Induction

Algorithm correctness
- Precondition: predicates describing the initial state (involving the input variables)
- Postcondition: predicates describing the final state (involving the input and output variables)

Example:
- Product(int m, double r) // precondition: m a non-negative integer, r: a real
  // postcondition: returns m * r
- Sort(int[] y) // precondition: y is an array of integers
  // postcondition: y is a permutation of the original array such that
  // y[i] ≤ y[i+1] for 0 ≤ i < y.length

Loop Invariant

Format:
- /Loop precondition:
  Loop
  // Loop invariant:
  exit when ExitCondition
  Loop Contents
  End
  // Loop postcondition:

To Prove:
- loop precondition → Loop invariant
- Loop Invariant (at one iteration) ∧ Loop contents ∧ !ExitCondition → Loop Invariant (at next iteration)
- ExitCondition is eventually true
- Loop invariant ∧ ExitCondition → Loop postcondition
Loop Invariant

double total = 0;
int i = 0;
// loop precondition: i = 0, m: non-negative integer: r is real, total=0
loop
  // loop invariant: total = i * r
  exit when i = m
  total = total + r;
  i = i + 1;
end
// loop postcondition: total = m * r

Using Precondition/Postcondition with recursive algorithms

// precondition A is an array of elements of length ≥1
Min(A[]) {
  If length(A) =1
    return A[1]
  potentialMin = Min(A[2..n])
  return lower of potentialMin and A[1]
}
// postcondition: return result is ≤ all elements of A

Set Theory

Set: Contains zero or more elements, repetitions ignored, order doesn’t matter
- Example with no elements – {} (alternative syntax: Ø)
- Example with three elements – {john, sally, harry} = {harry, sally, john} = {harry sally, harry, john}
- Example with infinite elements – {x: x is a positive integer}
- What about:
  - Ø
Ordered set: Contains zero or more elements, repetitions not allowed, order does matter (not often used)
- (john, sally, harry) ≠ (sally, harry, john)
Ordered list: Contains zero or more elements, repetitions allowed, order does matter (used often)
- Ordered 4-tuple
  - (john, sally, harry, sally)
- Ordered pair
  - (x, y)
- Ordered triple
  - (x, y, z)
Set notation

- \( x \in S \): \( x \) is an element of \( S \)
- \( S \subseteq T \): \( S \) is a subset of \( T \); \( \forall x, x \in S \implies x \in T \)
- \( S \subset T \): \( S \) is a proper subset of \( T \); \( S \subseteq T \land (\exists x, x \in T \land x \notin S) \)
- \( S = T \): \( S \) is equal to \( T \); \( S \subseteq T \land T \subseteq S \)
- \( |S| \): The cardinality (size) of \( S \); the number of elements of \( S \)
- \( k \)-set
  - A set with cardinality \( k \)
- \( \{x : x > 1\} \) (or \( \{x | x > 1\} \))
  - Defines the set by stating the properties shared by the elements

Set Operations

- \( S \cup T \): Union; \( \{x : x \in S \lor x \in T\} \)
- \( S \cap T \): Intersection; \( \{x : x \in S \land x \in T\} \)
- \( S - T \) (or \( S \setminus T \)): Difference; \( \{x : x \in S \land x \notin T\} \)

Often we have some universe \( U \), in mind

- \( S^C \) (or \( \sim S \)): Complement of \( S \); \( U - S \)
- \( S \oplus T \): Symmetric difference; \( (S - T) \cup (T - S) \)
- \( S \times T \)
  - (Cartesian) product; \( \{(x, y) : x \in S \land y \in T\} \)
- \( S \times T \times U \)
  - (Cartesian) product; \( \{(x, y, z) : x \in S \land y \in T \land z \in U\} \)

Venn Diagrams

- Rectangle is universe
- Circle (or ellipse) for each set
- Overlap represents intersection

Example:

- \( A \subseteq B \)
  - \( A \cap B \)
  - \( A \cap B^C \)

Proof Using Venn Diagrams

- Proving \( \text{LHS} = \text{RHS} \) (or \( \text{LHS} \neq \text{RHS} \))
  - Create general Venn diagram for given sets
  - (optional) Number each region
    - (Given \( k \) sets, how many regions?)
  - Shade in regions for LHS
  - Shade in regions for RHS
  - Compare

Warning: has limitations
Example Proof

Prove \((A \cap B)^C = A^C \cup B^C\)

Proof Using Elements

To prove LHS=RHS
- Prove LHS \subseteq RHS
- Prove RHS \subseteq LHS

To prove LHS \subseteq RHS
- If \(x \in \text{LHS}\), then ...
- Thus, \(x \in \text{RHS}\)
- This shows that LHS \subseteq RHS

Example: prove \((A \cap B)^C = A^C \cup B^C\)
- If \(x \in (A \cap B)^C\), then \(x \notin (A \cap B)\)
- Thus, \(x \in A^C\) or \(x \in B^C\)
- So, \(x \in A^C\) or \(x \in B^C\)
- Hence, \(x \in A^C\) or \(x \in B^C\)
- This shows that \((A \cap B)^C \subseteq A^C \cup B^C\)

Algebraic Rules for Sets

<table>
<thead>
<tr>
<th>Property</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Associative</td>
<td>(A \cup (B \cup C) = (A \cup B) \cup C)</td>
</tr>
<tr>
<td>Distributive</td>
<td>(A \cup (B \cap C) = (A \cup B) \cap (A \cup C))</td>
</tr>
<tr>
<td>DeMorgan's</td>
<td>((A \cup B)^C = A^C \cap B^C), ((A \cap B)^C = A^C \cup B^C)</td>
</tr>
<tr>
<td>Idempotent</td>
<td>(A \cup A = A), (A \cap A = A)</td>
</tr>
<tr>
<td>Double</td>
<td>((A^C)^C) = A</td>
</tr>
<tr>
<td>Negation</td>
<td>(A \cup (A \cap B) = A), (A \cap (A \cup B) = A)</td>
</tr>
<tr>
<td>Absorption</td>
<td>(A \cup (A \cap B) = A), (A \cap (A \cup B) = A)</td>
</tr>
<tr>
<td>Commutative</td>
<td>(A \cup B = B \cup A), (A \cap B = B \cap A)</td>
</tr>
<tr>
<td>Bound</td>
<td>(A \cup \emptyset = A), (A \cap \emptyset = \emptyset)</td>
</tr>
<tr>
<td>Negation</td>
<td>(A \cup A^C = U), (A \cap A^C = \emptyset)</td>
</tr>
</tbody>
</table>

Proof Using Algebraic Rules

\[ A = (B^C \cup A) \cap (A \cup B) \]
### Tabular Method

A similar to truth tables

\[
A = (B^c \cup A) \cap (A \cup B)
\]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>(B^c)</th>
<th>(B^c \cup A)</th>
<th>(A \cup B)</th>
<th>((B^c \cup A) \cap (A \cup B))</th>
</tr>
</thead>
<tbody>
<tr>
<td>⊤</td>
<td>⊤</td>
<td>⊤</td>
<td>⊤</td>
<td>⊤</td>
<td>⊤</td>
</tr>
<tr>
<td>⊤</td>
<td>⊥</td>
<td>⊤</td>
<td>⊤</td>
<td>⊤</td>
<td>⊤</td>
</tr>
<tr>
<td>⊥</td>
<td>⊤</td>
<td>⊤</td>
<td>⊤</td>
<td>⊤</td>
<td>⊤</td>
</tr>
<tr>
<td>⊥</td>
<td>⊥</td>
<td>⊤</td>
<td>⊤</td>
<td>⊤</td>
<td>⊤</td>
</tr>
</tbody>
</table>

### Tabular Method vs. Venn Diagram

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Using Venn diagrams for counting

100 Students
- 10 don’t take math or science
- 60 take math
- 70 take science
- How many take both

### Counting the subsets of a set

The **powerset** of a set is the set of all subsets

- If \(S = \{a, b, c\}\), the powerset is:

If S is a finite set with n elements, how many elements in the powerset?
Russell’s Paradox
There’s a town where the barber shaves a person if and only if the person doesn’t shave himself

- Who shaves the barber?

Halting Problem
Assume there exists routine:

- Boolean Halt(String program, String input) (returns true if program executed on input halts, false otherwise)
- We can write:
  Boolean Trouble(String program) {
    if Halt(program, program)
      loop forever
    else
      stop
  }

Partition
Given a set S = \{a_1, a_2, … a_n\}:
- A partition P is a set of subsets of S such that:
  - the elements of P cover S
  - The elements of P are pairwise disjoint
  - No element of P is empty
  - The elements of P are called the blocks of the partition

Number of Partitions
Given S = \{a_1, a_2, ..., a_n\}:
- How many partitions of S are there?
- How many partitions of size k are there?
  - If a_n is in a new block:
  - S(n, k) is the Stirling number of the second kind
  - S(n, k) =
Calculating $S(n, k)$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>