Recursive definition of L^*, L is a language over some alphabet Σ.

Ver1

1. $\Lambda \in L^*$

2. For any $x \in L^*$ and $y \in L$,

   $xy \in L^*$

3. No string is in L^* unless it can be obtained by using rules 1 and 2.

Example

Let $L = \{a, ab\}$

- Using rule 1, $\Lambda \in L^*$.

- One application of Rule 2, we have

  $\Lambda a = a$
  $\Lambda ab = ab$

  i.e. $L^1 = \{\Lambda, a, ab\}$

- Second application of Rule 2, we have L^2.

  $L^2 = \{\Lambda, a, ab, aab, aba, aa, abab\}$

In same way, a string obtained by concatenating K elements of L, can be obtained by K applications

Ver 2

1. $\Lambda \in L^*$

2. For any $x \in L$, $x \in L^*$

3. For any two elements x and y of L^*, $xy \in L^*$

4. No string is in L^* unless it can be obtained by using rules 1, 2 and 3.

Both of these definitions are equal and define L^*.

Recursive Definition — Fully parenthesized Algebraic Expressions

- Let Σ be the alphabet $\{ i, (,), +, - \}$
- Fully Parenthesized means exactly one pair of parenthesis for every Operator

Follows the definition of AE involving binary operators "+" and "−" and the identifier "i".

1. $i \in AE$

2. For any $x, y \in AE$, both $(x+y)$ and $(x-y)$ are elements of AE

3. No string is in AE unless it can be obtained by using rules (1) and (2).

Examples of strings in AE are

i, $(i+i)$, $(i-i)$, $((i+i)-i)$, $((i-(i-i))+i)$

Finite subsets of Natural Number

F, a set of subsets of Natural Numbers, is defined as follows:

1. $\phi \in F$
2. For any $n \in N$, $\{n\} \in F$
3. For any A and B in F, $A \cup B \in F$
4. Nothing is in F unless it can be obtained

Example

Consider the following language, defined recursively :

1. $\Lambda \in L$

2. For any $y \in L$, both $0y$ and $0y1$ are in L

3. No string is in L unless it can be obtained by rules 1 and 2

Here, string in L are of the form

$$0^i 1^j, \quad i \geq j \geq 0$$

Let's prove that every string of this form is in L.

$$\text{Let } A = \{ 0^i 1^j \mid i \geq j \geq 0 \}$$

We have to prove that $A \subseteq L$

To prove $A \subseteq L$, i.e.

For every $n \geq 0$, every $x \in A$ with $|x| = n$ is an element of L.

Basis: Every $x \in A$ with $|x| = 0$ is an element of L

Proof: $|x| = 0 \Rightarrow x$ is 'Λ'

As per Rule 1 in defⁿ of L, $\Lambda \in L$.

Hypothesis: $K \geq 0$, and every x in A with $|x| \leq K$ is an element of L.

stronger PMI

Induction Stmt: Every x in A with $|x| = K+1$ is an element of L

Proof

Suppose $x \in A$ and $|x| = K+1$

$\therefore \; x = 0^i 1^j$, where $i \geq j \geq 0$

and $i + j = k + 1$

Case I $\quad i > j$

$\therefore \; x = 0y$ where $y = 0^m 1^n$,

$m \geq n \geq 0$

Also, $|y| = k$, $\therefore$ From Induction hypo.

$y \in L$

$\therefore$ Using 2nd statement (Rule) $0y \in L$

$\therefore \; x \in L$

Case II $\quad i = j$

There is atleast '1' Zero and '1' One, in x ($\because |x| = k+1$)

i.e. $|x|$ is atleast '2' and k is atleast '1'

$\therefore \; x = 0y1$ for some y.

Moreover, $y \in A$ ($\because i = j$)

$\therefore \; y = 0^m 1^m$, $m \geq 0$ (Here $m = i-1 = j-1$)

Also, $|y| \leq k$ (As such $|y|$ is $k-1$)

$\therefore$ From hypothesis, $y \in L$.

$\therefore$ Using rule 3, for any $y \in L$, $0y1 \in L$, we can generate

Recursive Definition – More Examples

* Recursive definition for the set B of positive integers divisible by 2 or 7.

1. $2 \in B$; $7 \in B$
2. For every $x \in B$ and every $n \in N$, the set of Natural Numbers, $x * n \in B$

* Recursive definition for the set A of all the strings of the form $0^i 1^j$, where $j \le i \le 2j$

1. $\Lambda \in A$
2. For every $x \in A$, both $0x1$ and $00x1$ are in A

* Recursive definition for the strings in $\{0, 1\}^*$ containing substring 00. Let B be such set.

1. $00 \in B$
2. For any $x \in B$, all the strings

* Recursive definition for set A of all strings in $\{0,1\}^*$, not containing substring '00'

1. $\Lambda \in A$; $0 \in A$

2. For $x \in A$, both $1x$ and $01x \in A$

OR

2. For $x \in A$, both $x1$ and $x10$ are in A

→ One more example of proof on strings using P. M. I.

Strings of the form $0y1$ must contain the substring 01

Let $P(N)$ be the statement :

If $|x| = N$ and $x = 0y1$ for some string $y \in \{0,1\}^*$, Then x contains the substring 01. We will prove this for $N \geq 2$.

Basis: $P(2)$ is True, i.e. $|x|=2$ and $x=0y1$ for some string $y \in \{0,1\}^*$, then x contains the substring 01.

Proof $|x|=2$ and $x=0y1 \Rightarrow x=01$, which is obviously true

Hypothesis: $k \geq 2$ and $P(k)$, i.e. if $|x|=k$ and $x=0y1$ for some string $y \in \{0,1\}^*$, then x contains the substring 01

Induction Stmt $P(k+1)$ is True, i.e. if $|x|=k+1$ and $x=0y1$ for some $y \in \{0,1\}^*$, then x contains the substring 01.

Proof $|x|=k+1$, $x=0y1$

$\therefore |y1| = k$ { Here y is Non-Null as k is atleast 2}

$\therefore$ y begins with either '0' or '1' as it is Non-Null

Case I $\quad Y$ begins with '1'

Then, $x = 0y1$ contains substring 01

as it is prefix of x

Case II $\quad Y$ begins with '0'

Here, $|y1| \geq 2 \quad (\because |y1| = k)$

Also, $y1$ begins with '0' and ends with '1'.

$\therefore$ $y1$ has the form $0z1$ for some $z \in \{0,1\}^*$

$\therefore$ $y1$ contains substring 01 using Induction Hypothesis.

$\therefore$ $x = 0y1$ also contains substring 01.

{ This can also be proved by taking string $0y$ instead of $y1$. It's length is also k. The cases would be : y ends with '0', hence x ends with suffix 01. If Y ends with 1, then it is of the form $0z1$ for $z \in \Sigma^*$ $\therefore$ it contains substring 01 from both the }

Let's continue with the language L and set A of Page (38). Let's prove that $L \subseteq A$ (converse), using a very different induction variable.

It's no. of times, Rule 2 is applied in generating x in L

(Off course it can also proved using $|x|$)

To prove,

For every $n \geq 0$, every $x \in L$, obtained by n applications of rule 2, is an element of A.

Basis: $x \in L$, x is obtained by 0 applications of Rule 2 in L, is an element of A.

Proof: x is obtained by '0' applications of Rule 2, therefore x is 'Λ'.

But, $\Lambda = 0^0 1^0$, $\therefore$ $\Lambda \in A$

Hypothesis: $k \geq 0$, and every ~~x~~ string in L that can be obtained by k applications of rule 2 is an element of A.

Induction

Stmt: Any string in L that can be obtained by k+1 applications of rule 2 is in A

Proof Let x be an element of L that is obtained by k+1 applications of rule 2.

$\therefore x = 0y$ or $x = 0y1$

where y is obtained by K applications of Rule 2 in L.

$\Rightarrow y \in A$ ($\because$ Hypo.), $\therefore y = 0^i 1^j, i \geq j \geq 0$

$\therefore x = 0y = 0^{i+1} 1^j, i+1 \geq j \geq 0$ (Actually, $i+1 > j$)

$\therefore x \in A$

or $x = 0y1 = 0^{i+1} 1^{j+1}, i+1 \geq j+1 > 0$

or $i+1 \geq j+1 \geq 0$

$\therefore x \in A$

✱ Structural Induction

- This is an induction proof based on the structure of the defⁿ.
- Here, integer N is not explicitly used.

Structural Induction Proof of $L \subseteq A$.

To prove: $L \subseteq A$

Basis : We must show that $\Lambda \in A$.

This is True because $\Lambda = 0^0 1^0$.

Hypothesis : The string $y \in L$ is an element of A.

Induction Statement : Both $0y$ and $0y1$ are elements of A.

Proof $y \in A \Rightarrow y = 0^i 1^j$, $i \geq j \geq 0$

$\therefore 0y = 0^{i+1} 1^j \in A$

and $0y1 = 0^{i+1} 1^{j+1}$ also belongs to A.

Example 2 - Structural Induction

Property of Fully Parenthesized Algebraic Expressions, already defined as :

1. $i \in AE$

2. For any x and y in AE, both $(x+y)$ and $(x-y)$ are in AE.

3. No other strings are in AE.

To prove : No string in AE contains the substring ")(".

Basis step: The string i doesn't contain the substring ")(".

Induction Hypothesis : x and y are the strings that don't contain the substring ")(".

Induction Stmt : Neither $(x+y)$ nor $(x-y)$ contain the substring ")(".

Proof :

In both the expressions, the symbol that preceds x is not ")", the symbol following x is not "(", the symbol preceding y is not ")", the symbol following y is not "(".

$\therefore$ Only way ")(" could appear in $(x+y)$ and $(x-y)$ would be ~~it for~~ for it to occur in x or y separately.

However, using hypothesis, we know that x and y don't contain the substring ")(".

[Here, we have made the hypothesis weaker than we really needed, i.e. proved slightly more than was necessary]

Here, in hypothesis, we could have written :

" If x and y are any strings in AE not containing ")(", then neither $(x+y)$ nor $(x-y)$ contain ")(". "

In our Induction step, we showed this not only for x and y in AE, but for any x and y.

This simplification is often, though not always, possible.

Recursive definition for the language of strings with more a's than b's

Let $L \subseteq \Sigma^*$, where $\Sigma = \{a, b\}$

1. $a \in L$.

2. For any $x \in L$, $ax \in L$.

3. For any x and y in L, all the strings bxy, xby, xyb are in L.

4. No other strings are in L.

Let's prove that every element of L has more a's than b's, using Structural Induction

(We will prove something stronger than required)

Ex. 3 To prove: Every element of L has more a's than b's

Basis: The string 'a' has more a's than b's.

Hypothesis

x and y are the strings containing more a's than b's

Induction Statement

Each of the strings ax, bxy, xby, xyb has more a's than b's

Proof ax has more a's than b's because x has.

Since both x and y have more a's than b's, xy has atleast two more a's than b's.

and therefore any string formed by inserting one more b still has atleast one more 'a' than 'b'.

Recursive definition of Length and Reverse Function

Length	Reverse x^r means Rev(x)
1. $\|\Lambda\| = 0$	1. $\Lambda^r = \Lambda$
2. For any $x \in \Sigma^*$, $a \in \Sigma$ $\|xa\| = \|x\| + 1$	2. For any $x \in \Sigma^*, a \in \Sigma$, $(xa)^r = a x^r$

Ex. 4 Structural Induction Proof on property of Length Function.

For every x and y in Σ^*,

$$|xy| = |x| + |y|$$

The proof is based on $|y|$

Statement

For every y, $|xy| = |x| + |y|$

Basis : y is 'Λ'

$$|x\Lambda| = |x| + |\Lambda|$$

is True because $|\Lambda| = 0$

Induction Hypothesis Y is a string for which the statement holds.

i.e. $|xY| = |x| + |Y|$

Induction Statement $|x(ya)| = |x| + |(ya)|$

Proof $|x(ya)| = |(xy)a|$ ($\because$ Concatenation is associative)

$= |(xy)| + 1$ ($\because$ defn of Length fn)

$= (|x| + |y|) + 1$ ($\because$ Hypo.)

$= |x| + (|y| + 1)$ ($\because$ addition is associative)

$= |x| + (|(ya)|)$ ($\because$ defn of length fn)

$=$ RHS