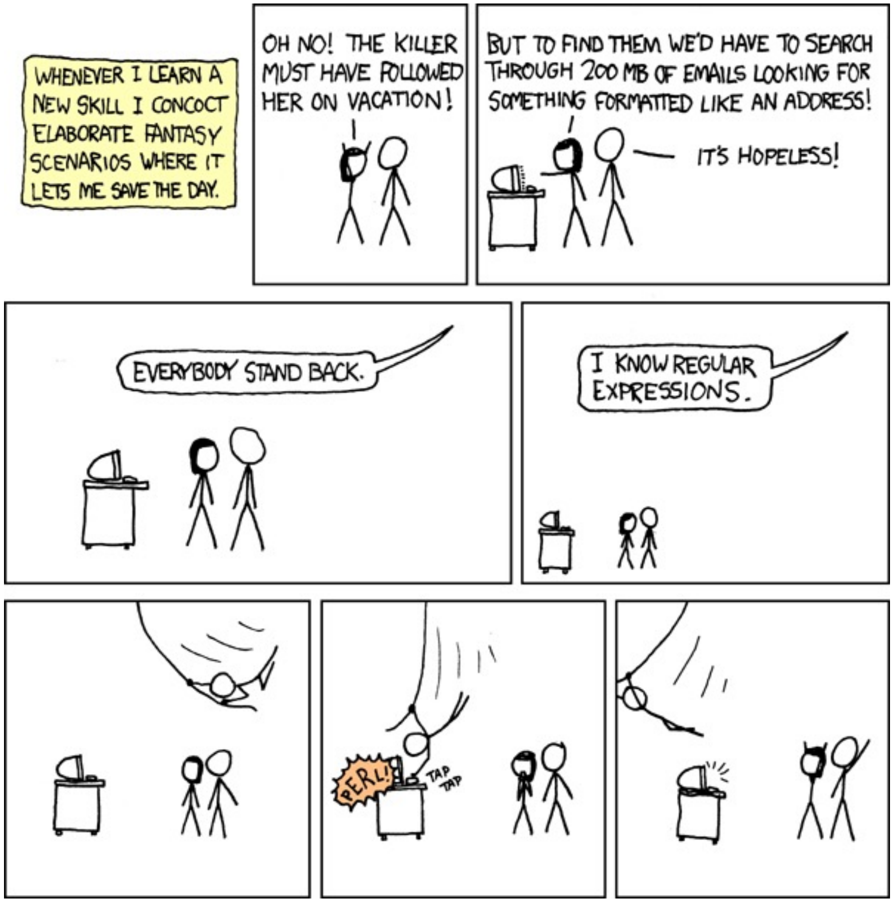
CSE 311: Foundations of Computing

Lecture 20: Structural Induction, Regular Expressions



Last Time: Recursive Definitions

- Any recursively defined set can be translated into a Java class
- Any recursively defined function can be translated into a Java function
  - some (but not all) can be written more cleanly as loops
- Recursively defined functions and sets are our mathematical models of code and the data it operates on

Last time: Structural Induction

How to prove $\forall\, x \in S, P(x)$ is true:

Base Case: Show that $P(u)$ is true for all specific elements u of S mentioned in the *Basis step*

Inductive Hypothesis: Assume that P is true for some arbitrary values of *each* of the existing named elements mentioned in the *Recursive step*

Inductive Step: Prove that $P(w)$ holds for each of the new elements w constructed in the *Recursive step* using the named elements mentioned in the Inductive Hypothesis

Conclude that $\forall\, x \in S, P(x)$

Linked Lists of Integers

- **Basis:** null ∈ **Lists**
- **Recursive step:**

  If L ∈ **Lists and** v ∈ ℤ, **then** Node(v, L) ∈ **Lists**

Examples:

– null	[]
– Node(1, null)	[1]
– Node(1, Node(2, null))	[1, 2]

Functions on Linked Lists

Set of numbers stored in a list:

- **values**(null) = ∅
- **values**(Node(v, L)) = {v} ∪ **values**(L)

Example:

values(Node(1, Node(2, null))

= {1} ∪ **values**(Node(2, null)	**Def of values**
= {1} ∪ {2} ∪ **values**(null)	**Def of values**
= {1} ∪ {2} ∪ ∅	**Def of values**
= {1, 2}	**Def of** ∪

Functions on Linked Lists

Remove the numbers that don't satisfy p(v):

- **filter**$_p$(null) = null
- **filter**$_p$(Node(v, L)) = Node(v, **filter**$_p$(L)) **if** p(v)
- **filter**$_p$(Node(v, L)) = **filter**$_p$(L) **otherwise**

Example: p(v) := v < 2

filter$_p$(Node(1, Node(2, null)))

= Node(1, **filter**$_p$(Node(2, null))) **Def filter**$_p$

= Node(1, **filter**$_p$(null)) **Def filter**$_p$

= Node(1, null) **Def filter**$_p$

Claim: $x \in \textbf{values}(\textbf{filter}_p(L))$ iff $p(x) \wedge x \in \textbf{values}(L)$

Claim: $x \in$ **values**(**filter**$_p$(L)) iff $p(x) \wedge x \in$ **values**(L)

Q(L) := "$x \in$ **values**(**filter**$_p$(L)) iff $p(x) \wedge x \in$ **values**(L) for all $x \in \mathbb{Z}$".
We will prove Q(L) for $L \in$ **Lists** by structural induction.

Claim: $x \in \textbf{values}(\textbf{filter}_p(L))$ iff $p(x) \land x \in \textbf{values}(L)$

$Q(L)$:= "$x \in \textbf{values}(\textbf{filter}_p(L))$ iff $p(x) \land x \in \textbf{values}(L)$ for all $x \in \mathbb{Z}$".
We will prove $Q(L)$ for $L \in$ **Lists** by structural induction.

Base Case: Let $x \in \mathbb{Z}$ **be arbitrary.**

LHS is $x \in \textbf{values}(\textbf{filter}_p(\text{null}))$

$\equiv x \in \textbf{values}(\text{null})$	Def of $\textbf{filter}_p$
$\equiv x \in \emptyset$	Def of **values**
$\equiv F$	Def of $\emptyset$

RHS is $p(x) \land x \in \textbf{values}(\text{null})$

$\equiv p(x) \land x \in \emptyset$	Def of **values**
$\equiv p(x) \land F$	Def of $\emptyset$
$\equiv F$	Domination

These are equivalent as required (LHS $\equiv F \equiv$ **RHS).**
Since x was arbitrary, this shows that $Q(\text{null})$ **holds.**

Claim: $x \in$ **values**(**filter**$_p$(L)) iff $p(x) \land x \in$ **values**(L)

Q(L) := "$x \in$ **values**(**filter**$_p$(L)) iff $p(x) \land x \in$ **values**(L) for all $x \in \mathbb{Z}$".
We will prove Q(L) for L ∈ **Lists** by structural induction.

Base Case: … so Q(null) **holds.**

Inductive Hypothesis: Suppose Q(L) **holds for an arbitrary list L, i.e., we have** $x \in$ **values**(**filter**$_p$(L)) iff $p(x) \land x \in$ **values**(L)**.**

Inductive Step: Goal: Prove Q(Node(v, L)) **for all** $v \in \mathbb{Z}$

Claim: $v \in \textbf{values}(\textbf{filter}_p(L))$ iff $p(x) \wedge x \in \textbf{values}(L)$

$Q(L) :=$ "$x \in \textbf{values}(\textbf{filter}_p(L))$ iff $p(x) \wedge x \in \textbf{values}(L)$ for all $x \in \mathbb{Z}$".
We will prove Q(L) for $L \in$ **Lists** by structural induction.

Base Case: ... so Q(null) **holds.**

Inductive Hypothesis: Suppose Q(L) **holds for an arbitrary list L,**
i.e., we have $x \in \textbf{values}(\textbf{filter}_p(L))$ iff $p(x) \wedge x \in \textbf{values}(L)$.

Inductive Step: **Goal: Prove** Q(Node(v, L)) **for all** $v \in \mathbb{Z}$

Let $v, x \in \mathbb{Z}$ **be arbitrary. We go by cases. Suppose** $\neg p(v)$.

$x \in \textbf{values}(\textbf{filter}_p(\text{Node}(v, L)))$

$\equiv x \in \textbf{values}(\textbf{filter}_p(L))$	Def $\textbf{filter}_p$
$\equiv p(x) \wedge x \in \textbf{values}(L)$	IH

...

$\equiv p(x) \wedge x \in \textbf{values}(\text{Node}(v, L))$

Claim: $v \in$ **values**(**filter**$_p$(L)) iff $p(x) \land x \in$ **values**(L)

Q(L) := "$x \in$ **values**(**filter**$_p$(L)) iff $p(x) \land x \in$ **values**(L) for all $x \in \mathbb{Z}$".
We will prove Q(L) for L ∈ **Lists** by structural induction.

Base Case: **... so** Q(null) **holds.**

Inductive Hypothesis: Suppose Q(L) **holds for an arbitrary list L,**
i.e., we have $x \in$ **values**(**filter**$_p$(L)) iff $p(x) \land x \in$ **values**(L)**.**

Inductive Step: **Goal: Prove** Q(Node(v, L)) **for all** $v \in \mathbb{Z}$

Let $v, x \in \mathbb{Z}$ **be arbitrary. We go by cases. Suppose** $\neg p(v)$**.**

$x \in$ **values**(**filter**$_p$(Node(v, L)))	
$\equiv x \in$ **values**(**filter**$_p$(L))	Def **filter**$_p$
$\equiv p(x) \land x \in$ **values**(L)	IH

If $\neg p(x)$**, then this and** $p(x) \land x \in$ **values**(Node(v, L)) **are equivalent as they are both false. So now suppose** $p(x)$**...**

Claim: $v \in$ **values**(**filter**$_p$(L)) iff $p(x) \wedge x \in$ **values**(L)

Q(L) := "$x \in$ **values**(**filter**$_p$(L)) iff $p(x) \wedge x \in$ **values**(L) for all $x \in \mathbb{Z}$".
We will prove Q(L) for L ∈ **Lists** by structural induction.

Base Case: ... so Q(null) **holds.**

Inductive Hypothesis: Suppose Q(L) **holds for an arbitrary list L,**
i.e., we have $x \in$ **values**(**filter**$_p$(L)) iff $p(x) \wedge x \in$ **values**(L).

Inductive Step: **Goal: Prove** Q(Node(v, L)) **for all** $v \in \mathbb{Z}$

Let $v, x \in \mathbb{Z}$ **be arbitrary. We go by cases. Suppose** $\neg p(v)$.

$x \in$ **values**(**filter**$_p$(Node(v, L)))	
$\equiv x \in$ **values**(**filter**$_p$(L))	Def **filter**$_p$
$\equiv p(x) \wedge x \in$ **values**(L)	IH
...	
$\equiv p(x) \wedge (x \in \{v\} \vee x \in$ **values**(L))	
$\equiv p(x) \wedge (x \in \{v\} \cup$ **values**(L))	Def ∪
$\equiv p(x) \wedge (x \in$ **values**(Node(v, L)))	Def **values**

suppose p(x)...

Claim: $v \in$ **values**(**filter**$_p$(L)) iff $p(x) \land x \in$ **values**(L)

Q(L) := "$x \in$ **values**(**filter**$_p$(L)) iff $p(x) \land x \in$ **values**(L) for all $x \in \mathbb{Z}$".
We will prove Q(L) for L ∈ **Lists** by structural induction.

Base Case: ... so Q(null) **holds.**

Inductive Hypothesis: Suppose Q(L) **holds for an arbitrary list L, i.e., we have** $x \in$ **values**(**filter**$_p$(L)) iff $p(x) \land x \in$ **values**(L)**.**

Inductive Step: **Goal: Prove** Q(Node(v, L)) **for all** $v \in \mathbb{Z}$

Let $v, x \in \mathbb{Z}$ **be arbitrary. We go by cases. Suppose** $\neg p(v)$**.**

$x \in$ **values**(**filter**$_p$(Node(v, L)))	
$\equiv x \in$ **values**(**filter**$_p$(L))	Def **filter**$_p$
$\equiv p(x) \land x \in$ **values**(L)	IH
$\equiv p(x) \land (F \lor x \in$ **values**(L))	Identity
$\equiv p(x) \land (x \in \{v\} \lor x \in$ **values**(L))	??
$\equiv p(x) \land (x \in \{v\} \cup$ **values**(L))	Def ∪
$\equiv p(x) \land (x \in$ **values**(Node(v, L)))	Def **values**

suppose p(x)...

Claim: $v \in$ **values(filter$_p$**(L)) iff $p(x) \land x \in$ **values**(L)

Q(L) := "$x \in$ **values(filter$_p$**(L)) iff $p(x) \land x \in$ **values**(L) for all $x \in \mathbb{Z}$".
We will prove Q(L) for $L \in$ **Lists** by structural induction.

Base Case: ... so Q(null) **holds.**

Inductive Hypothesis: Suppose Q(L) **holds for an arbitrary list L,**
i.e., we have $x \in$ **values(filter$_p$**(L)) iff $p(x) \land x \in$ **values**(L)**.**

Inductive Step: Goal: Prove Q(Node(v, L)) **for all** $v \in \mathbb{Z}$

Let $v, x \in \mathbb{Z}$ **be arbitrary. We go by cases. Suppose** $\neg p(v)$**.**

$x \in$ **values(filter$_p$**(Node(v, L)))	
$\equiv x \in$ **values(filter$_p$**(L))	Def **filter$_p$**
$\equiv p(x) \land x \in$ **values**(L)	IH
$\equiv p(x) \land (F \lor x \in$ **values**(L))	Identity
$\equiv p(x) \land (x \in \{v\} \lor x \in$ **values**(L))	$x \neq v$ as $p(x)$ but $\neg p(v)$
$\equiv p(x) \land (x \in \{v\} \cup$ **values**(L))	Def $\cup$
$\equiv p(x) \land (x \in$ **values**(Node(v, L)))	Def **values**

suppose p(x)...

Claim: $v \in$ **values**(**filter**$_p$(L)) iff $p(x) \wedge x \in$ **values**(L)

Q(L) := "x ∈ **values**(**filter**$_p$(L)) iff $p(x) \wedge x \in$ **values**(L) for all $x \in \mathbb{Z}$".
We will prove Q(L) for L ∈ **Lists** by structural induction.

Base Case: ... **so** Q(null) **holds.**

Inductive Hypothesis: **Suppose** Q(L) **holds for an arbitrary list L,**
i.e., we have x ∈ **values**(**filter**$_p$(L)) iff $p(x) \wedge x \in$ **values**(L).

Inductive Step: Goal: Prove Q(Node(v, L)) **for all** $v \in \mathbb{Z}$

Let $v, x \in \mathbb{Z}$ **be arbitrary. We go by cases. Suppose** $\neg p(v)$**.**

x ∈ **values**(**filter**$_p$(Node(v, L)))
≡ ...
≡ $p(x) \wedge$ (x ∈ **values**(Node(v, L)))

Thus, by cases ($p(x)$ **&** $\neg p(x)$)**, the claimed bicondition holds.**
Since x **was arbitrary, we have shown** Q(Node(v, L))**.**

Claim: $v \in \textbf{values}(\textbf{filter}_p(L))$ iff $p(x) \wedge x \in \textbf{values}(L)$

$Q(L)$:= "$x \in \textbf{values}(\textbf{filter}_p(L))$ iff $p(x) \wedge x \in \textbf{values}(L)$ for all $x \in \mathbb{Z}$".
We will prove $Q(L)$ for $L \in$ **Lists** by structural induction.

Base Case: ... so $Q(\text{null})$ **holds.**

Inductive Hypothesis: Suppose $Q(L)$ **holds for an arbitrary list L,**
i.e., we have $x \in \textbf{values}(\textbf{filter}_p(L))$ iff $p(x) \wedge x \in \textbf{values}(L)$.

Inductive Step: **Goal: Prove** $Q(\text{Node}(v, L))$ **for all** $v \in \mathbb{Z}$

Let $v, x \in \mathbb{Z}$ **be arbitrary. We go by cases. Suppose** $p(v)$.

$x \in \textbf{values}(\textbf{filter}_p(\text{Node}(v, L)))$	
$\equiv x \in \textbf{values}(\text{Node}(v, \textbf{filter}_p(L)))$	Def $\textbf{filter}_p$
$\equiv x \in \{v\} \cup \textbf{values}(\textbf{filter}_p(L))$	Def **values**
$\equiv x \in \{v\} \vee x \in \textbf{values}(\textbf{filter}_p(L))$	Def $\cup$
$\equiv x \in \{v\} \vee (p(x) \wedge x \in \textbf{values}(L))$	IH

Claim: $v \in$ **values**(**filter**$_p$(L)) iff $p(x) \wedge x \in$ **values**(L)

Q(L) := "$x \in$ **values**(**filter**$_p$(L)) iff $p(x) \wedge x \in$ **values**(L) for all $x \in \mathbb{Z}$".
We will prove Q(L) for L ∈ **Lists** by structural induction.

Base Case: ... so Q(null) **holds.**

Inductive Hypothesis: Suppose Q(L) **holds for an arbitrary list L,**
i.e., we have $x \in$ **values**(**filter**$_p$(L)) iff $p(x) \wedge x \in$ **values**(L).

Inductive Step: **Goal: Prove** Q(Node(v, L)) **for all** $v \in \mathbb{Z}$

Let $v, x \in \mathbb{Z}$ **be arbitrary. We go by cases. Suppose** p(v).

$x \in$ **values**(**filter**$_p$(Node(v, L)))	
$\equiv x \in$ **values**(Node(v, **filter**$_p$(L)))	Def **filter**$_p$
$\equiv x \in \{v\} \cup$ **values**(**filter**$_p$(L))	Def **values**
$\equiv x \in \{v\} \vee x \in$ **values**(**filter**$_p$(L))	Def ∪
$\equiv x \in \{v\} \vee (p(x) \wedge x \in$ **values**(L))	IH
$\equiv (x \in \{v\} \vee p(x)) \wedge (x \in \{v\} \vee x \in$ **values**(L))	Distributivity
$\equiv (x \in \{v\} \vee p(x)) \wedge (x \in$ **values**(Node(v, L)))	Def ∪, **values**

Claim: $v \in$ **values**(**filter**$_p$(L)) iff $p(x) \land x \in$ **values**(L)

Q(L) := "$x \in$ **values**(**filter**$_p$(L)) iff $p(x) \land x \in$ **values**(L) for all $x \in \mathbb{Z}$".
We will prove Q(L) for $L \in$ **Lists** by structural induction.

Base Case: … so Q(null) **holds.**

Inductive Hypothesis: Suppose Q(L) **holds for an arbitrary list L,**
i.e., we have $x \in$ **values**(**filter**$_p$(L)) iff $p(x) \land x \in$ **values**(L)**.**

Inductive Step: Goal: Prove Q(Node(v, L)) **for all** $v \in \mathbb{Z}$

Let $v, x \in \mathbb{Z}$ **be arbitrary. We go by cases. Suppose** p(v)**.**

$x \in$ **values**(**filter**$_p$(Node(v, L)))

$\equiv \ldots$

$\equiv (x \in \{v\} \lor p(x)) \land (x \in$ **values**(Node(v, L)))

If $x \in \{v\}$ **is false, then the first part is** $F \lor p(x) \equiv p(x)$.

If true, then $x = v$, **and first part and** p(x) **are both true. Thus,**

$\equiv p(x) \land (x \in$ **values**(Node(v, L)))

Claim: $v \in$ **values**(**filter**$_p$(L)) iff $p(x) \land x \in$ **values**(L)

Q(L) := "$x \in$ **values**(**filter**$_p$(L)) iff $p(x) \land x \in$ **values**(L) for all $x \in \mathbb{Z}$".
We will prove Q(L) for L $\in$ **Lists** by structural induction.

Base Case: … so Q(null) **holds.**

Inductive Hypothesis: Suppose Q(L) **holds for an arbitrary list L,**
i.e., we have $x \in$ **values**(**filter**$_p$(L)) iff $p(x) \land x \in$ **values**(L).

Inductive Step: Goal: Prove Q(Node(v, L)) **for all** $v \in \mathbb{Z}$

Let $v, x \in \mathbb{Z}$ **be arbitrary. We go by cases. Suppose** p(v).

$x \in$ **values**(**filter**$_p$(Node(v, L)))
$\equiv$ …
$\equiv p(x) \land (x \in$ **values**(Node(v, L)))

Thus, by cases, the claimed bicondition holds.
Since x **was arbitrary, we have shown** Q(Node(v, L)).

Hence, we have shown Q(L) **for all lists by structural induction.**

Theoretical Computer Science

Languages: Sets of Strings

- Subsets of strings are called *languages*
- Examples:
  - $\Sigma^* =$ All strings over alphabet Σ
  - Palindromes over Σ
  - Binary strings that don't have a 0 after a 1
  - Binary strings with an equal # of 0's and 1's
  - Legal variable names in Java/C/C++
  - Syntactically correct Java/C/C++ programs
  - Valid English sentences

Foreword on Intro to Theory C.S.

- Look at different ways of defining languages
- See which are more expressive than others
  - i.e., which can define more languages

- Later: connect ways of defining languages to different types of (restricted) computers
  - computers capable of recognizing those languages i.e., distinguishing strings in the language from not

- Consequence: computers that recognize more expressive languages are more powerful

Regular Expressions

Regular expressions over Σ

- **Basis:**

  ε is a regular expression (could also include $\varnothing$)

  a is a regular expression for any $a \in \Sigma$

- **Recursive step:**

  If **A** and **B** are regular expressions then so are:

  A $\cup$ **B**

  AB

  A*

Each Regular Expression is a "pattern"

ε matches only the **empty string**

a matches only the one-character string *a*

A ∪ **B** matches all strings that either **A** matches or **B** matches (or both)

AB matches all strings that have a first part that **A** matches followed by a second part that **B** matches

A* matches all strings that have any number of strings (even 0) that **A** matches, one after another (**ε** ∪ **A** ∪ **AA** ∪ **AAA** ∪ **...**)

Definition of the *language* matched by a regular expression

Language of a Regular Expression

The language defined by a regular expression:

$$\mathrm{L}(\varepsilon) = \{\varepsilon\}$$

$$\mathrm{L}(a) = \{a\}$$

$$\mathrm{L}(A \cup B) = L(A) \cup L(B)$$

$$\mathrm{L}(AB) = \{x \bullet y \mid x \in L(A), y \in L(B)\}$$

$$\mathrm{L}(A^*) = \bigcup_{n=0}^{\infty} A^n$$

A^n defined recursively by

$$A^0 = \emptyset$$

$$A^{n+1} = A^n A$$

Examples

001*

0*1*

Examples

001*

{00, 001, 0011, 00111, ...}

0*1*

Any number of 0's followed by any number of 1's

Examples

$(0 \cup 1)\ 0\ (0 \cup 1)\ 0$

$(0^*1^*)^*$

Examples

$(\mathit{0} \cup \mathit{1})\ \mathit{0}\ (\mathit{0} \cup \mathit{1})\ \mathit{0}$

{0000, 0010, 1000, 1010}

$(\mathit{0}^*\mathit{1}^*)^*$

All binary strings

Examples

$(0 \cup 1)^* \, 0110 \, (0 \cup 1)^*$

$(00 \cup 11)^* \, (01010 \cup 10001) \, (0 \cup 1)^*$

Examples

$(0 \cup 1)^* \, 0110 \, (0 \cup 1)^*$

Binary strings that contain “0110”

$(00 \cup 11)^* \, (01010 \cup 10001) \, (0 \cup 1)^*$

Binary strings that begin with pairs of characters followed by “01010” or “10001”

Examples

- All binary strings that have an even # of 1's

Examples

- All binary strings that have an even # of 1's

  e.g., 0*(10*10*)*

Examples

- All binary strings that have an even # of 1's

  e.g., 0*(10*10*)*

- All binary strings that *don't* contain 101

Examples

- All binary strings that have an even # of 1's

  e.g., 0*(10*10*)*

- All binary strings that *don't* contain 101

  e.g., 0*(1 ∪ 1000*)*(0* ∪ 10*)

  at least two 0s between 1s

Regular Expressions in Practice

- Used to define the "tokens": e.g., legal variable names, keywords in programming languages and compilers
- Used in **`grep`**, a program that does pattern matching searches in UNIX/LINUX
- Pattern matching using regular expressions is an essential feature of PHP
- We can use regular expressions in programs to process strings!

Regular Expressions in Java

- Pattern p = Pattern.compile("a*b");
- Matcher m = p.matcher("aaaaab");
- boolean b = m.matches();

`[01]` a 0 or a 1 `^` start of string `$` end of string

`[0-9]` any single digit `\.` period `\,` comma `\-` minus

`.` any single character

ab	a followed by b	(**AB**)
(a|b)	a or b	(**A** ∪ **B**)
a?	zero or one of a	(**A** ∪ ε)
a*	zero or more of a	**A***
a+	one or more of a	**AA***

- e.g. `^[\-+]?[0-9]*(\.|\,)?[0-9]+$`

  General form of decimal number e.g. 9.12 or -9,8 (Europe)

Limitations of Regular Expressions

- Not all languages can be specified by regular expressions
- Even some easy things like
  - Palindromes
  - Strings with equal number of 0’s and 1’s
- But also more complicated structures in programming languages
  - Matched parentheses
  - Properly formed arithmetic expressions
  - etc.

Context-Free Grammars

- A Context-Free Grammar (CFG) is given by a finite set of substitution rules involving
  - A finite set **V** of *variables* that can be replaced
  - Alphabet Σ of *terminal symbols* that can't be replaced
  - One variable, usually **S**, is called the *start symbol*

- The substitution rules involving a variable **A**, written as

$$\mathbf{A} \rightarrow w_1 \mid w_2 \mid \cdots \mid w_k$$

where each w_i is a string of variables and terminals

– that is $w_i \in (\mathbf{V} \cup \Sigma)^*$

How CFGs generate strings

- Begin with start symbol **S**

- If there is some variable **A** in the current string you can replace it by one of the w's in the rules for **A**
  - $\mathbf{A} \to w_1 \mid w_2 \mid \cdots \mid w_k$
  - Write this as $x\mathbf{A}y \Rightarrow xwy$
  - Repeat until no variables left

- The set of strings the CFG describes are all strings, containing no variables, that can be *generated* in this manner (after a finite number of steps)

Example Context-Free Grammars

Example: $\mathbf{S} \rightarrow 0\mathbf{S}0 \mid 1\mathbf{S}1 \mid 0 \mid 1 \mid \varepsilon$

Example Context-Free Grammars

Example: $\mathbf{S} \rightarrow 0\mathbf{S}0 \mid 1\mathbf{S}1 \mid 0 \mid 1 \mid \varepsilon$

The set of all binary palindromes

Example Context-Free Grammars

Example: $\mathbf{S} \rightarrow 0\mathbf{S}0 \mid 1\mathbf{S}1 \mid 0 \mid 1 \mid \varepsilon$

The set of all binary palindromes

Example: $\mathbf{S} \rightarrow 0\mathbf{S} \mid \mathbf{S}1 \mid \varepsilon$

Example Context-Free Grammars

Example: $\mathbf{S} \rightarrow 0\mathbf{S}0 \mid 1\mathbf{S}1 \mid 0 \mid 1 \mid \varepsilon$

The set of all binary palindromes

Example: $\mathbf{S} \rightarrow 0\mathbf{S} \mid \mathbf{S}1 \mid \varepsilon$

0*1*