063173

THE UNIVERSITY *of York*

B.Sc., B.Eng. and M.Eng. Examinations 1996

DEPARTMENT OF COMPUTER SCIENCE

Part IIIb

Natural Language Processing

Time allowed: **Three hours**

Silent non-programmable calculators may be used.
Candidates should answer **four** questions from this paper.

1 (25 marks)

(i) [4 marks] **Briefly** explain what is meant by the following terms:

(a) [2 marks] Filler-gap dependencies; and

(b) [2 marks] Unbounded dependencies.

(ii) [6 marks] The following Definite Clause Grammar (given in Prolog-GRN) uses a *single* argument to indicate the presence or absence of a gap within its categories.

```
s(GapInfo) --> np(nogap), vp(GapInfo).
s(nogap)   --> np(nogap), aux, s(npgap).
s(nogap)   --> pp(nogap), aux, s(ppgap).

vp(GapInfo) --> vd, np(nogap),   np(GapInfo).
vp(GapInfo) --> vd, np(GapInfo), pp(nogap).
vp(GapInfo) --> vd, np(nogap),   pp(GapInfo).

pp(GapInfo) --> p, np(GapInfo).
pp(ppgap)   --> [].

np(nogap) --> [punch].
np(nogap) --> [sausages].
np(nogap) --> [who].
np(nogap) --> [what].
np(npgap) --> [].

p   --> [to].
aux --> [did].
vd  --> [give].
vd  --> [gave].
```

(a) [2 marks] Draw the phrase-structure tree that this grammar assigns to the sentence:

"To who did Punch give sausages?"

taking the root category to be `s(nogap)`.

(b) [4 marks] **Briefly** explain how this grammar enforces the constraint that there can only be a gap in a verb phrase when the sentence begins with a preposed filler of the same category as the gap.

(iii) [15 marks]

(a) [13 marks] Give a Definite Clause Grammar (in Prolog-GRN), based on the one given in part (ii), which

- uses *gap threading* to enforce the filler-gap dependencies, **and** which
- uses an extra argument to some or all of the categories to enforce the proper syntactic distribution of *wh*-noun phrases (e.g. the following is ungrammatical

  **"To Punch did who give sausages?"*

  but *is* generated by the grammar given in part (ii) and should *not* be generated by your revised grammar).

(b) [2 marks] Draw the phrase-structure tree that your grammar assigns to the sentence

"To who did Punch give sausages?"

taking the root category to be your grammar's equivalent to `s(nogap)`.

2 (25 marks)

This question refers to the Prolog implementation of a (grammar-specific) CF-PSG parser given below. Lines of the program are numbered for ease of reference from the question. You should also use them in your answers.

```
100 parse(Input) :- parse([], [s], Input, []).
110
120 parse(Xs, Xs, Ys, Ys).
130
140 parse(Xs, XFs, Ys, YFs) :-
150   move(Xs, X1s, Ys, Y1s),
160   parse(X1s, XFs, Y1s, YFs).
170
180 move([vp, np|Xs],    [s|Xs],    Ys, Ys).
190 move([vi|Xs],        [vp|Xs],   Ys, Ys).
200 move([nbar, det|Xs], [np|Xs],   Ys, Ys).
210 move([n|Xs],         [nbar|Xs], Ys, Ys).
220
230 move(Xs, [vi|Xs],  [cried|Ys], Ys).
240 move(Xs, [vi|Xs],  [died|Ys],  Ys).
250 move(Xs, [det|Xs], [the|Ys],   Ys).
260 move(Xs, [n|Xs],   [boy|Ys],   Ys).
```

(i) [$2\frac{1}{2}$ marks] Is the Prolog parser given above a *bottom-up* parser or a *top-down* parser? Explain your answer in detail, with reference to the code.

(ii) [$2\frac{1}{2}$ marks] Does the parser use a *depth-first* or *breadth-first* search strategy? Explain your answer in detail, with reference to the code.

(iii) [3 marks] Define what is meant by the terms *local ambiguity* and *global ambiguity*. Give an example of each.

(iv) [9 marks]

(a) [4 marks] Here are more lines of the program given earlier. These extra lines extend the syntactic coverage of the system.

```
270 move([vi, comp|Xs],  [rel|Xs],  Ys, Ys).
280 move([rel, nbar|Xs], [nbar|Xs], Ys, Ys).
290 move(Xs, [comp|Xs], [that|Ys], Ys).
```

The extended system can now parse the sentence

> *"The boy that cried died."*

Explain carefully why the phrase *"the boy"* is locally ambiguous within this sentence.

(b) [5 marks] The programmer now replaces line 200 by two lines (200 and 205) as follows:

```
200 move([nbar, det|Xs], [np|Xs], [cried|Ys], [cried|Ys]).
205 move([nbar, det|Xs], [np|Xs], [died|Ys],  [died|Ys]).
```

Explain **in detail** how these changes introduce a mechanism for avoiding the fruitless search caused by the local ambiguity described in part (iv)(a).

(v) [8 marks]

(a) [3 marks] Explain the difference between the following three types of parser:

- all-path/all-parse parsers,
- one-parse parsers, and
- one-path parsers.

(b) [5 marks] In anticipation of further extensions to the program, particularly extensions that would introduce *global ambiguities*, the programmer turns the parser into a *one-path parser* by adding a Prolog cut to line 150, so that lines 140-160 become

```
140 parse(Xs, XFs, Ys, YFs) :-
150   move(Xs, X1s, Ys, Y1s), !,
160   parse(X1s, XFs, Y1s, YFs).
```

- Explain **in detail** why this is now a one-path parser ($2\frac{1}{2}$ marks); and
- Explain **in detail** what it is that will determine which particular path is chosen ($2\frac{1}{2}$ marks).

3 (25 marks)

(i) [5 marks]

(a) [2 marks] Explain, with examples, the differences between *term unification* and *graph unification*.

(b) [3 marks] In a Definite Clause Grammar, the grammar still has a *monadic CF-PSG backbone*. Explain what this means and exemplify the *advantages* of formalisms that *abandon* such a backbone.

(ii) [3 marks] The *Fundamental Principle* in chart parsing creates *new* edges; it does not modify existing edges. Explain carefully what goes wrong if, instead of spawning new edges, the Fundamental Principle *modifies* incomplete edges. (Use diagrams in your answer but restrict your attention to CF-PSG parsing.)

(iii) [19 marks] This part of the question uses the Prolog implementation of a chart parser for BH-GRN grammars shown on the next page. (It is simpler but less flexible than the one covered in the lectures.)

Lines of the program are numbered for ease of reference from the question. You should also use them in your answers.

Although it has not been shown, you should assume a standard definition of `append/3`, *viz* `append(L1, L2, L3)` is true if list `L3` is the result of concatenating list `L2` onto the end of list `L1`.

Your attention is drawn to lines 260-280, which check whether the edge at the front of the agenda is already on the chart. If it is, that edge is not added to the chart again.

```
parse(Input, Start) :-
   create_all_lexical_edges(0, 1, Input),
   findall(edge(0, 0, Start, RHS),
           (Start ---> RHS ),
           Edges),
   add_to_chart(Edges).

create_all_lexical_edges(I, J, []).

create_all_lexical_edges(I, J, [Word|Words]) :-
    \+ ((Cat ===> [Word]), \+ assert(edge(I, J, Cat, []))),
    K is J + 1,
    create_all_lexical_edges(J, K, Words).

add_to_chart([]).

add_to_chart([edge(VtxI, VtxJ, Cat, Need)|Edges]) :-
   edge(VtxI, VtxJ, Cat, Need), !,
   add_to_chart(Edges).

add_to_chart([edge(VtxI,VtxJ,ICat,[NCat|NCats])|Edges0]) :-
   assert(edge(VtxI, VtxJ, ICat, [NCat|NCats])),
   findall(edge(VtxJ, VtxJ, NCat, RHS),
           (NCat ---> RHS),
           Edges1),
   findall(edge(VtxI, VtxK, ICat, NCats),
           edge(VtxJ, VtxK, NCat, []),
           Edges2),
   append(Edges0, Edges1, Edges3),
   append(Edges3, Edges2, Edges),
   add_to_chart(Edges).

add_to_chart([edge(VtxJ, VtxK, CCat, [])|Edges0]):-
   assert(edge(VtxJ, VtxK, CCat, [])),
   findall(edge(VtxI, VtxK, ICat, NCats),
           edge(VtxI, VtxJ, ICat, [CCat|NCats]),
           Edges1),
   append(Edges0, Edges1, Edges),
   add_to_chart(Edges).
```


Continued.

(a) [5 marks] Which lines implement the 'seek' process in this chart parser. Does the parser use bottom-up seeking or top-down seeking. Explain your answer.

(b) [3 marks] Is the parser one whose search strategy is depth-first or breadth-first? Explain your answer in detail with reference to the code.

(c) [3 marks] We want to change the parser's search strategy so that it is the opposite of what you said it was in part (b).

Give the line numbers and the revised Prolog for each line that must change.

(d) [4 marks] Alf, a programmer, deletes lines 260-280. He thinks it would be better to maintain a set of vertex-category pairs that would record which categories have been *sought* at which vertices and use this to avoid 're-seeking'.

Explain **in detail** the *advantages* that Alf's modification will have over the original program.

(e) [4 marks] Betty, another programmer, tells Alf:

> "Your modification is silly. The original program carried out both subtree sharing *and* node packing. Your revised program only does subtree sharing."

Explain, with examples, what is meant by *subtree sharing* and by *node packing*.

4 (25 marks)

(i) [4 marks] Briefly explain what is meant by *sequential, syntax-driven semantic translation* and *interleaved, syntax-driven semantic translation.*

(ii) [7 marks] Give a direct Prolog implementation of a top-down parser, including interleaved, syntax-driven semantic translation (using an extra argument to each Prolog predicate) for the grammar and lexicon that are given in the *appendix* to this paper. Define any auxiliary predicates that you use.

The program you give as your answer might be one that is run as follows:

```
?- s(Translation, [punch, hit, judy], []).
Translation = hit(p, j)
```

(iii) [14 marks] The semantic translation of the sentence:

"Judy was hit by Punch."

is to be the same as the semantic translation of the sentence:

"Punch hit Judy."

Using this fact and any other information you need from the *appendix*, complete the following tables of information, stating any assumptions that you make in your answer.

Syntax rule	Semantic translation rule	Semantic type of LHS
VP $\rightarrow$ Vx VPp		
VPp $\rightarrow$ Vp PPb		
PPb $\rightarrow$ Pb NP		

Word : Category	Logic translation	Semantic type
was : Vx		
hit : Vp		$\langle \mathbf{e}, \langle \mathbf{e}, \mathbf{t} \rangle \rangle$
by : Pb		

5 (25 marks)

(i) [6 marks] **Briefly** explain what is meant by each of the following terms:

(a) [$1\frac{1}{2}$ marks] Truth-conditional semantics;

(b) [$1\frac{1}{2}$ marks] Model-theoretic semantics;

(c) [$1\frac{1}{2}$ marks] Compositionality; and

(d) [$1\frac{1}{2}$ marks] The rule-to-rule hypothesis.

(ii) [19 marks] A computational linguist has been given the grammar and lexicon that are given in the *appendix* to this paper.

She wants to add the following grammar rules:

$$\mathsf{VP} \rightarrow \mathsf{VP}\ \mathsf{PP}$$
$$\mathsf{PP} \rightarrow \mathsf{P}\ \mathsf{NP}$$

and a number of lexical entries such as those shown here:

$$\begin{array}{r}
\textit{on} : \mathsf{P} \\
\textit{in} : \mathsf{P} \\
\textit{London} : \mathsf{NP} \\
\textit{Tuesday} : \mathsf{NP}
\end{array}$$

But, she has to decide what logic translations (and, hence, model-theoretic denotations) to assign to the sentences she can now generate. She is comparing two options, illustrated below:

Sentence	Option 1	Option 2
"Punch hit Judy"	$hit1(p, j)$	$\exists e\ hit1(e, p, j)$
"Punch hit Judy on Tuesday"	$hit2(p, j, tue)$	$\exists e(hit1(e, p, j) \wedge day(e, tue))$
"Punch hit Judy on Tuesday in London"	$hit3(p, j, tue, ldn)$	$\exists e(hit1(e, p, j) \wedge day(e, tue) \wedge place(e, ldn))$

(a) [3 marks] Option (2) is said to use *event semantics*. Explain what is meant by this term.

(b) [7 marks] Describe **in detail** the advantages of option (2) over option (1).

(c) [9 marks] Revise the tables given in the *appendix* so that event semantics is used. (Hint: In option (2), $hit1$ is a 3-ary predicate; this affects the logic translation and semantic type that you need to give in your revised lexicon.)

N.B. You are being asked only to revise what is in the *appendix*. You are not being asked to give any semantic treatment for the new grammar rules and lexical entries given above.

6 (25 marks)

An A.I. planner is a system that proposes a sequence of actions which, if executed by a suitable robot, would transform a given initial world state into some desired goal state.

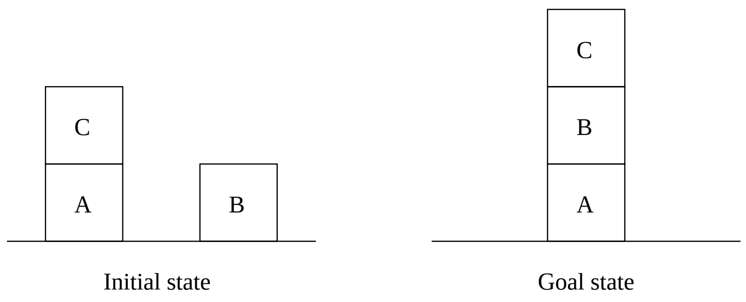
For example, if the robot's repertoire of operations includes the following for moving blocks around on a table:

`pickup(x)`	:	pick up block x
`place(x, y)`	:	place block x on y,

then an A.I. planner might propose the following sequence of actions:

```
pickup(c)
place(c, table)
pickup(b)
place(b, a)
pickup(c)
place(c, b)
```

to transform the depicted initial state into the depicted goal state:



A.I. planners proceed by search —trying out different sequences of actions until they find one that is successful.

Such planners have commercial applications, e.g. planning how to move cargo around a rail network to meet the demands for raw material of a number of factories served by the network.

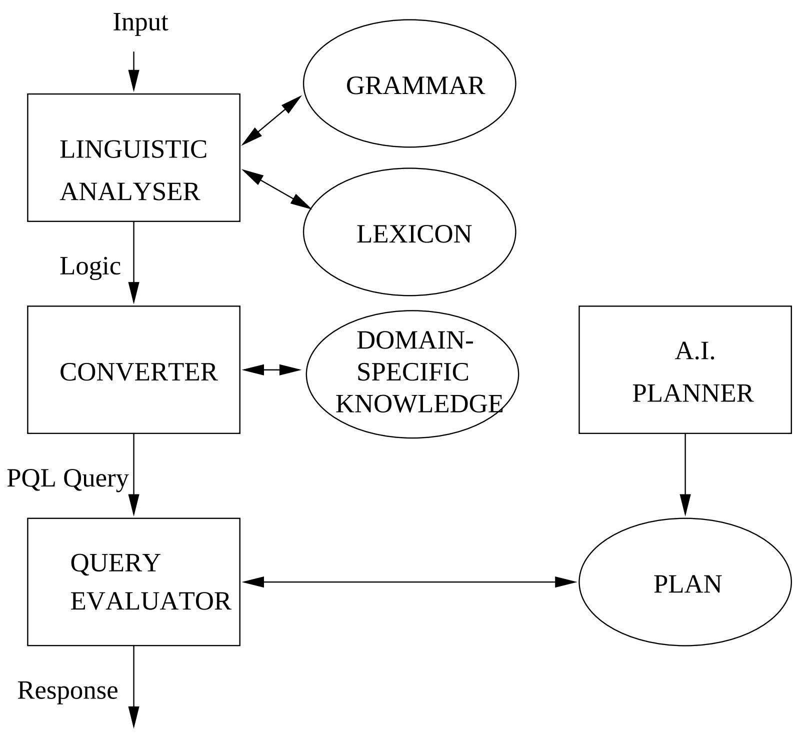
An interactive facility, enabling a user to query the plan or even contribute to the construction of the plan might be useful in certain cases.

Crouch & Pulman[1] report the development of a natural language interface to a planner. In this question, you reconstruct some of the reasoning that lies behind their design choices.

(i) [6 marks] Explain **in detail** why one might decide to use a natural language interface to an A.I. planning system.

In your answer, include the desirable criteria that *any* interface to a planner that is to be used in a business environment should satisfy, the reasons why natural language interfaces (in their ideal form) *do* satisfy these criteria, and the reasons why alternative interfaces do *not* satisfy all or some of these criteria. (By way of alternative interfaces, restrict your answer to menu-based interfaces and PQL, which is a formal language described by Crouch & Pulman and which you can think of as the plan-query-language-equivalent of the database query language, SQL.)

(ii) [19 marks] Crouch & Pulman use a modular, sequential architecture, similar to (though not identical to) that depicted below:



[1]Crouch, R.S. & S.G. Pulman: 'Time and modality in a natural language interface to a planning system', *Artificial Intelligence*, vol.63, pp.265-304, 1993

You can think of the linguistic analyser as comprising just a parser and a semantic translator, and making use of a general-purpose grammar and lexicon.

(a) [6 marks]

- **Briefly** describe the software engineering advantages that are supposed to accrue from using architectures similar to the one depicted above when building natural language interfaces to *databases*. (3 marks)
- For each advantage you gave, **briefly** explain whether the advantage is relevant to the construction of natural language interfaces to *planners*. (3 marks)

(b) [3 marks] A major problem with modular sequential architectures is disambiguation.

Crouch & Pulman exploit the fact that their system operates in a *limited domain* and is faced, therefore, only with utterances drawn from a *sublanguage*.

Explain these two terms and say how this helps disambiguation.

(c) [10 marks] A *generate-and-test* approach to disambiguation is usually impractical. Some authors propose the use of *underspecified semantic representations* instead.

- Say what is meant by "a generate-and-test approach to disambiguation." (2 marks)
- List **three** of the problems of this approach. (3 marks)
- Say what is meant by "underspecified semantic representations". (2 marks)
- **Briefly** discuss how relevant underspecified semantic representations are to Crouch & Pulman's architecture. (3 marks)

APPENDIX

The following grammar and lexicon are used in both questions (4) and (5).

Syntax rules	Semantic translation rules	Semantic type of LHS
S $\rightarrow$ NP VP	S′ = VP′(NP′)	$\mathbf{t}$
VP $\rightarrow$ Vi	VP′ = Vi′	$\langle \mathbf{e}, \mathbf{t} \rangle$
VP $\rightarrow$ Vt NP	VP′ = Vt′(NP′)	$\langle \mathbf{e}, \mathbf{t} \rangle$

(where α' represents the semantic translation of syntactic category α)

Word : Category	Logic translation	Semantic type
Punch : NP	p	$\mathbf{e}$
Judy : NP	j	$\mathbf{e}$
died : Vi	$\lambda x[died1(x)]$	$\langle \mathbf{e}, \mathbf{t} \rangle$
hit : Vt	$\lambda y[\lambda x[hit1(x, y)]]$	$\langle \mathbf{e}, \langle \mathbf{e}, \mathbf{t} \rangle \rangle$