Constructions by induction

Lecture 15
ICOM 4075
What is induction?

Intuitively speaking, *induction* is a procedure for “constructing the next element out of the previous one(s)”. As such, induction involves:

1. A sense of precedence: What is the next element? (this is an often implicit but nonetheless, necessary condition)

2. An inductive rule consisting of a starting element or *Basis* and a rule or *Induction* (Basis and induction are always made explicit)
The sense of order

For an induction to be possible, it is necessary to have a \textit{one-to-one and onto} mapping

$$f : \mathbb{N} \rightarrow A$$

where $\mathbb{N}$ is an infinite subset of $\text{Nat}$, the set of natural numbers, and $A$ is the set of objects to be constructed by induction. Thus,

a) The set $A = \{f(n) : n \in \mathbb{N}\}$, and within $A$:

b) $f(n+1)$ is said to be \textbf{successor} of $f(n)$

\textbf{Remark}: There may be many different one-to-one and onto mappings from $\mathbb{N}$ into $A$, each one of them defines a different successor for each element in $A$. 
**Illustration of successor**

Consider the language over the alphabet \{a, b\}

\[ A = \{\lambda, ab, abab, ababab, \ldots \} \]

and the mapping:

\[ f : \text{Nat} \rightarrow A \]

\[ f(0) = \lambda, f(n) = ab \ldots ab \]

\[ n \text{ times} \]

**Under this mapping** \[ f(1) = ab, f(2) = abab, f(3) = ababab, \text{ etc...} \]

and thus,

\[ abab \text{ is the successor of } ab \]

\[ ababab \text{ is the successor of } abab \]

\[ abababab \text{ is the successor of } ababab, \text{ etc...} \]
The induction

As pointed out earlier, a constructor consists of a starting element, called basis, and a relation between one or more of the previous objects in A called induction.

This relation pairs a fixed number or predecessors with their successor.
Consider again the language over the alphabet \( \{a, b\} \)

\[
A = \{\lambda, ab, abab, ababab, \ldots \}
\]

This language can be defined by induction with the following rule:

**Basis:** \( \lambda \) is in \( A \)

**Induction:** If \( x \) is in \( A \) then \( xab \) is in \( A \)

Using the rule:

- \( ab \) is in \( A \) since \( x = \lambda \) is in \( A \) and \( xab = ab \)
- \( abab \) is in \( A \) since \( x = ab \) is in \( A \) and \( xab = abab \)
- \( ababab \) is in \( A \) since \( x = abab \) is in \( A \) and \( xabab = ababab \)
Computational construction of elements in a inductive set

**Definition**: If the objects in a set can be constructed by induction, the set is said to be **inductive**

Following is a procedure for constructing a string of length $2n$ in the previous language, without using intermediate storage:

$$\text{StringOfLength}(n): y = \lambda$$

While $|y| < 2n$

$$y \leftarrow yab$$

Return $y$

A similar procedure can be used to construct strings of any length, in any inductive language.
Other examples of inductive sets

Example 1: \( A = \{3, 5, 7, 9, \ldots \} \) is inductive. In fact, \( A \) satisfies the pre-condition with, for instance, the mapping

\[
f: \{0, 1, 2, \ldots \} \to \{3, 5, 7, \ldots \}
\]

\[
f(n) = 2n + 3
\]

As for the constructor, we observe that the relation between an object \( a \) and its successor is \((a, a + 2)\). Also, the first element is 3. So, we get

1. **Basis**: first element is 3 in \( A \)
2. **Induction**: (expressed as a logical sentence): If \( a \) is in \( A \), then \( a + 2 \) is in \( A \)
More examples of inductive sets

In the previous example the mapping was shown only as a way to verify the pre-condition for induction. There are instances, however, in which the function itself is part of the inductive rule. Next is an example.

**Definition**: The Fibonacci numbers are defined as

1. **Basis**: The first number is 0, the second number is 1
2. **Induction**: Define $F : \text{Nat} \rightarrow \text{Fibonacci Numbers}$ by setting $F(0) = 0$, $F(1) = 1$ and the relation $F(n) = F(n-1) + F(n-2)$, $n \geq 2$
Constructing a Fibonacci number by induction

The next pseudo code constructs a Fibonacci number inductively

Inductive_Fibo(n)
    If n = 0 return 0
    If n = 1 return 1
    U ← 1
    V ← 0
    Z ← 1
    For j = 2 to n
        V ← U
        U ← Z
        Z ← U + V
    Return Z
A non-inductive set

Many sets cannot be described by an induction. Among them, a chief example is the set of real numbers. The problem with real numbers is in the impossibility of identifying a successor. In order to see that, ask yourself: What is the successor of 0? What is the real number immediately to the right of 0?

Indeed, the impossibility to answer this question has to do with the fact that there is no one-to-one and onto mapping from any subset of the naturals into the real numbers.
Observation

Despite the fact that the set of real numbers is not an inductive set, the following “inductive rule” is still true:

“If x is real and y is real then xy is real”

What is missing in this case is the above mentioned one – to – one mapping with the set of naturals (that’s why is important to make sure that such a map exists)
Inductive languages

Recall that a formal language is a set of strings over an alphabet. A language is said to be inductive if it is inductive as a set.

Example: The language \( \{0, 1\}^* \) is inductive.

1. **Basis**: \( \lambda, 0, 1 \) are in \( \{0, 1\}^* \)

2. **Inductive rule**: If \( s \) is in \( \{0, 1\}^* \) and \( a \) is in \( \{0, 1\} \), then \( as \) is in \( \{0, 1\}^* \)
Using the inductive construction of \( \{0, 1\}^* \)

A first question that may arise regarding the inductive production of a string in \( \{0, 1\}^* \) is:

**What is the mapping \( f: \mathbb{N} \to \{0, 1\}^* \)?**

As in the other cases, there is more than one such mapping. Among them, the most common is the mapping based on the **lexicographic** order of \( \{0, 1\}^* \):

\[
\begin{align*}
f(0) &= \lambda, \\
f(1) &= 0, \\
f(2) &= 1, \\
f(3) &= 00, \\
f(4) &= 01, \\
f(5) &= 10, \\
f(6) &= 11, \\
f(7) &= 000, \\
f(8) &= 001, \\
f(9) &= 010, \\
&\text{etc...}
\end{align*}
\]

As it can be inferred from a close look to this mapping, the inductive construction of a string in \( \{0, 1\}^* \) will be much involved than the previous constructions.
Given an inductive definition find the language

A common problem is: Given a definition by induction, find the set – theoretical description of the corresponding language
Given an inductive definition find the language

A common problem is: Given a definition by induction, find the set – theoretical description of the corresponding language.

Example:

1. **Basis**: a is in the language

2. **Inductive Rule**: if x is in the language, then xb is in the language

What is the language?
Given an inductive definition find the language

A common problem is: Given a definition by induction, find the set – theoretical description of the corresponding language.

Example:

1. **Basis**: a is in the language
2. **Inductive Rule**: if x is in the language, then xb is in the language
   
   **What is the language?**

   **Answer**: the language is \{ab^n : n \text{ natural}\}
And the opposite problem:

Given the set theoretical description of a language, is there an inductive definition?

**Example**: Is $L = \{a^n b^n : n \text{ natural}\}$ inductive?

**Answer**:

Yes. Here is an inductive construction:

**Basis**: $\lambda$ is in $L$

**Inductive rule**: If $x$ is in $L$, then $axb$ is in $L$
And the opposite problem:

Given the set theoretical description of a language, is there an inductive definition?

**Example:** Is \( L = \{a^n b^n : n \text{ natural} \} \) inductive?

**Answer:**

Yes. Here is an inductive construction:

**Basis:** \( \lambda \) is in \( L \)

**Inductive rule:** If \( x \) is in \( L \), then \( axb \) is in \( L \)
Well-defined Boolean formulas

Recall that a Boolean formula is an expression consisting of Boolean variables and conjunctions, disjunctions and/or negations.

Some Boolean formulas are not well-defined. For example: x and or not y

How can well-defined Boolean formulas be characterized? Here is a first attempt:

**Definition:** A Boolean formula is well-defined if it corresponds to an actual logical sentence.
Well-defined Boolean formulas (cont.)

If we were to use the previous definition to implement a program that recognizes whether a Boolean formula is or is not well-defined, we will encounter at least one major problem:

How to characterize an “actual logical sentence”
Well-defined Boolean formulas (cont.)

The next inductive definition eliminates this problem

**Definition**: Let $V$ be a finite set of Boolean variables. Consider the alphabet $A = V \cup \{\text{and, or, not}\}$. Then, the following strings over $A$ are well-defined Boolean expressions:

- **a.** Each $x$ in $V$
- **b.** $A$ and $B$, where $A, B$ are well-defined Boolean expressions
- **c.** $A$ or $B$, where $A, B$ are well-defined Boolean expressions
- **d.** not $A$, where $A$ is a well-defined Boolean expression
- **e.** No other string over $A$ is a well-defined Boolean expression
Summary

• Definition of induction
• Order, successors
• Constructors: Basis and Induction
• Inductive sets
• Non-inductive sets
• Inductive languages
• Lexicographic successor
• Well-defined Boolean formulas
Exercises

1. Find an inductive constructor for each of the following languages:
   a) \{a^nbc^n: n \text{ natural}\}
   b) \{s: s \text{ is a string over \{a, b\} with the same number of a’s and b’s}\}

2. Devise an inductive procedure (pseudo-code) for constructing any given string for each of the languages in 1.

3. Find an inductive constructor for each of the following sets:
   a) \{x \text{ natural: } \text{ceil}(x/2) \text{ is even}\}
   b) \{x \text{ natural: } x \mod 6 = 1\}
   c) \{x: x \text{ is a string over \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} }\}
4. Define a one-to-one and onto mapping
   \( f: \text{Nat} \to \{ s: s \text{ string over \{a, b\} with the same number of a's and b's}\} \) and use it to define successors (hint: use lexicographic order)

4. Use induction to define the concept of well-defined algebraic expression as a string over \{x, y, z, +, -, /, , ( , ) \}

5. Apply your definition to decide whether the following strings over the alphabet \{x, y, z, +, -, /, , ( , ) \} are well-defined
   a) \(((x + y)z +)y\)
   b) \((z-x)(z-y)+(x-(x-y))\)
   c) \(-(x-y-z)+((xy)/z)\)
Exercises

5. Find the following Fibonacci numbers: F(4), F(6), F(11), F(14)

6. Using the lexicographic successor define inductively a mapping

   \[ f : \text{Nat} \rightarrow A^* \]

   where \( A = \{x, 1, \&, \#\} \)

7. Find the lexicographic successor of each of the following strings over the alphabet \( \{x, 1, \&, \#\} \)

   a) \&x1
   b) \&x&
   c) ######
   d) &1x1&#
   e) &1x#####