Lecture 4: Algorithm Correctness and Properties of Sets

Algorithm Correctness
(or how to prove programs are correct)

Pre-condition of the algorithm: a predicate that describes the initial state (before execution)

Post-condition of the algorithm: a predicate that describes the final state (after execution)

The algorithm is correct if it can be proved that if the pre-condition is true, the post-condition must be true.

Pre-condition $\Rightarrow$ Post-condition

ex 1:

program mystery
begin
    Pre-condition: $a = 3 \quad b = 5$
    .
    .
    end
    Post-condition: $a = 5 \quad b = 3$

ex 2:

$x = 2$
$z = x + y$
if ( $y > 0$ )
    $z = z + 1$
else
    $z = 0$

Pre-condition: $y = 6$
Post-condition: $z = 9$
Programs may be sliced into sequential parts and assertions inserted

program mystery2
begin
    Segment 1
    Segment 2
    Segment 3
    .
    .
    Segment n
end

What about loops?
Consider a while loop with Pre-Condition and Post-Condition and guard G.

To prove that the loop is “correct” do the following:

1. Write a loop invariant that “captures” the purpose of the loop; call it I(n). (Difficult!!) I(n) is a predicate that is true after n iterations of the loop.

2. Show that the pre-condition implies the loop invariant I(0). (Basis Step)

3. Show that if I(k) and guard G are true, then I(k+1) is true. (Inductive Step)
   In other words, if I(n) is true for n = k and the guard G is still true, then it follows that I(n) is true for n = k+1.

4. Show that guard G eventually becomes false (looping stops).

5. If the loop iterates a maximum of N times, show that I(N) implies the post-condition.
Example 1  
**Pre-condition:** m is a non-negative integer, x is real, Count = 0, Product = 0  
while ( Count != m)  
{  
    Product = Product + x;  
    Count = Count + 1;  
}  
**Post-condition:** Count = m and Product = m \cdot x

1. Write loop invariant I(n): (Remember that n represents the number of times the loop has iterated)

2. Show that the pre-condition implies that I(0) is true.

3. **Inductive Step:** Show that if I(k) and G are true, I(k+1) is true.  
   Assume G: and I(k): Product = k \cdot x and Count = k are true. (these are Product\text{old} and Count\text{old})  
   Show I(k+1): Product = (k+1) \cdot x and Count = k + 1 is true. (these are Product\text{new} and Count\text{new})

4. Show that G will eventually be false.

5. Show that the post-condition follows when the loop stops.
Example 2.

**Pre-condition:** Max = A[0]; Count = 0

while ( Count < 10)
{
 Count = Count + 1;
 if ( A[Count] > Max )
 Max = A[Count];
}

**Post-condition:** Max contains the largest value in A[0]..A[10]; Count = 10

1) Write loop invariant I(n):

2) Show that the pre-condition implies that I(0) is true.

3) Show I(k) and G imply I(k+1).

   I(k):
   I(k+1):

4) Show that G will eventually be false.

5) Show the post-condition follows when the loop stops
Properties of Sets

Axiom of Extension: A set is completely determined by its elements.

- Elements in a set are NOT ordered.
- All sets contain elements belonging to some "universal" set.
- Sets can be defined by rule or enumerated by listing all elements (or "enough" elements to exhibit a pattern and using an ellipsis ...).

<table>
<thead>
<tr>
<th>Set notation:</th>
<th>meaning</th>
<th>figure (Venn Diagram)</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x ∈ A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A ⊆ B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A ⊇ B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A ∪ B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A ∩ B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A – B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A^c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>∅</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Cartesian Product

A × B = \{ (a, b) \mid a ∈ A and b ∈ B \} (that set of all ordered pairs (a, b)) also defined for > 2 sets

ex. A = \{1, 2\} B = \{3, 4, 5\}

Is (A × B) × C = A × (B × C)? (Try it with small sample sets, say A = \{1, 2\}, B = \{3\}, C = \{7\})
**Partition** of set $A$: a collection of disjoint, non-empty subsets of $A$ whose union is $A$.

ex. If $A = \{1, 2, 3, 4, 5, 6\}$ find a partition of $A$.

\[
\begin{align*}
\text{Part}_1 &= \{ \} \\
\text{Part}_2 &= \{ \} \\
\text{Part}_3 &= \{ \}
\end{align*}
\]

ex. $Z = \{ \ldots, -2, -1, 0, 1, 2, \ldots \}$ Is $\{\{0\}, Z^+, Z^-\}$ a partition of $Z$?

??How many partitions of set $A$ are possible?

Suppose $A = \{2, 7, 9\}$. What are the possible partitions?

---

The **power set** of set $A$, $\mathcal{P}(A)$, is the set of all subsets of $A$.

ex. Find $\mathcal{P}(A)$ if $A = \{1, 2, 3, 4\}$.

<table>
<thead>
<tr>
<th>Cardinality of $A$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>…</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cardinality of $\mathcal{P}(A)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If $A$ has $n$ elements, $\mathcal{P}(A)$ has

ex: If $B = \{0, 1, 2\}$, how many elements are in $\mathcal{P}(B)$? What are they?
Formal Languages

\[ \Sigma = \text{set of symbols} = \text{alphabet} \]

a string over \( \Sigma \) = an ordered n-tuple of elements of \( \Sigma \) or \( \epsilon \), the null string

A formal language is a set of strings

\[ \Sigma^n = \]

\[ \Sigma^* = \text{set of all strings (over } \Sigma \text{ ) of finite length (Kleene star)} \]

ex. Let \( \Sigma = \{ a, b \} \)

\[ \Sigma^3 = \]

\[ \Sigma^* = \]

ex. Let language \( L \) be the set of all strings of length 2 or less over \( \Sigma = \{ x, y \} \)

Back to Sets: Demonstrate with a Venn diagram that

1) \( ( A \cap B ) \cup ( A \cap C ) = A \cap ( B \cup C ) \)  
2) \( ( A \cup B ) \cap ( A \cup C ) = A \cup ( B \cap C ) \)

Computer Implementation of Sets

Does the Java JDK have a Set?

Could we implement a set? How?

Suppose \( U = \{ 1, 2, 3, 4, 5, 6, 7, 8 \} \). How could we represent subsets of \( U \)? Say \( A = \{ 2, 3, 6 \} \), \( B = \{ 1, 2, 7 \} \)? Could we easily find \( A \cap B \) or \( A \cup B \)?