Cog356 Final Exam, Part 2 - Formal Systems

Preliminary Remarks

This part of your final exam pertains to formal systems, particularly those that we focussed on this semester. That said, the questions on this part of the exam are not intended to be particularly intense. Rather, they are intended to provide, as a collective, an integrative consideration of some of the material that we focussed on this semester, thus affording you an opportunity to author an accessible story about formal systems and generative processes.

When asked for definitions, present the definitions that we adopted in the course this semester, and do it with precision. When asked to work in M-mode, do so with care, begin sure to adhere to the conventions associated with the particular formal system in which you are working. When asked to write paragraphs, do so like you care about the product of your writing and thinking.

In crafting your solution document for this part of the final exam, please include enough structure and content of the exam so that I will not have any problems grading your work. If I have to look too hard to find responses to questions, you simply won't get credit for those responses.

Section 1 - Definitions

Provide definitions for the following concepts that are consistent with the way that we considered them in class this semester (rather, than, say, some relatively arbitrary site on the web).

1. Define **Post production system**.

   `Answer goes here`

2. Define **Recursive transition network**.

   `Answer goes here`

3. Define **formal language**.

   `Answer goes here`

4. Define **grammar**.

   `Answer goes here`

5. Define **context free grammar**.

   `Answer goes here`

6. Define **the language generated by a context free grammar**.

Answer goes here

7. Define **L-System**.

   Answer goes here

Section 2 - Post Production Systems

Consider the following Post production system, which we will call NMD, short for "nested matching delimiters":

- Well-formed strings consist of nested matching parentheses, brackets, and braces. Thus, for example: the following are well-formed strings in the NMD system:
  1. `[{}]`
  2. `((((((((((()))))))))))`
  3. `(((([[{}]])))))`
- There are three axioms:
  1. `()`
  2. `[]`
  3. {}
- There are three rules:
  1. For any string x, if (x) is a therem, then $[(x)]$
  2. For any string x, if $[x]$ is a therem, then $\{[x]\}$
  3. For any string x, if $\{x\}$ is a therem, then $(\{x\})$

1. Write down the nine (9) shortest theorems in this this Post production system.

   Answer goes here

2. How many theorems are there in this Post production system?

   Answer goes here

3. Show that the following string is a theorem in this Post production system: $(\{[()]\})$

   Answer goes here

4. Argue that the following string is **not** a theorem in this Post production system: $([])$

   Answer goes here

Section 3 - Tracing the ELL RTNs

Trace execution of the set of ELL recursive transition networks that appear in Appendix 1 on the sentence: "the powerful creature protected the creepy marble castle" by simply writing down the complete sequence of labels corresponding to the flow of execution through the networks as the sentence is recognized.

Answer goes here

Section 4 - CFG for Naturalesque Music Language

For this question, please refer to the materials in Appendix 2, the appendix on NML.

Anayizing the CFG

1. Please list the set of terminal symbols in the given CFG.

   Answer goes here

2. Please list the set of nonterminal symbols in the given CFG.

   Answer goes here

3. Please identify the start symbol for the the given CFG.

   Answer goes here

4. Is the language generated by the CFG finite or infinite?

   Answer goes here

Derivations

For this question, please restrict yourself to LEFTMOST DERIVATIONS (in which you consistently replace the leftmost nonterminal in a string of symbols). And, of course, please don't forget that a derivation is a sequence of direct derivations!

1. Show that "play a descending run on g" is a "simplecommand" in the CFG by deriving "play a descending run on g" from "simplecommand".

   Answer goes here

2. Show that “f2 is play a hill on d” is a “definition” in the CFG by deriving “f2 is play a hill on d” from “definition”.

   Answer goes here

3. Show that “f1 f1 f2 f1” is an “idseq” in the CFG by deriving “f1 f1 f2 f1” from “idseq”.

   Answer goes here

4. Show that “f3 is f1 f1 f2 f1” is a “definition” in the CFG by deriving “f3 is f1 f1 f2 f1” from “definition”.

   Answer goes here

5. Show that “play a descending run on g .” is a “sentence” in the CFG by deriving “play a descending run on g .” from “sentence”.

   Answer goes here

6. Show that “play f3 where f1 is play on c , f2 is play a hill on d , f3 is f1 f2 f2 f1 .” is a “sentence” in the CFG by deriving “play f3 where f1 is play on c , f2 is play a hill on d , f3 is f1 f2 f2 f1 .” from “sentence”.

   Answer goes here

Short Answer Questions

Although the syntax of NML was presented formally, its semantics was not. Still, you should be able to infer much of the semantics from the given demo, enough to answer the following questions.

1. How many notes are played when a sentence consisting of a simple “bag” command is issued to the system?

   Answer goes here

2. How many notes are played when a sentence consisting of a simple “run” command is issued to the system?

   Answer goes here

3. Which of the following simple commands is nondeterministic?

   (a) the bag command

   (b) the ascending run command

   (c) the descending run command

   (d) the run command

   Answer goes here

4. In JFugue mode, what will be produced when the following sentence is issued to the system:
   `>>> Play a bag on g.`

Answer goes here

5. In JFugue mode, what will be produced when the following sentence is issued to the system:
   `>>> Play a descending run on g.`

   Answer goes here

6. Write down a sentence in NML that will produce the following JFugue output:
   `JFugue:  C5Q C5Q C5H`

   Answer goes here

7. Write down a sentence in NML that will produce the following JFugue output:
   `JFugue:  C5Q D5Q C5H`

   Answer goes here

8. Write down a sentence in NML that will produce the following JFugue output:
   `JFugue:  D5Q E5Q D5H`

   Answer goes here

9. Write down a sentence in NML that will produce the following JFugue output:
   `JFugue:  F5Q E5Q D5H`

   Answer goes here

10. Write down a sentence in NML that will produce the following JFugue output:
    `JFugue:  C5Q D5Q C5H D5Q E5Q D5H F5Q E5Q D5H C5Q C5Q C5H`

    Answer goes here

Reflective Tasks

Generally speaking, one can find both good and bad things to say about virtually any formal modeling mechanism, whether it is a general purpose mechanism or a domain specific mechanism. With this in mind, please address each of the following tasks, taking just one paragraph to respond to each:

1. Please do your best, in a measured, insightful manner, to say something **positive** about CFGs, generally speaking, as a mechanism for capturing the syntax of languages.

   Answer goes here

2. Please do your best, in a measured, insightful manner, to say something **negative** about CFGs, generally speaking, as a mechanism for capturing the syntax of languages.

Answer goes here

3. Please do your best, in a measured, insightful manner, to say something **positive** about NML as a mechanism for describing sequences of musical notes.

   Answer goes here

4. Please do your best, in a measured, insightful manner, to say something **negative** about NML as a mechanism for describing sequences of musical notes.

   Answer goes here

Section 5 - Framing Markov Processes as Formal Systems

Background

The way that Markov processes were presented to you this semester, which is consistent with the way that they tend to be presented, may leave you wondering whether or not they actually fall within the realm of formal systems. After all, CFGs and L-systems and Post production systems all have rules (if-then rules; productions) which map something to something else, rules that are used to generate "special" strings (sentences or generations or theorems).

Although the Markov process computations took the form of repeated state transitions made by referencing a probability distribution table, given a random number, we might reconceive the computation of a valued sequence in terms of rules. Breaking that down a bit:

1. Think of a **sequence of values** generated by a Markov process as the analog to a **sentence**, or **generation**, or **theorem** in CFGs or L-systems or Post production systems.
2. Imagine creating **one rule for each state (perhaps including an artificial preliminary state)** by:
   (a) Creating the left hand side using (1) a random number, and (2) the current state.
   (b) Creating the right hand side to determine the next state by referencing the random number and the row of the probability distribution matrix indexed by the current state.

Tasks

Think way back to the Markov process treatment of the Traveling Salesman problem that we considered near the start of the semester. With the details of the problem in mind (look back to the relevant lessons for the details):

1. Write down **three** sequences that might be viewed as the Markov process analog to sentences/generations/theorems.

   Answer goes here

2. Write down **four** "if-then" rules, one for mapping city A to another city, one for mapping city B to another city, one for mapping city C to another city, and one for mapping the artifical initial state to one of the three cities.

   Answer goes here

Section 6 - Logic for Problem Solving

Some Definitions

1. What is an **interpretation** of a formula?

   Answer goes here

2. What does it mean for a formula to be **valid**?

   Answer goes here

3. What does it mean for a formula to be **inconsistent**?

   Answer goes here

4. What does it mean for two formulas to be **logically equivalent**?

   Answer goes here

Validity and Inconsistency

For each of the following formulas, indicate the words (all of the words) that fit the formula:

1. $((P \rightarrow Q) \wedge (P \wedge (\sim Q)))$
   - valid
   - invalid
   - inconsistent
   - consistent

   Answer goes here

2. $(P \vee (Q \rightarrow (\sim P)))$
   - valid

- invalid
- inconsistent
- consistent

Answer goes here

3. (((P $\rightarrow$ Q) $\wedge$ P) $\rightarrow$ Q)
   - valid
   - invalid
   - inconsistent
   - consistent

   Answer goes here

Normal Forms

1. Any well formed formula can be converted to a normal form.
   - true
   - false

   Answer goes here

2. Convert ((P $\vee \sim$ Q) $\rightarrow$ R) to disjunctive normal form (DNF). Do so by means of writing a sequence of logically equivalent forms, and citing the reason for each step taken in the transformational sequence of steps.

   Answer goes here

General Concepts

1. A literal is an atomic formula or its negation.
   - true
   - false

   Answer goes here

2. A clause is a disjunction of literals.
   - true
   - false

   Answer goes here

3. A Horn clause is a clause with exactly one positive literal.
   - true
   - false

   Answer goes here

4. Prolog is a Horn clause problem solver, which means that it operates in the context of a set of Horn clauses.

- true
- false

Answer goes here

Resolution

1. State the **resolution principle**.

   Answer goes here

2. Define what is meant by **resolution deduction**

   Answer goes here

3. Show that `p` logically follows from the wffs `( ( q` $\vee$ `t ) ` $\rightarrow$ `p )` and `( r` $\rightarrow$ `q )` and `r` and `t` by means of a resolution deduction involving the following Horn clauses:
   (a) $\sim$ q $\vee$ $\sim$ t $\vee$ p
   (b) $\sim$ r $\vee$ q
   (c) r
   (d) t
   (e) $\sim$ p

   Answer goes here

Section 7 - A Short Take on L-Systems

Imagine that a first year cognitive science student heard you mention L-systems in passing, and asked you to tell her a bit about them. Write one paragraph to reflect what you would say to her, which incorporates: (1) who invented L-systems, when, and for what purpose, (2) what L-systems can be used for, (3) something that you find particularly interesting about L-systems. The idea is to present the student with an abstract understanding of L-systems, but in a way that will entice them to want to learn more about L-systems.

Answer goes here

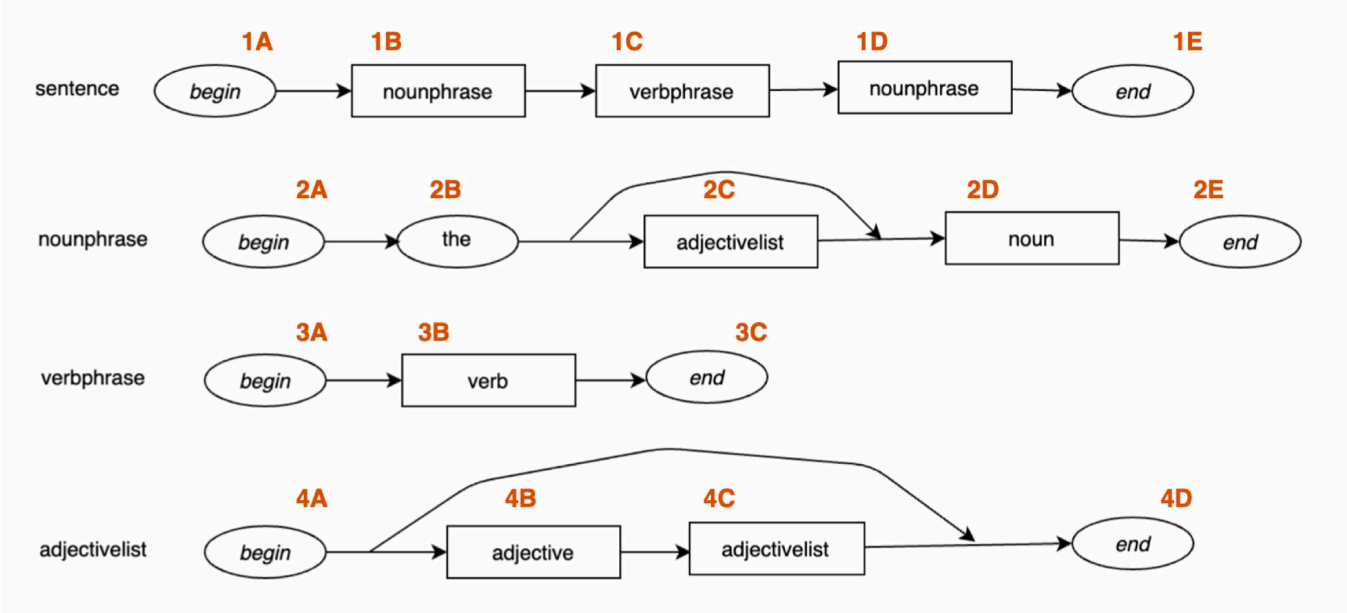
Section 8 - A Short Take on the Lambda Calculus

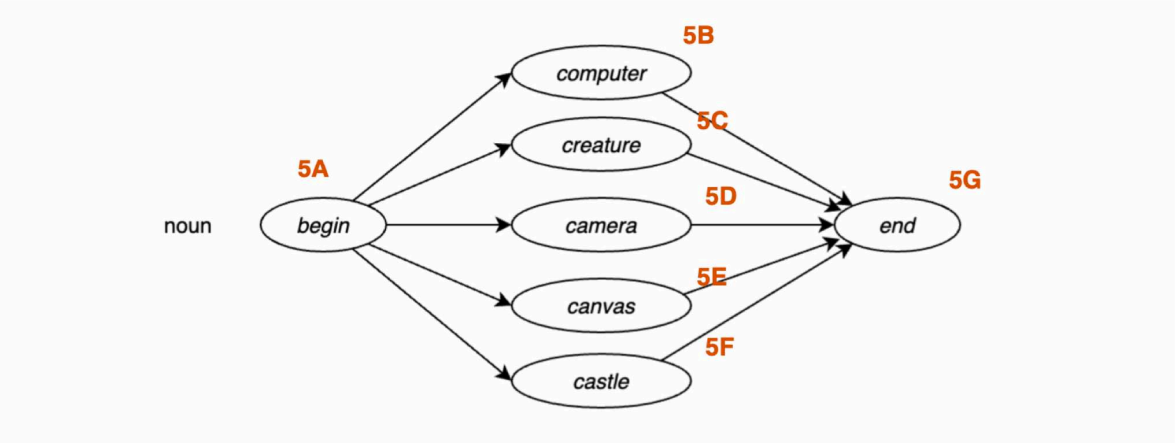
Imagine that a first year cognitive science student heard you mention the Lambda calculus in passing, and asked you to tell her a bit about them. Write one paragraph to reflect what you would say to her, which incorporates: (1) who invented the Lambda calculus, when, and for what purpose, (2) what the Lambda calculus can be used for, (3) something that you find particularly interesting about the Lambda calculus. The idea is to present the student with an abstract understanding of the Lambda calculus, but in a way that will entice them to want to learn more about

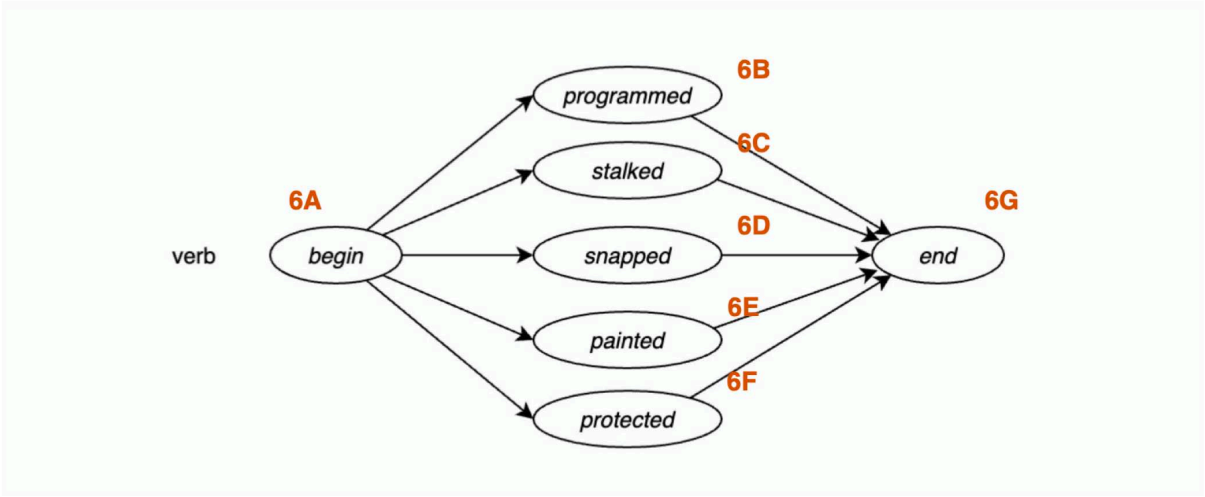
the Lambda calculus.

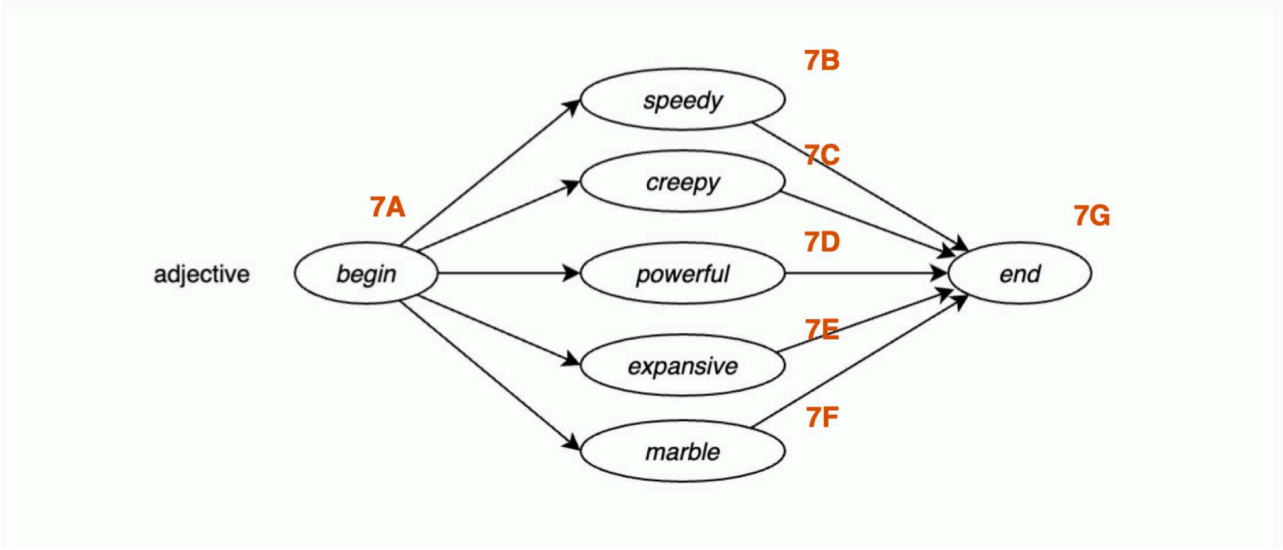
Answer goes here

Appendix 1: ELL RTNs






verb
6A
begin
6B
programmed
6C
stalked
6D
snapped
6E
painted
6F
protected
6G
end


adjective
7A
begin
7B
speedy
7C
creepy
7D
powerful
7E
expansive
7F
marble
7G
end

Appendix 2: Naturalesque Music Language, NML

This appendix describes a small language, extracted from a somewhat larger language, for describing musical melodies in a natural language like manner. The language presented here is called NML, and can be thought of as a bit of "pseudo natural" language for describing musical melodies.

The CFG

```
(1)  sentence --> simplecommand .
(2)  sentence --> complexcommand .
(3)  sentence --> query ?

(4)  simplecommand --> play on notename
(5)  simplecommand --> play a bag on notename
(6)  simplecommand --> play a hill on notename
(7)  simplecommand --> play an ascending run on notename
(8)  simplecommand --> play a descending run on notename
(9)  simplecommand --> play a run on notename

(10) complexcommand --> play id where defseq

(11) query --> how many notes were played
(12) query --> how many notename notes were played
(13) query --> what was the distribution of the notes played

(14) defseq --> definition
(15) defseq --> defseq , definition

(16) definition --> id is simplecommand
(17) definition --> id is idseq

(18) id --> f1
(19) id --> f2
(20) id --> f3
(21) id --> f4
(22) id --> f5
(23) id --> f6
(24) id --> f7
(25) id --> f8
(26) id --> f9

(27) idseq --> id
(28) idseq --> idseq id

(29) notename --> a
(30) notename --> b
(31) notename --> c
(32) notename --> d
(33) notename --> e
(34) notename --> f
```

```
(35) notename --> g
```

Demo

The following demo illustrates an interpreter for NML. For this particular demo, imagine that the "JFugue" mode has been established, so that playing a sequence of notes amounts to displaying a representation of the sequence of notes in JFugue notation. (Alternate modes are "ABC", in which case the display of the sequence of notes is in ABC notation, "sonic", in which case the sequence can be heard as it is played, and "MIDI", in which case the sequence of notes is written to a MIDI file.)

```
>>> Play on c.
JFugue: C5W

>>> Play a bag on c.
JFugue: C5Q C5Q C5H

>>> Play a bag on d.
JFugue: D5Q D5Q D5H

>>> Play a hill on c.
JFugue: C5Q D5Q C5H

>>> Play a hill on d.
JFugue: D5Q E5Q D5H

>>> Play an ascending run on e.
JFugue: E5Q F5Q G5H

>>> Play a descending run on g.
JFugue: G5Q F5Q E5H

>>> Play a run on f.
JFugue: F5Q E5Q D5H

>>> Play a run on f.
JFugue: F5Q E5Q D5H

>>> Play a run on f.
JFugue: F5Q G5Q A5H

>>> Play a run on f.
JFugue: F5Q E5Q D5H

>>> Play f3 where f1 is play on c, f2 is play a hill on d, f3 is f1 f2 f2 f1.
JFugue: C5W D5Q E5Q D5H D5Q E5Q D5H C5W

>>> How many notes were played?
8

>>> How many c notes were played?
2
```

```
>>> How many d notes were played?
4

>>> How many e notes were played?
2

>>> what was the distribution of the notes played?
C notes played: 2
D notes played: 4
E notes played: 2

>>> Play f7 where f1 is play an ascending run on c, f2 is play a hill on d,
f3 is play a bag on c, f4 is play f1 f1 f2 f1, f5 is play a descending run on g,
f6 is play f5 f5 f5 f2, f7 is f4 f4 f6 f4.
JFugue: C5Q D5Q E5H C5Q D5Q E5H D5Q E5Q D5H C5Q C5Q C5H
C5Q D5Q E5H C5Q D5Q E5H D5Q E5Q D5H C5Q C5Q C5H
G5Q F5Q E5H G5Q F5Q E5H G5Q F5Q E5H D5Q E5Q D5H
C5Q D5Q E5H C5Q D5Q E5H D5Q E5Q D5H C5Q C5Q C5H

>>>
```