Ludic Computing Course CO345

Tutorial 2

Visual Grammars

and some Applications

Simon Colton & Alison Pease
Computational Creativity Group
Department of Computing
Imperial College, London

ccg.doc.ic.ac.uk

Question 1

$\Sigma = \{c, q, t\}$

$N = \{S, X\}$

P: 1. $S \rightarrow XcSt$

2. $S \rightarrow tcq$

3. $X \rightarrow cX$

4. $X \rightarrow qq$

a. What will be the second word with no non-terminal symbols in produced by a systematic (breadth first) application of the given context free grammar to S?

b. Draw a parse tree for this grammar, starting with S. Do not expand any node with more than five letters in

c. Translating c to be a circle, q to be a square and t to be a triangle, write an equivalent CFDG file (don't worry about transformations). What would your answer to part a. be with a linear rendering of the shapes?

Answer 1

$\Sigma = \{c, q, t\}$

$N = \{S, X\}$

P: 1. $S \rightarrow XcSt$

2. $S \rightarrow tcq$

3. $X \rightarrow cX$

4. $X \rightarrow qq$

a. If we apply the production rules from P in a breadth first way, starting with S, we get:

$S \Rightarrow_1 XcSt$
$S \Rightarrow_2 tcq$
$XcSt \Rightarrow_1 XcXcStt$
$XcSt \Rightarrow_2 Xctcqt$
$XcSt \Rightarrow_3 cXcSt$
$XcSt \Rightarrow_4 qqcSt$
$XcXcStt \Rightarrow_1 XcXcXcSttt$
$XcXcStt \Rightarrow_2 XcXctcqtt$
$XcXcStt \Rightarrow_3 cXcXcStt$
$XcXcStt \Rightarrow_3 XccXcStt$
$XcXcStt \Rightarrow_4 qqcXcStt$
$XcXcStt \Rightarrow_4 XcqqcStt$
$Xctcqt \Rightarrow_3 cXctcqt$
$Xctcqt \Rightarrow_3 qqctcqt$

second word with no non-terminal symbols in it

Answer 1

$\Sigma = \{c, q, t\}$

$N = \{S, X\}$

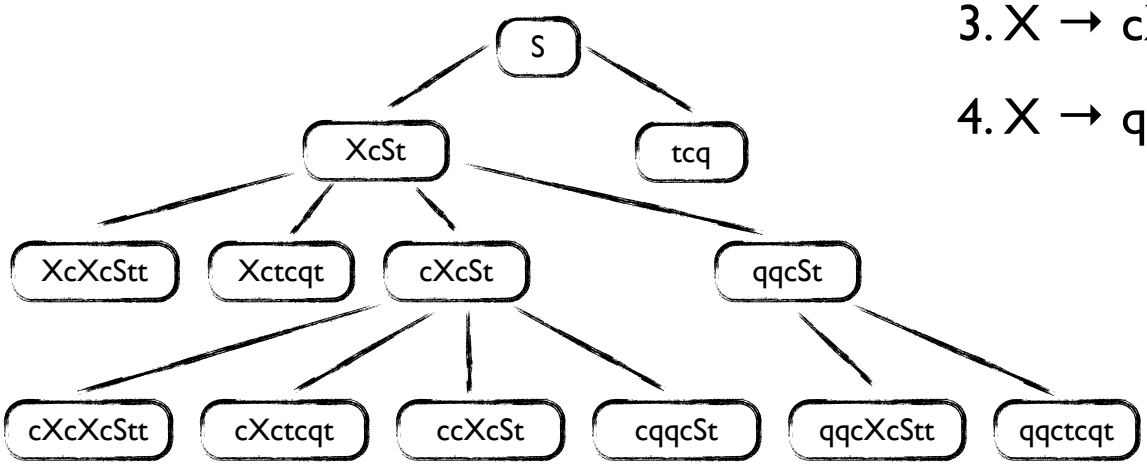
P: 1. $S \rightarrow XcSt$

2. $S \rightarrow tcq$

3. $X \rightarrow cX$

4. $X \rightarrow qq$

b.



Answer 1

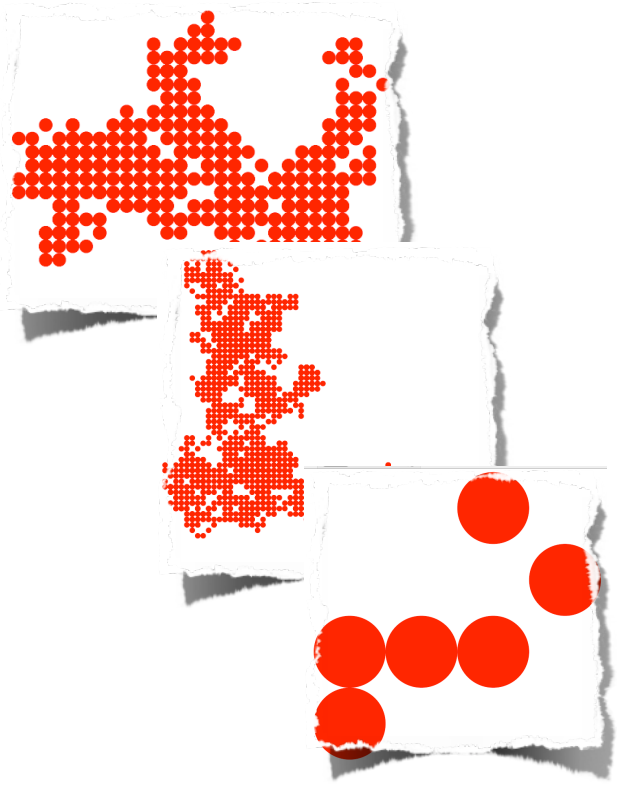
c. On the right is the equivalent CFDG file for the given context free grammar. The answer from part a. was: qqctcqt. This would be interpreted as: ■■●▲●■▲

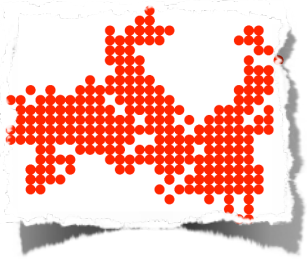
```
startshape S

rule S {
  X { }
  CIRCLE { }
  S { }
  TRIANGLE { }
}

rule S {
  TRIANGLE { }
  CIRCLE { }
  SQUARE { }
}

rule X {
  CIRCLE { }
  X { }
}

rule X {
  SQUARE { }
  SQUARE { }
}
```

Question 2

a. Write down a CFDG file which could produce these three designs (with different random seeds)

b. Use your answer to a. to explain how randomness is handled by the CFDG software





Answer 2

a. The grammar needs to iteratively place red circles in any of the eight adjacent positions. It also needs to stop on average after a non-trivial number of circles have been place, hence it will need a rule which has no non-terminal symbols in (we can use the empty rule for this). The file on the right produces these pictures.

b. The software uses an attached probability number to calculate the probability of each rule being used at any iteration. Note that each of the rules has a probability number of 100, except the final rule, which has 1. This means that the circle generating rules are all 100 times more likely to be used than the termination rule, hence a non-trivial number of circles gets placed, on average

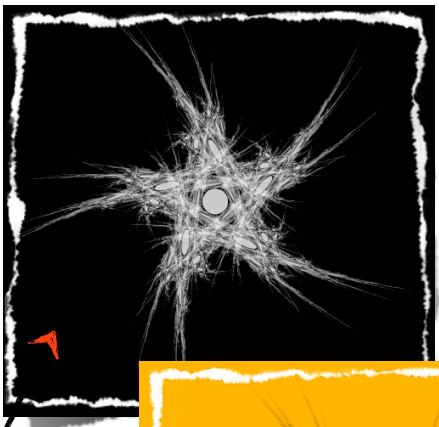
```
startshape CIRCLE_FILL

rule CIRCLE_FILL 100 {
    CIRCLE { hue 2 saturation 1 brightness 1 }
    CIRCLE_FILL { x 1 y 1}
}

rule CIRCLE_FILL 100 {
    CIRCLE { hue 2 saturation 1 brightness 1 }
    CIRCLE_FILL { x 0 y 1}
}

rule CIRCLE_FILL 100 {
    CIRCLE { hue 2 saturation 1 brightness 1 }
    CIRCLE_FILL { x 1 y 0 }
}

rule CIRCLE_FILL 100 {
    CIRCLE { hue 2 saturation 1 brightness 1 }
    CIRCLE_FILL { x -1 y -1 }
}

rule CIRCLE_FILL 100 {
    CIRCLE { hue 2 saturation 1 brightness 1 }
    CIRCLE_FILL { x 0 y -1 }
}

rule CIRCLE_FILL 100 {
    CIRCLE { hue 2 saturation 1 brightness 1 }
    CIRCLE_FILL { x -1 y 0 }
}

rule CIRCLE_FILL 100 {
    CIRCLE { hue 2 saturation 1 brightness 1 }
    CIRCLE_FILL { x -1 y 1 }
}

rule CIRCLE_FILL 100 {
    CIRCLE { hue 2 saturation 1 brightness 1 }
    CIRCLE_FILL { x 1 y -1 }
}

rule CIRCLE_FILL { }
```

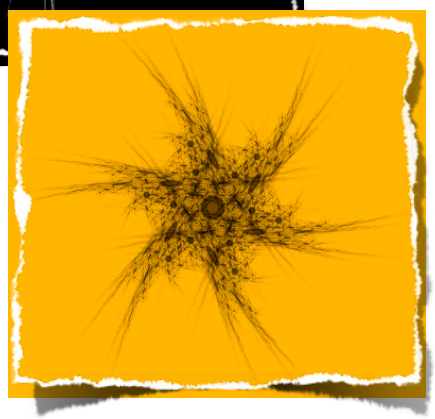
Question 3

a. Describe the way in which this design grammar generates shapes. Draw pictures if it helps.

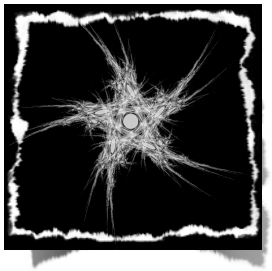
b. Describe how the CFDG software renders the shapes

c. Change the CFDG file to produce the second image on the right



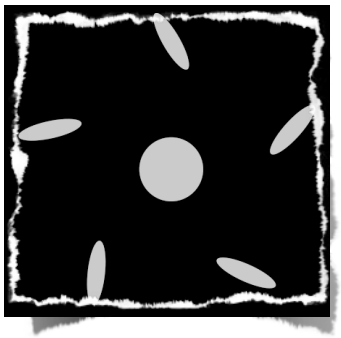


```
startshape ACE // by marksmit

background { brightness -1 }

rule ACE {
    CIRCLE{ brightness 1 alpha -0.3 }
    5* { rotate 72 } ACE { size 0.27 1.01 y 2 rotate 30 }
}
```

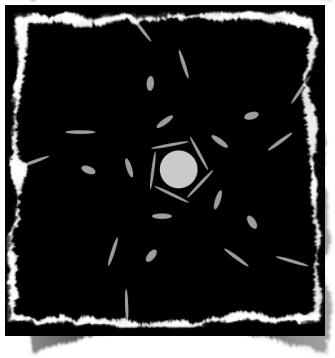


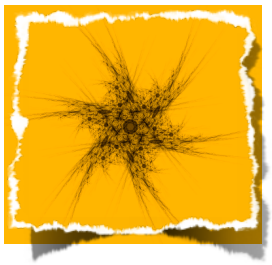
Answer 3

a. The grammar places a circle on the canvas, then stretches and rotates five circles, then places them around the edge. For each of the new ovals, it does the same, so the shape gets very stretched, which gives the picture its pointy arms



b. The software places each shape on the canvas, but when this is beyond the canvas boundaries, all the shapes are translated and reduced in size, so that the whole image fits on the canvas. This produces a zooming out effect





Answer 3

c. These alterations will produce the desired picture:

```
startshape ACE // by marksmit, altered by simoncolton

background { brightness 1 saturation 1 hue 36 }

rule ACE {
    CIRCLE{  brightness 0 alpha -0.3 }
    6* { rotate 60 } ACE { size 0.27 1.01 y 2 rotate 30 }
}
```

Changed the background colour to orange

Changed the number of circles to be placed around the border, which now gives six arms

Changed the rotation angle, so that the arms are equally placed around the circle

Change the foreground colour to be black

Question 4

a. What would be output from the first five iterations of this L-System?

b. If A was interpreted as a turtle command: "move forward 1cm" and B was interpreted as turn right 90 degrees, draw the five diagrams that would be produced by this interpretation. Start your turtle facing towards the top of the page

alphabet: {A B}
initiator: A

productions:
(A → AB)
(B → A)

Answer 4

a. n = 0 : A

n = 1 : AB

n = 2 : ABA

n = 3 : ABAAB

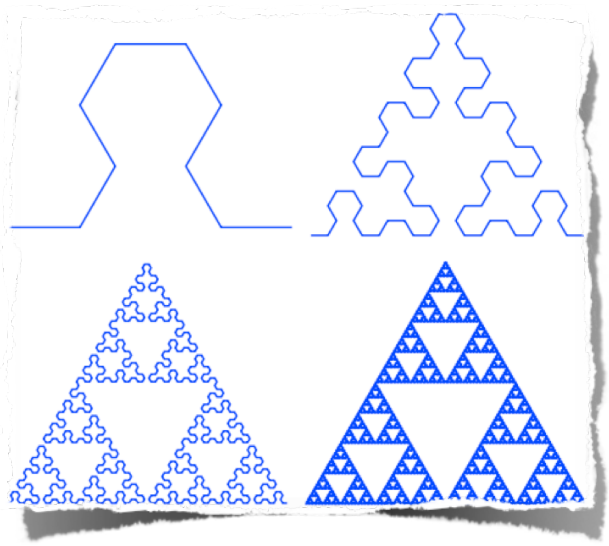
n = 4 : ABAABABA

This is the original grammar that Lindenmayer constructed to model the growth patterns of algae

b.

Question 5

a. If the diagrams on the right are iterations of an L-system when interpreted as commands for turtle graphics, what is the grammar, and what is the interpretation?



Answer 5

a. This is a modified version of the Sierpinski triangle. The grammar rules are as follows:

alphabet: {A, B, +, -}
initiator: A
productions:
(A → B−A−B)
(B → A+B+A)

To draw the picture, A and B must both be interpreted as "draw forward", + means "turn right by the given angle" and - means "turn left by the given angle". In the pictures, the angle is 60 degrees