ (A \cup B) \cup C = A \cup (B \cup C). \] \[ (A \cap B) \cap C = A \cap (B \cap C). \]

Idempotency: \[ A \cup A = A. \] \[ A \cap A = A. \]

Distributivity: \[ A \cup (B \cap C) = (A \cup B) \cap (A \cup C). \] \[ A \cap (B \cup C) = (A \cap B) \cup (A \cap C). \]

Absorption: \[ (A \cup B) \cap A = A. \] \[ (A \cap B) \cup A = A. \]

Identity: \[ A \cup \emptyset = A. \]

Zero: \[ A \cap \emptyset = \emptyset. \]

Self Inverse: \[ \neg\neg A = A. \]
Closures under properties

A binary relation $R$ on a set $A$ is **closed under** property $P$ iff $R$ possesses $P$.

- $\leq$ on the integers, $P = \text{transitivity}$

Sometimes, if $R$ is not closed under $P$, we may want to ask what elements would have to be added to $R$ to make it closed under $P$.

The **closure** of $R$ under $P$ is a smallest set that includes $R$ and that is closed under $P$. 
Closures under properties: examples

Let $R = \{(1, 2), (2, 3), (3, 4)\}$.

The transitive closure of $R$ is:

The reflexive closure of $R$ is:
Closures under properties: examples

Let $R = \{(1, 2), (2, 3), (3, 4)\}$.

The transitive closure of $R$ is:

$\{(1, 2), (2, 3), (3, 4), (1, 3), (1, 4), (2, 4)\}$

The reflexive closure of $R$ is:

$\{(1, 2), (2, 3), (3, 4), (1, 1), (2, 2), (3, 3), (4, 4)\}$
Closures under functions

A set $A$ is closed under $f$ iff, whenever all $n$ of $f$’s arguments are elements of $A$, the value of $A$ is also in $A$.

e.g. positive integers are closed under addition

Q. Are positive integers closed under subtraction?

Let $f$ be function of $n$ arguments drawn from a set $A$. A set $A'$ is a closure of $A$ under $f$ iff:

- $A \subseteq A'$
- $A'$ is closed under $f$
- There is no smaller set $A''$ that contains $A$ and is closed under $f$
Closures under functions: examples

Is $\mathbb{N}$ closed under addition? What’s the closure of $\mathbb{N}$ under addition?

Is $\mathbb{N}$ closed under subtraction? What’s the closure of $\mathbb{N}$ under subtraction?

Is the set of even length strings of a’s and b’s closed under concatenation? Closure?

Is the set of odd length strings of a’s and b’s closed under concatenation? Closure?
Proof Techniques

- Proof by construction
- Proof by contradiction
- Proof by counterexample
- Proof by case enumeration
- Mathematical induction
- The pigeonhole principle
- Reasoning about programs
- Proving cardinality
- Diagonalization
Proof by Construction

- $\exists x(Q(x))$
- $\forall x(\exists y(P(x,y)))$

- On way to prove the assertion is to show an algorithm that finds the value that we claim must exist.
- Example: There is an infinite number of primes.
An Infinite Number of Primes

Assume that the set $P$ of prime numbers is finite. So there exists some value of $n$ such that $P = \{p_1, p_2, p_3, \ldots p_n\}$. Let:

$$q = (p_1 p_2 p_3 \ldots p_n) + 1.$$  

Since $q$ is greater than each $p_i$, it is not on the list of primes. So it must be composite and have at least one prime factor $p_k \in P$. Then $q$ must have at least one other factor, some integer $i$ such that:

$$q = ip_k.$$   

$$(p_1 p_2 p_3 \ldots p_n) + 1 = ip_k.$$  

$$(p_1 p_2 p_3 \ldots p_n) - ip_k = -1.$$  

Since $p_k$ is prime, it divides both terms on the left. Factoring it out, we get:

$$p_k(p_1 p_2 p_{k-1} p_{k+1} \ldots p_n - i) = -1.$$  

$$p_k = -1/(p_1 p_2 p_{k-1} p_{k+1} \ldots p_n - i).$$  

But, since $(p_1 p_2 p_{k-1} p_{k+1} \ldots p_n - i)$ is an integer, this means that $|p_k| < 1$. But that cannot be true since $p_k$ is prime and thus greater than 1. So $q$ is not composite. Since $q$ is greater than 1 and not composite, it must be prime, contradicting the assumption that all primes are in the set $\{p_1, p_2, p_3, \ldots p_n\}$.  

Prove by Counterexample

• One is enough

Consider the following claim:

Let $A$, $B$, and $C$ be any sets. If $A - C = A - B$ then $B = C$

We show that this claim is false with a counterexample:

Let $A = \emptyset$, $B = \{1\}$, and $C = \{2\}$

$A - C = A - B = \emptyset$

But $B \neq C$
Proof by Case Enumeration

Suppose that the postage required to mail a letter is always at least 6¢. Prove that it is possible to apply any required postage to a letter given only 2¢ and 7¢ stamps. We prove this general claim by dividing it into two cases, based on the value of \( n \), the required postage:

1. If \( n \) is even (and 6¢ or more), apply \( n/2 \) 2¢ stamps.
2. If \( n \) is odd (and 6¢ or more), then \( n \geq 7 \) and \( n-7 \geq 0 \) and is even. 7¢ can be applied with one 7¢ stamp. Apply one 7¢ stamp and \( (n-7)/2 \) 2¢ stamps.
Mathematical Induction

The principle of mathematical induction:
If: \( P(b) \) is true for some integer base case \( b \), and
For all integers \( n \geq b \), \( P(n) \rightarrow P(n+1) \)
Then: For all integers \( n \geq b \), \( P(n) \)

An induction proof has three parts:
1. A clear statement of the assertion \( P \).
2. A proof that that \( P \) holds for some base case \( b \), the smallest value with which we are concerned.
3. A proof that, for all integers \( n \geq b \), if \( P(n) \) then it is also true that \( P(n+1) \). We’ll call the claim \( P(n) \) the induction hypothesis.
The sum of the first $n$ odd positive integers is $n^2$. We first check for plausibility:

1. $(n = 1)$ $1 = 1 = 1^2$.
2. $(n = 2)$ $1 + 3 = 4 = 2^2$.
3. $(n = 3)$ $1 + 3 + 5 = 9 = 3^2$.
4. $(n = 4)$ $1 + 3 + 5 + 7 = 16 = 4^2$, and so forth.

The claim appears to be true, so we should prove it.
Sum of First $n$ Positive Integers

Let $Odd_i = 2(i - 1) + 1$ denote the $i^{th}$ odd positive integer. Then we can rewrite the claim as:

$$\forall n \geq 1 \quad (\sum_{i=1}^{n} Odd_i = n^2)$$

The proof of the claim is by induction on $n$:

Base case: take 1 as the base case. $1 = 1^2$.

Prove: $\forall n \geq 1(((\sum_{i=1}^{n} Odd_i = n^2) \rightarrow (\sum_{i=1}^{n+1} Odd_i = (n + 1)^2))$)

$$\sum_{i=1}^{n+1} Odd_i = \sum_{i=1}^{n} Odd_i + Odd_{n+1}$$

$$= n^2 + Odd_{n+1}.$$ (Induction hypothesis.)

$$= n^2 + 2n + 1.$$ (Odd$_{n+1} = 2(n+1-1) + 1 = 2n + 1.$)

$$= (n + 1)^2.$$
Pigeonhole Principle

Suppose that we have n pigeons and k holes. Each pigeon must fly into a hole. If n > k, then there must be at least one hole that contains more than one pigeon.

More formally,

Consider any function \( f: A \rightarrow B \).

The *pigeonhole principle* says:

If \(|A| > |B|\) then \( f \) is not one-to-one.
Reasoning About Programs

- Correctness properties, including:
  - The program eventually halts.
  - When it halts, it has produced the desired output.

- Performance properties, including:
  - Time requirements, and
  - Space requirements.
To prove termination of a program $P$ with a loop, we will generally rely on the existence of some well-founded set $(S, R)$ such that:

- There exists some bijection between each step of $P$ and some element of the set $S$,
- The first step of $P$ corresponds to a maximal (with respect to $R$) element of $S$,
- Each successive step of $P$ corresponds to a smaller (with respect to $R$) element of $S$, and
- $P$ halts on or before it executes a step that corresponds to a minimal (with respect to $R$) element of $S$. 
Choosing a Well-Founded Set

\[ P(s: \text{string}) = \]

\begin{verbatim}
While length(s) > 0 do:
    Remove the first character from \( s \) and call it \( c \).
    if \( c = a \) return \text{True}.
Return \text{False}.
\end{verbatim}

Let \( S = \{0, 1, 2, \ldots, |s|\} \). \((S, \leq)\) is a well-founded set whose least element is 0. Associate each step of the loop with \(|s|\) as the step is about to be executed. The first pass through the loop is associated the initial length of \( s \), which is the maximum value of \(|s|\) throughout the computation. \(|s|\) is decremented by one each time through the loop. \( P \) halts when \(|s|\) is 0 or before (if it finds the character a). So the maximum number of times the loop can be executed is the initial value of \(|s|\).
Proving a Program Computes Correct Result

- Loop Invariants: a predicate $I$ that describes a property that does not change during the execution of an iterative process.

- Induction
Loop Invariants

To use a loop invariant $I$, we must prove:

- $I$ is true on entry to the loop.

- The truth of $I$ is maintained at each pass through the loop.

- $I$, together with the loop termination condition, imply whatever property we wish to prove is true on exit from the loop.
The Coffee Can Problem

Given a coffee can that contains some white beans and some black beans, do:

Until no further beans can be removed do:

1. Randomly choose two beans.
2. If the two beans are the same color, throw both of them away and add a new black bean.
3. If the two beans are different colors, throw away the black one and return the white one to the can.

• Would this process always halt?
• What color is the remaining bean?
Cardinality

We will be concerned with three cases:
• finite sets,
• countably infinite sets, and
• uncountably infinite sets.

A set $A$ is **finite** and has cardinality $n \in \mathbb{N}$ iff either:

• $A = \emptyset$, or
• there is a bijection from $\{1, 2, \ldots, n\}$ to $A$, for some $n$.

A set is **infinite** iff it is not finite.
Countably Infinite Sets

\( \mathbb{N} \) is countably infinite. Call its cardinality \( \aleph_0 \)

A is \textit{countably infinite} and also has cardinality \( \aleph_0 \) iff there exists some bijection \( f : \mathbb{N} \to A \)

A set is \textit{countable} iff it is either finite or countably infinite.

To prove that a set \( A \) is countably infinite, it suffices to find a bijection from \( \mathbb{N} \) to it.
Even Numbers

The set $E$ of even natural numbers is countably infinite.

To prove this, we offer the bijection:

$Even : \mathbb{N} \rightarrow E,$

$Even(x) = 2x.$

<table>
<thead>
<tr>
<th>$\mathbb{N}$</th>
<th>$E$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>$\ldots$</td>
<td>$\ldots$</td>
</tr>
</tbody>
</table>
Enumerations

Sometimes, bijection is not as obvious. Then a good way to think about the problem of finding a bijection from $\mathbb{N}$ to some set $A$, is to turn it into the problem of finding an enumeration of $A$.

An *enumeration* of a set $A$ is simply a list of the elements of $A$ in some order. Each element of $A$ must occur in the enumeration exactly once.

Example: an enumeration for $\mathbb{Z}$?
Enumerating Countably Infinite Sets

**Theorem:** A set $A$ is countably infinite iff there exists an infinite enumeration of it.

**Proof:** We prove the if and only-if parts separately.

*If $A$ is countably infinite, then there exists an infinite enumeration of it:* Since $A$ is countably infinite, there exists a bijection $f$ from $\mathbb{N}$ to it. We construct an infinite enumeration of $A$ as follows: For all $i \geq 1$, the $i^{th}$ element of the enumeration of $A$ will be $f(i - 1)$.

*If there exists an infinite enumeration $E$ of $A$, then $A$ is countably infinite:* Define $f: \mathbb{N} \rightarrow A$, where $f(i)$ is the $(i+1)^{st}$ element of the list $E$. The function $f$ is a bijection from $\mathbb{N}$ to $A$, so $A$ is countably infinite.

**Note:** used to show “countably infinite”, as well as “not countably infinite”
Finite Union

**Theorem:** The union $U$ of a finite number of countably infinite sets is countably infinite.

**Proof:** by enumeration of the elements of $U$:

$S_1[1], S_2[1], \ldots S_n[1],$
$S_1[2], S_2[2], \ldots S_n[2],$

... checking before inserting each element to make sure that it is not already there.

Can we do this? Enumerating all the elements of the 1st set, then the 2nd ...
Countably Infinite Union

**Theorem:** The union \( U \) of a countably infinite number of countably infinite sets is countably infinite.

**Proof:** by enumeration of the elements of \( U \).

<table>
<thead>
<tr>
<th></th>
<th>Set 1</th>
<th>Set 2</th>
<th>Set 3</th>
<th>Set 4</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element 1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Element 2</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Element 3</td>
<td>6</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Note: use the previous simple enumeration technique, we’d never get to the 2\(^{nd}\) element of any sets …
- check make sure the element not already there
Diagonalization

• Cantor's diagonal argument
• 1891 by Georg Cantor, there are infinite sets that are more numerous than natural numbers
  • His first proof about “real numbers are more numerous than natural numbers” appeared in 1874
  • but this is a more powerful and general technique
• Used in a wide range of proofs: chain of influence
  • Russell’s paradox
  • Godel’s incompleteness theorems
  • Turing's answer to the Entscheidungsproblem
  • Turing’s undecidability of the Halting problem

Georg Cantor (1845 – 1918)
• German mathematician
• Creator of modern set theory, introduced infinity in set theory
• Diagonal argument
Russell’s Paradox

• By Bertrand Russell in 1901
• Showed that the naive set theory leads to a contradiction.
• A set exists whose members are those objects (and only those objects) that satisfy the criterion; but this assumption is disproved by:

A set containing exactly the sets that are not members of themselves

• Applied version: there are some versions of this paradox that are closer to real-life situations and may be easier to understand for non-logicians.
• Barber paradox: A barber shaves all and only those men in town who do not shave themselves.
  • construct a set containing men that the barber shaves. whether the barber himself should be in the set?

• Applications:
  • By Kurt Gödel, in incompleteness theorem by formalizing the paradox
  • By Turing, in undecidability of the Halting problem (and with that the Entscheidungsproblem) by using the same trick
Russell’s Paradox

In the middle of the night I got such a fright that woke me with a start,
For I dreamed of a set that contained itself, \textit{in toto}, not in part.
If sets can thus contain themselves, then they might also fail
To hold themselves as members, and this leads me to my tale.

Now Frege thought he finally had the world inside a box,
So he wrote a lengthy tome, but up popped paradox.
Russell asked, ``You know that Epimenides said oft
A Cretan who tells a lie does tell the truth, \textit{nicht war, dumkopf}?''

And here's a poser you must face if continue thus you do,
What make you of the following thought, tell me, do tell true.
The set of all sets that contain themselves might cause a soul to frown,
But the set of all sets that \textit{don't} contain themselves will bring you down!''

Now Gottlob Frege was no fool, he knew his proof was fried.
He published his tome, but in defeat, while in his beer he cried.
And Bertrand Russell told about, in books upon our shelves,
The barber in town who shaves all those who do not shave themselves.
Bertrand Russell (1872 – 1970)
- English philosopher, logician, mathematician …
- In 1950, awarded Nobel in literature
- Russell’s family, one of the most influential in England, politically, land-owning, Earl, Duke, Baron, prime minister
- Has a (even more) famous student, Ludwig Wittgenstein
- The two are widely referred to as the greatest philosophers of last century

Ludwig Wittgenstein (1889 – 1951)
- Wittgenstein family, one of the richest in the world
- Famous schoolmate: Adolf Hitler (6 days older, 2 grades below), were 14 or 15 years old.
- Some say they never knew each other. Some say the interaction/competition/conflict/jealousy generated hatred toward Jews, thus changed the history
- Alan Turing attended his lectures at Cambridge after coming back from Princeton (with Alonzo Church)
Diagonalization

• \(\mathbb{N}\) is countably infinite. Call its cardinality \(\aleph_0\).
• There are sets with more than \(\aleph_0\) elements (not countably infinite)
  • e.g., set of real numbers
• The power set of the integers is not countable.
• \(S\) is a countably infinite set, \(P(S)\) is not countable.

• To prove it, use diagonalization.
• Proof by contradiction.
• To show a set \(A\) is not countably infinite, we assume that it is, in which case there would be some enumeration of it. Every element of \(A\) would have to be on that list somewhere.
• But we show how to construct an element of \(A\) that cannot be on the list, no matter how the list was constructed.
• Thus, there exists no enumeration of \(A\), so \(A\) is not countably infinite.
### Diagonalization

$s$ is a countably infinite set, $P(s)$ is not countable.

<table>
<thead>
<tr>
<th></th>
<th>Elem 1 of $S$</th>
<th>Elem 2 of $S$</th>
<th>Elem 3 of $S$</th>
<th>Elem 4 of $S$</th>
<th>Elem 5 of $S$</th>
<th>………</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elem 1 of $P(S)$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>………</td>
</tr>
<tr>
<td>Elem 2 of $P(S)$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>………</td>
</tr>
<tr>
<td>Elem 3 of $P(S)$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>………</td>
</tr>
<tr>
<td>Elem 4 of $P(S)$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>………</td>
</tr>
<tr>
<td>Elem 5 of $P(S)$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>………</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>………</td>
</tr>
</tbody>
</table>

A set that is not in the table:

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>………</th>
</tr>
</thead>
</table>