SCC.312 Languages and Compilation (Week 11)

Paul Rayson

p.rayson@lancaster.ac.uk

B50 InfoLab21

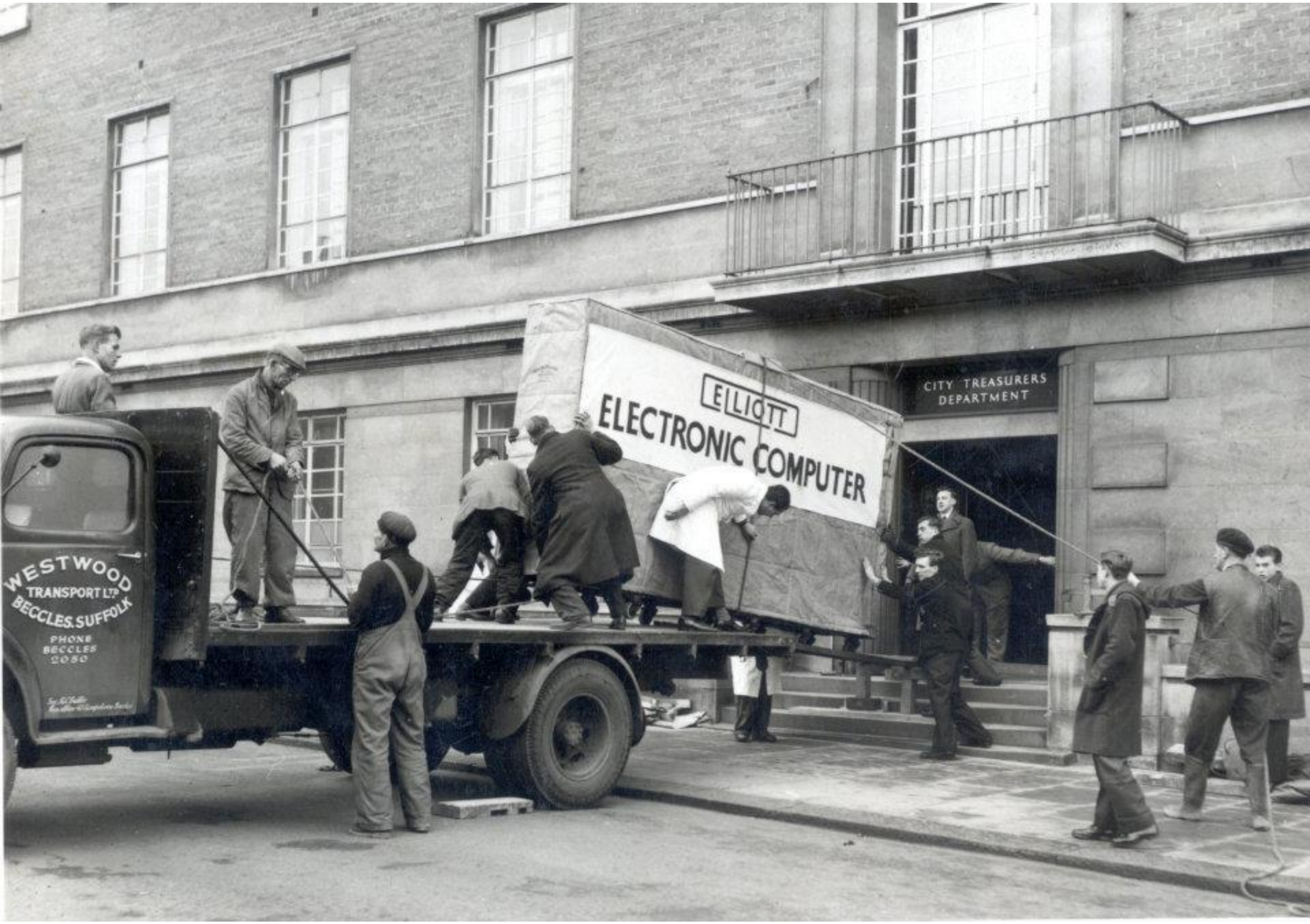
Course Content (weeks 11-15)

- Languages
  - Basic properties and definitions of languages
- Grammars
  - Different types of phase structure grammars and their relation to one another
- Abstract Machines (Automata)
  - Different machines to recognise grammars
  - Their limitations and uses

Famous faces in Computer Science?

Formal Computer Science

- Why?
  - for representation, design and understanding.
  - abstract foundation of all computing activity that is independent of device and programming language.
  - Compilers and interpreters


ELLIOTT
ELECTRONIC COMPUTER
CITY TREASURERS DEPARTMENT
WESTWOOD TRANSPORT LTD
BECCLES. SUFFOLK
PHONE BECCLES 2050

This week's topics

- Introduction to the elements and methods of defining formal languages
- Definitions of basic elements of languages
  - Strings, sentences, alphabets, operations
- Different ways of defining languages
  - Set definitions, decision programs, grammars
- Introduction to phrase structure grammars

Week 1 Learning Objectives

By the end of week 1 you should be able to:

Obj1: identify the roles of *formal grammars* (that is, why we need them)

Obj2: generate a *derivation tree* to show how a valid string can be derived from a specified grammar (top-down)

Obj3: generate a *parse tree* to show how a sentence may be broken down into its constituent parts (bottom-up)

- N.B. the above points apply equally to fragments of *natural languages* (for example, English) and *programming languages*

Meta level notes on Learning Outcomes etc will appear in sticky notes

Example Languages

- The English Language
  - The set of all grammatically correct English sentences
- Java Programming Language
  - The set of all syntactically valid Java source file programs
- How can we define languages like these?

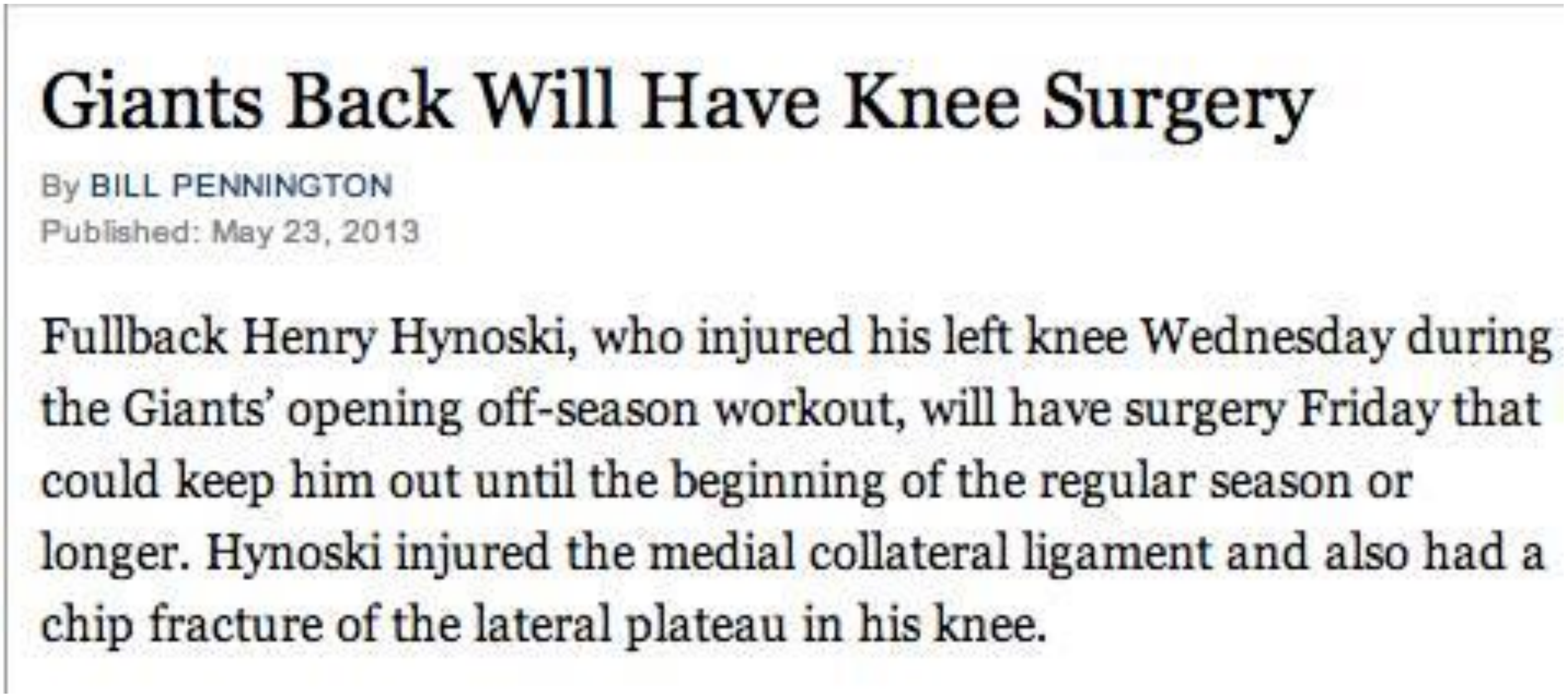
Nine ‘hads’ in a row

- Simon Mayo Drivetime, BBC Radio 2, Monday 3rd February 2014
- http://www.bbc.co.uk/programmes/b03t35j0
- "Simon, where Rebecca had had 'had’, had had 'had had’; 'had had' was the correct answer.”

- http://en.wikipedia.org/wiki/James_while_John_had_had_had_had_had_had_had_had_had_had_had_a_better_effect_on_the_teacher

Today in "linguists are not kidding when they say that language enables you to understand sentences that have never been said before in the entirety of human history"

... See Gretchen McCulloch's thread on Twitter/X:



Giants Back Will Have Knee Surgery

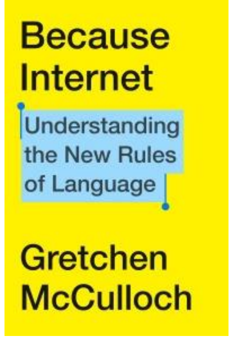
By BILL PENNINGTON
Published: May 23, 2013

Fullback Henry Hynoski, who injured his left knee Wednesday during the Giants' opening off-season workout, will have surgery Friday that could keep him out until the beginning of the regular season or longer. Hynoski injured the medial collateral ligament and also had a chip fracture of the lateral plateau in his knee.

Pigeons wearing tiny cowboy hats spotted in Las Vegas

Charity concerned for welfare of birds, nicknamed Cluck Norris and Coo-lamity Jane

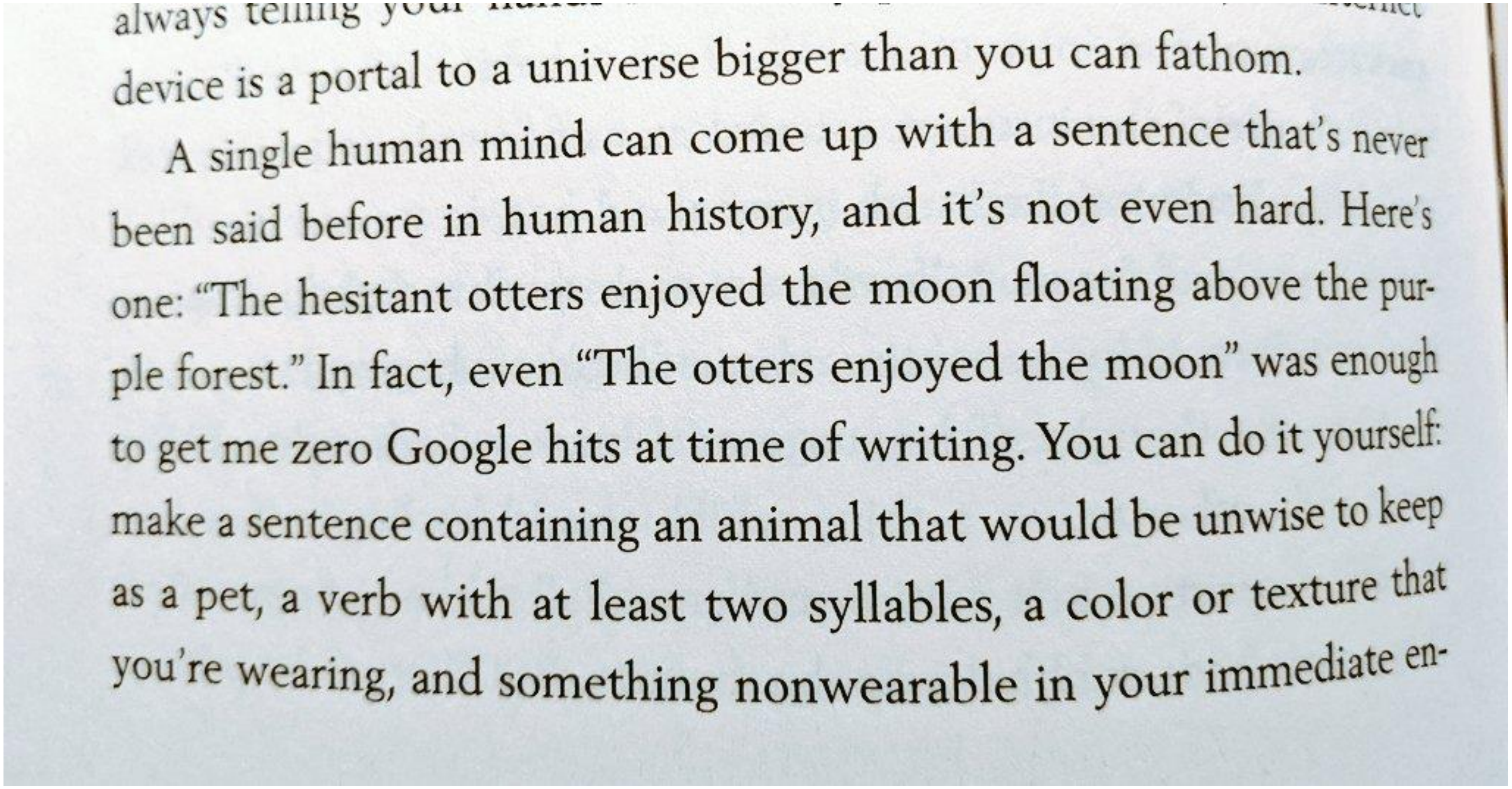
Read Gretchen McCulloch's (2019) book …
https://gretchenmcculloch.com/book/



Because Internet

Understanding the New Rules of Language

Gretchen McCulloch



always telling your

device is a portal to a universe bigger than you can fathom.

A single human mind can come up with a sentence that's never been said before in human history, and it's not even hard. Here's one: "The hesitant otters enjoyed the moon floating above the purple forest." In fact, even "The otters enjoyed the moon" was enough to get me zero Google hits at time of writing. You can do it yourself: make a sentence containing an animal that would be unwise to keep as a pet, a verb with at least two syllables, a color or texture that you're wearing, and something nonwearable in your immediate en-

Defining a Language

- In this lecture we will be looking at 3 ways to define a language
  - Set Definitions
  - Decision Programs
  - Grammars

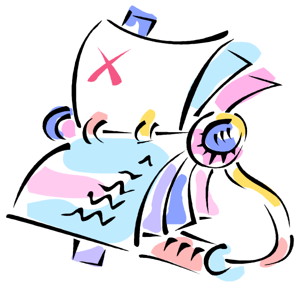
- Before we start we need to define the various parts that make up a language.

Definition - Alphabet

- An **alphabet** is a finite set of symbols
  - The symbols are atomic
  - No duplicate symbols
- Examples
  - {*0, 1, 2, 3, 4, 5, 6, 7, 8, 9*} – set of numbers
  - {*a, e, i, o, u*} – set of vowels
  - {*a, b, e, 1, x, y, q, 6, 3*} – set of symbols

Remember '*Set*' Abstract Data Type in SCC120?

Definition - String



- A **string** is a sequence of atomic symbols taken from a finite alphabet
- Example alphabet – {*a, b, c, e, t*}
- Example strings from this alphabet are
  - *cat, bat, tab*
  - *aaa, abc, aba, ttb*

Definition – Empty String

- There is a special string called the **empty string**
  - This contains no symbols at all
  - It is usually denoted by the symbol ε (epsilon)
  - Note: JFlap uses λ (lambda) as the empty string (by default)

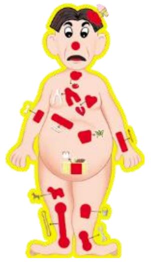
Definition - Language

- A **language** is the complete set of strings that can be formed from the symbols of an alphabet
  - The set may possibly be finite or infinite
- Example alphabet – {*a, b*}
  - Strings – *a, b, aa, ab, ba, bb, aaa, aab,* etc.
  - Language – {*a, b, aa, ab, ba, bb, aaa, aab,* etc}

Definition - Sentence

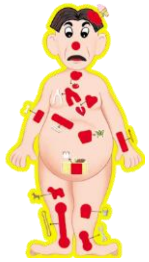
- When a string is a member of the language it is often called a **sentence** of the language
- Example Language – {*aa, ab, ba, bb*}
  - Example sentence – *ba*
- Not all sentences are sequences of unbroken symbols as we shall see later
  - if i > 1 then x := x + 1

Operation - Length



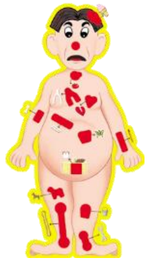
- The number of symbols in the string x
- Written as: |x|
- Examples:
  - |*abcabca*| = 7
  - |abacab| = 6
  - |*a*| = 1
  - |ε| = 0

Operation - Concatenation



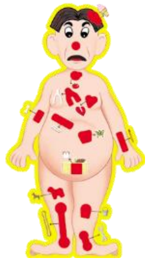
- The string formed by the string x followed by the string y
- Written as: xy
- Examples: let x = *abca* and y = *ca*
  - xy = *abcaca*   yx = *caabca*
  - xε = *abca*   εx = *abca*

Operation - Power



- The string formed by writing down n copies of the string x
- Written as: x^n (where n is a whole number)
- Examples: let x = *abca*
  - x^3 = *abcaabcaabca*
  - x^1 = x = *abca*
  - $x^0 = \varepsilon$

Operation - Index



- The nth symbol in the string x
  - Treats the string as an array of symbols
- Written as: x_n (where n is a whole number)
- Examples: let x = *abca*
  - $x_1 = a$
  - $x_2 = b$
  - $x_3 = c$

Defining a Language

- There are 3 ways to define a language
  - Set Definitions
  - Decision Programs
  - Grammars

Set Definitions of Languages

- If a language is such that the strings follow certain patterns we are able to specify it by providing a set definition
  - Makes use of the 4 operations
- Example: $\{a^i b^i : i \geq 1\}$

Form of Set Definitions

- A set definition has 3 parts

$$\{a^i b^i : i \geq 1\}$$

$a^i b^i$	is the function to produce strings
:	is “such that” (sometimes written as |)
$i \geq 1$	the range of arguments to the function

Examples of Set Definitions

- $\{a^i b^i : i \geq 1\}$
  - All non empty strings of ‘a’s followed by ‘b’s where there is an equal number of each
- $\{a^i b^i : i \geq 0\}$
  - Same as above, but also allows ε (when i = 0, $a^0 b^0 = \varepsilon\varepsilon = \varepsilon$)
- $\{a^i b^i c^{i+j} : i \geq 1, j \geq 0\}$
- $\{a^i b^{2j-1} : i \geq 1, j \geq 1\}$

Finite and Infinite Sets

- All of the previous examples specify infinite sets.
- Set definitions can also specify finite sets.
- Example: $\{a^i b^j : 1 \leq i \leq 3, j = i + 1\}$
  - i = 1, j = 2 = *abb*
  - i = 2, j = 3 = *aabbb*
  - i = 3, j = 4 = *aaabbbb*

Other Set Definition Examples

- $\{a^i b^{i \bmod 3} : i \geq 1\}$
  - All strings consisting of any number of ‘a’s followed by a number of ‘b’s calculated by dividing the number of ‘a’s by 3
  - (e.g. a^4b, a^6, $a^{11}b^2$, ...)
- $\{a^i c^j : i \geq 3, j \geq 0\} \cup \{a^i b^j : 1 \leq i < 3, j \geq 0\}$
  - can also use $\cap$

Defining a Language

- There are 3 ways to define a language
  - Set Definitions
  - Decision Programs
  - Grammars

Decision Programs

- You input a string and the program tells you if the string belongs to the language
- The language specified by the decision program is the set of all the strings for which the program says “YES”
- A compiler is partly a decision program for the corresponding programming language

Example of a Decision Program

```
1:   read(sym);          {assume read just gives us the next symbol in the string being examined}
         case sym of
               eos:        goto N;         {assume read returns special symbol “eos” if at end of string}
               “a”: goto 2;
               “b”         goto N;
               “c”         goto N;
         endcase;                  {case statement selects between alternatives as in Pascal}
    2:   read(sym);
         case sym of
               eos:        goto Y;         {if we get here we have a string of one a which is OK}
               “a”: goto 3;
               “b”: goto 6;        {we can have a b after one a}
               “c”: goto N         {any cs must follow three or more as - here we’ve only had one}
         endcase;
    3:   read(sym);
         case sym of
               eos: goto Y;        {if we get here we’ve read a string of two as which is OK}
               “a”: goto 4;
               “b”: goto 6;
               “c”: goto N;        {any cs must follow three or more as - here we’ve only had two}
         endcase;
    4:   read(sym);
         case sym of
               eos: goto Y;        {if we get here we’ve read a string of three or more as which is OK}
               “a”: goto 4;        {we loop here because we allow any number of as ? 3}
               “b”: goto N;        {b can only follow one or two as}
               “c”: goto 5;        {cs are OK after three or more as}
         endcase;
    5:   read(sym);
         case sym of
               eos: goto Y;        {if we get here we’ve read ? 3 as followed by ? 1 cs which is OK}
               “a”: goto N;        {as after cs are not allowed}
               “b”: goto N;        {bs are only allowed after one or two as}
               “c”: goto 5;        {we loop here because we allow any number of cs after ? 3 as}
         endcase;
    6:   read(sym);
         case sym of
               eos: goto Y;        {we get here if we’ve read 1 or 2 as followed by ? 0 bs - OK}
               “a”: goto N;        {no as allowed after bs}
               “b”: goto 6;        {we loop here because we allow any number of bs after 1 or 2 as}
               “c”: goto N;        {no cs are allowed after bs}
         endcase;
    Y:   write(“yes”);
         goto E;
    N:   write(“no”);
         goto E;
    E:   {end of program}
```

$$\{a^i c^j : i \geq 3, j \geq 0\} \cup$$
$$\{a^i b^j : 1 \leq i < 3, j \geq 0\}$$

https://pixabay.com/photos/tongyeong-nature-landscape-bench-2384216/