UNIVERSITY OF YORK

BA Degree Examinations 2003-4

Department of Language and Linguistic Science

**L334 Computational Syntax and Semantics**

Time allowed: ONE hour

Answer ALL questions

(1) Given the following grammar,

| | |
|---|---|
| S → NP VP | *Kim*: NP |
| NP → D N | *the*: D |
| VP → V NP | *dog*: N |
| PP → P NP | *in*: P |
| NP → NP PP | *park*: N |
| | *saw*: V |

a. Provide a bottom-up, left-to-right derivation for the string *Kim saw the dog in the park*. (9 marks)

| | | |
|---|---|---|
| Kim saw the dog in the park | ⇐ NP saw the dog in the park | (Kim : NP) |
| | ⇐ NP V the dog in the park | (V : saw) |
| | ⇐ NP V D dog in the park | (D ; the) |
| | ⇐ NP V D N in the park | (N ; dog) |
| | ⇐ NP V NP in in the park | (NP → D N) |
| | ⇐ NP V NP P the park | (P ; in) |
| | ⇐ NP V NP P D park | (D ; the) |
| | ⇐ NP V NP P D N | (N ; park) |
| | ⇐ NP V NP P NP | (NP → D N) |
| | ⇐ NP V NP P NP | (NP → D N) |
| | ⇐ NP V NP PP | (PP → P NP) |
| | ⇐ NP V NP | (NP → NP PP) |
| | ⇐ NP VP | (VP → V NP) |
| | ⇐ S | (S → NP VP) |

b. State what kind of derivation it is. (1 mark)

A rightmost derivation in reverse

c. What problem would a top-down parser encounter in parsing the string *Kim saw the dog in the park* as defined by the above grammar, and why? (10 marks)

The grammar is left-recursive (because of the rule NP → NP PP). Therefore, a top-down, left-to-right parser, if forced to find all parses, will enentually encounter infinite search and fail to terminate.

(2) Under what circumstances would two grammars $G_1$ and $G_2$ be said to be *weakly equivalent*?

When they define the same language (i.e. $\mathcal{L}(G_1) = \mathcal{L}(G_2)$), even though they may assign it different analyses.

(3) a. What kind of language is $a^n b^n$?

Context free, or Type 2 in the Chomsky hierarchy

b. How is this class of languages defined?

$A \longrightarrow \alpha$; by rules with a single non-terminal on the left hand side and a string of terminals and non-terminals on the right hand side

c. What class of languages does the language defined by the following grammar belong to?

S → a S
S → a
S → b S
S → b

It is a regular language (left-regular, to be precise), or a Type 3 language in the Chomsky hierarchy.

d. What is the relationship between the classes of languages referred to in questions (a) and (c) above?

Regular languages are a proper subset of the context free languages.

(4) a. What is the difference between a recogniser and a parser? (4 marks)

A recogniser simply states whether a given string of words is well-defined with respect to a grammar. A parser additionally assigns a structure to each string.



Continued

b. What is the significance of this difference for the time complexity results for context-free languages? (4 marks)

The (worst case) time complexity for recognition of context free languages is $O(n^3)$ (proportional to the cube of the length of the input string). The (worst case) time complexity for parsing of context free languages is exponential.

(5) Outline the operation of a shift-reduce parser and show the states of the stack and buffer during a parse of the string *Kim saw the dog* using the grammar in question (1) above.

A shift-reduce parser is a bottom-up, typically left-to-right, parser. The parser uses two data structures: 1. a buffer that contains the unprocessed part of the input string and 2. a stack (first-in last-out store) on which the parse is assembled. The parser carries out two operations: 1. shift, which takes an item off the input buffer and puts it on the top of the stack and 2. reduce, which matches the top of the stack against the right hand side of the rules of the grammar and which, if it finds a match, replaces the matching items on the top of the stack with the left hand side of the rule. The parser starts off with the stack empty and terminates when the input buffer is empty and there is a single item on the stack.

| Stack | Buffer | Operation | Rule |
|---|---|---|---|
| [] | <Kim saw the dog> | | |
| [Kim] | <saw the dog> | Shift | |
| [NP] | <saw the dog> | Reduce | (NP : Kim) |
| [NP saw] | <the dog> | Shift | |
| [NP V] | <the dog> | Reduce | (V : saw) |
| [NP V the] | <dog> | Shift | |
| [NP V D] | <dog> | Reduce | (D : the) |
| [NP V D dog] | < > | Shift | |
| [NP V D N] | < > | Reduce | (N : dog) |
| [NP V NP] | < > | Reduce | (NP → D N) |
| [NP VP] | < > | Reduce | (VP → V NP) |
| [S] | < > | Reduce | (S → NP VP) |

(6) a. How does a Definite Clause Grammar (DCG) differ from a CF-PSG?

The category symbols in a DCG may take arguments.

b. What is the weak generative capacity of DCGs?

It is Chomsky Type 0, the recursively enumerable languages.

c. Write a minimal DCG in either BH or Prolog Grammar Rule Notation (making clear which you are using) that will generate the following sentences of English. Give arguments (both constants and variables) informative names.

i. These children are smart
ii. This child is smart
iii. The child is smart
iv. The children are smart
v. A child is smart
vi. The sheep are smart
vii. The sheep is smart

Grammar in BH Grammar Rule Notation:

```
s  ---> [np(Number), vp(Number)].
np(Number) ---> [d(Number), n(Number)].
vp(Number) ---> [v(Number), adj].

n(singular) ===> [child].
n(plural) ===> [children].
n(_Number) ===> [sheep].
d(singular) ===> [this].
d(plural) ===> [these].
d(_Number) ===> [the].
v(singular) ===> [is].
v(plural) ===> [are].
adj ===> [smart].
```

Grammar in Prolog Grammar Rule Notation:

```
s  --> np(Number), vp(Number).
np(Number) --> d(Number), n(Number).
vp(Number) --> v(Number), adj.

n(singular) --> [child].
n(plural) --> [children].
n(_Number) --> [sheep].
d(singular) --> [this].
d(plural) --> [these].
d(_Number) --> [the].
v(singular) --> [is].
v(plural) --> [are].
adj --> [smart].
```

**Total marks: 70**

End of examination