Lecture 12: Strings and Recursion

Dr John Levine

CS103 Machines, Languages and Computation
November 6th 2015

University of Strathclyde Science

The Second Half of the Class

- Week 7: Recursion and Languages: how to use recursion to allow for potentially unbounded sentences.
- Week 8: Recursion and Strings: how to specify an infinite set of strings using a finite set of rules.
- Week 9: Recursion and Logical Definitions: what does it mean to be someone's ancestor?
- Week 10: The λ-calculus: how can we create and use computable functions using only symbols?
- Week 11: Recursion and Function Definitions: how can we use recursion to create functions?

University of Strathclyde Science

The Second Half of the Class

- Week 7: Recursion and Languages: how to use recursion to allow for potentially unbounded sentences.
- Week 8: Recursion and Strings: how to specify an infinite set of strings using a finite set of rules.
- Week 9: Recursion and Logical Definitions: what does it mean to be someone's ancestor?
- Week 10: The λ-calculus: how can we create and use computable functions using only symbols?
- Week 11: Recursion and Function Definitions: how can we use recursion to create functions?

University of Strathclyde Science

What is Recursion?

- Recursion is a method of defining structures in which the structure being defined may be used within its own definition.
- The term is also used more generally to describe a process of repeating objects in a self-similar way.
- Recursion leads to “nested” structures. Sometimes the nature of the nesting is clear, but often we have to look hard to find it and encode the recursive process.
- Recursion can be found in many places: in languages, logical definitions, data structures, programs, …

University of Strathclyde Science

Context-Free Grammars

- Most computer languages can be specified used context-free grammars
- Rules are of the form:

  Symbol → list of Symbols and Terminal Symbols

  e.g.

  Exp → “if” Cond “then” Exp
  Exp → “print” Var
  Cond → Var “=” Value
  Var → “a” | “b” | “c” | … | “y” | “z” (| means “or”)
  Value → “0” | “1”

University of Strathclyde Science

Coping with Repeating Structure

Exp → “if” Cond “then” Exp
Exp → “print” Var
Cond → Var “=” Value
Var → “x” | “y” | “z”
Value → “0” | “1”

- This grammar can generate repeating structures:

```
if a=0 then
        if b=1 then
                if c=0 then
                        if d=0 then …
```

University of Strathclyde Science

Coping with Repeating Structure

Exp → “if” Cond “then” Exp
Exp → “print” Var
Cond → Var “=” Value
Var → “x” | “y” | “z”
Value → “0” | “1”

- This grammar can generate repeating structures:

```
if a=0 then
        if b=1 then
                if c=0 then
                        if d=0 then …
```

University of Strathclyde Science

A Larger Grammar

Sentence → NounPhrase Verb2 NounPhrase
NounPhrase → Name
NounPhrase → Article NounGroup
NounGroup → Noun
NounGroup → Adjective Noun

Name → “Brian” | “Beryl”
Verb2 → “ate” | “fed” | “chased” | “kissed”
Article → “a” | “the”
Adjective → “big” | “purple” | “ugly” | “fat” | “eccentric”
Noun → “frog” | “dog” | “clock” | “monster” | “princess”

University of Strathclyde Science

Derivation of a Sentence

- We can derive sentences with grammars just like we did with the MIU-system:

  Sentence
  NounPhrase Verb2 NounPhrase
  Name Verb2 NounPhrase
  “Brian” Verb2 NounPhrase
  “Brian” “ate” NounPhrase
  “Brian” “ate” Article NounGroup
  “Brian” “ate” “a” NounGroup
  “Brian” “ate” “a” Adjective Noun
  “Brian” “ate” “a” “big” Noun
  “Brian” “ate” “a” “big” “frog” ■

University of Strathclyde Science

Assignment 6

- Generate some random sentences using the grammar.
- The grammar can generate “Brian ate a big frog” or “Brian ate a purple frog”.
- But I want it to generate “Brian ate a big purple frog”.
- Or even “Brian ate a big fat ugly purple eccentric frog”.
- Can you modify my grammar so it can generate noun phrases containing any number of adjectives, without increasing the number of rules?
- Reading: Chapter 5 of Gödel, Escher, Bach.

University of Strathclyde Science

Recursive Rules

- A rule is recursive if the same symbol appears on both sides of the rule:

  S → “a” S (recursive rule)
  S → “b” (non-recursive rule)

- Such rules allow unlimited looping to occur:

  S
  “a” S
  “a” “a” S
  “a” “a” “a” S
  “a” “a” “a” “a” S
  “a” “a” “a” “a” “b” ■

University of Strathclyde Science

Generating Strings

- Sometimes we want use our rules to generate strings, as in the MIU-system:

  $S \rightarrow aS$ (recursive rule)
  $S \rightarrow b$ (non-recursive rule)

- Such rules allow unlimited looping to occur:

  S
  aS
  aaS
  aaaS
  aaaaS
  aaaab ■

Regular Expressions

- We can describe infinite sets of strings using *regular expressions*.
- For example: M(I*U*)* is this set of strings:

  {M, MI, MU, MII, MIU, MUI, MUU, MIII, MIIU, MIUI, …}

- In a regular expression:
  X* means “X repeated zero or more times”
  X^+ means “X repeated at least once”
  X^n means “X repeated exactly n times”
  X (with no superscript) means “exactly one X”
- X can be a character or another regular expression

University of Strathclyde Science

Grammars and Languages

- We can use regular expressions to describe the set of strings generated by a grammar.
- Such a set of strings is often referred to as a *language*.
- For example:

$$S \rightarrow aS$$
$$S \rightarrow b$$

- Exercise: describe the language (i.e. the set of strings) generated by this grammar as a regular expression.
- Answer: a*b.

University of Strathclyde Science

Assignment 7

- Finish reading Chapter 5 of Gödel, Escher, Bach.
- Write a grammar which can generate strings of the form a^+b^+, e.g. ab, aaab, abbbb, abbb, but *not* aaa, bbb ($^+$ means “repeated at least once”).
- Write a grammar which can generate strings of the form $(a^+b^+)^+$, e.g. ab, aabb, aaaabbaabb, abababbb, but *not* aaaa, aabbaa, bbabbaaabb, etc.
- Can you write a grammar to can generate strings of the form a^nb^n, e.g. ab, aabb, aaabbb, aaaabbbb, but *not* abbb, aaab, aaaabb, etc?