UNIVERSITY OF EDINBURGH
COLLEGE OF SCIENCE AND ENGINEERING
SCHOOL OF INFORMATICS

Date: 23 August 2005 **Time: 14:30-16:30**

ARTIFICIAL INTELLIGENCE 2Ah

Examiners	J. Barnden	(External)
	H. Thompson	(Chair)

INSTRUCTIONS TO CANDIDATES

1. Candidates in the third or later years for the degree of M.A. (Arts), B. Comm, B.Sc (Social Sciences) or LL.B should put a cross (X) in the box on the front cover of the script book.
2. **Answer Parts A and B.** Parts A and B are worth 50% each. Answer ONE question from Part A and ONE question from Part B. If you attempt additional questions in either Part A or Part B, cross out one answer; if you do not, then the examiners will cross out the last question you answered in each Part.
3. Answer Parts A and B in separate script books – one script book for each Part.

Write as legibly as possible.

CALCULATORS ARE NOT PERMITTED.

THIS EXAMINATION WILL BE MARKED ANONYMOUSLY

Part A
ANSWER ONE QUESTION FROM PART A

1. Consider these statements:

   i. If I had flowers, ribbons and wire, then I could make a garland.

   ii. I could make a garland from flowers and soft branches.

   iii. Making a garland means I must have flowers.

   iv. We are always out of ribbons or soft branches.

   Is it possible to make a garland?

   (a) Represent statements **i** ... **iv** and the question in the propositional logic.

   [10%]

   (b) Is it sometimes possible to make a garland, and, if so, what type(s)? Represent this problem in CNF, and use the Davis-Putnam method to solve it.

   [10%]

   (c) What would it mean for this set of statements to be satisfiable?

   [4%]

   (d) Suppose that we represent a constraint satisfaction problem in Prolog as follows.

   - Let: $[x_{i,1}, x_{i,2}, \ldots x_{i,n}]$ be possible values for a variable v_i.
   - The Prolog term $poss(v_i, [x_{i,1}, x_{i,2}, \ldots x_{i,n}])$ links the values to the variable.
   - Problem list $C = [poss(v_1, [x_{1,1}, x_{1,2}, \ldots x_{1,n}]), poss(v_2, [x_{2,1}, x_{2,2}, \ldots x_{2,m}]),$ $\ldots, poss(v_k, [x_{k,1}, x_{k,2}, \ldots x_{k,p}])]$ lists all variables in a problem and each variable's allowed values.
   - Predicate `bad(Va,Xab,Vc,Xcd)` is true if assigning value `Xab` to variable `Va` and value `Xcd` to variable `Vc` at the same time violates a constraint.

   Your task is to write a Prolog predicate `searchbad(+C,[-Va,-Xab,-Vc,-Xcd])` that finds a pair of variables (`Va, Vc`) and values (`Xab, Xcd`) in the input problem list `C` that violate a constraint.

   [6%]

QUESTION 1 CONTINUES ON THE NEXT PAGE

(e) Answer these questions with 1-2 sentences each:

(i). What is entailment and how can it be tested?

(ii). What are the five components in a formal definition of a search problem?

(iii). Write $(A \wedge B) \supset C$ in conjunctive normal form.

(iv). When using a state evaluation function in a hill-climbing search algorithm, what bad result could occur? Explain.

(v). What problem does depth-first search have that iterative deepening solves?

(vi). Is $(p \supset q) \supset (q \supset p)$ valid? Show why or why not.

(vii). What does AC3 do to simplify a constraint satisfaction problem?

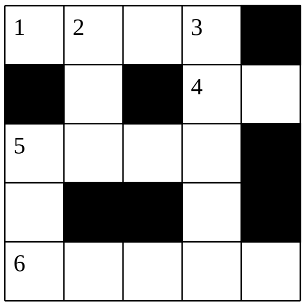
(viii). What is the effect of executing this Prolog constrained logic programming code: `X in 1..10, Y in 1..10, X #> Y+6.`

(ix). What causes combinatorial explosion in search problems?

(x). What does optimal mean when we say a search algorithm is optimal?

[20%]

2. You are to develop an automated crossword puzzle solver, based on a constraint-satisfaction process. The puzzle board is:



Assume that there are no definitions supplied, only a list `L` of possible words to fill the rows and columns. When a number appears in a box, it can be the start of either a horizontal word (if it has an edge or dark square to the left) or a vertical word (if it has an edge or dark square above) or both.

(a) How would you formulate the puzzle solver as a constraint satisfaction problem? Draw the constraint graph that corresponds to this puzzle board.

[5%]

(b) Assuming that the constraint satisfaction solver is not able to find a complete solution, how would you extend the process to find a complete solution?

[5%]

(c) Give three heuristics you could use in the post-CSP search process to improve efficiency. In each case, explain why the heuristic might work.

[5%]

(d) Suppose that we represent a clause (i.e., a disjunction of literals) as a Prolog list of atoms. Eg. $a \vee e \vee \neg c$ is represented by `[a,e,not(c)]`. We then represent a set of clauses by a list of lists, where the inside list is defined as above. Eg. $\{a \vee b, \neg c \vee d \vee b\}$ is represented by `[[a,b],[not(c),d,b]]`. A term `X` is a *pure literal term* if and only if it appears as `X` everywhere in the set, or as `not(X)` everywhere in the set. The term `b` is a pure literal in the example above. Write a Prolog predicate `pure(+InList,-Term,-Result)`. `InList` is the input set of clauses represented as a list. Among the output variables, `Term` is a pure literal term and `Result` is the simplified input set. The Prolog predicate `pure` simplifies the set by treating the `Term` as `true` and removing all clauses in the set that now evaluate to `true`. For example, if you have $\{a \vee b, b \vee c, c \vee a\}$ then `b` appears in only one form, and so is a pure literal. `pure` could select `b` as the pure literal returned in `Term` and yield the simplified `Result=[[c,a]]`.

[15%]

QUESTION 2 CONTINUES ON THE NEXT PAGE

(e) Answer these questions with 1-2 sentences each:

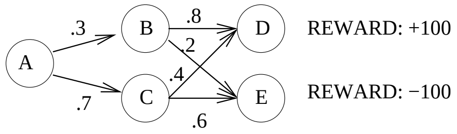
(i). What is entailment and how can it be tested?

(ii). Is depth-first search a complete search strategy? Is breadth-first search? Explain why.

(iii). Name one algorithm for efficiently finding a solution to a satisfaction problem and one algorithm for efficiently finding a solution to a validity problem. Make clear which algorithm answers each question.

(iv). How can a truth table be used in a satisfiability problem? Is this a good idea?

(v). What is the expected reward/utility at node A?



(vi). What is the search problem that arises in the stereo vision process? Explain why it can be a significant computational problem.

(vii). What is “fan-out” and why is this an issue for search algorithms?

(viii). Suppose you have a set of logical formulas that describe how an aeroplane controls the engines, concluding “safe flying”. Should we analyse the set of formulas as a satisfaction or a validity problem? Explain why.

(ix). Give two heuristics for reducing computational costs in a search problem.

(x). How can we tell if a problem is likely to be efficiently solved by Davis-Putnam?

[20%]

Part B
ANSWER ONE QUESTION FROM PART B

1. Consider the following two sentences in first-order logic:

   (A) $\forall$ X $\exists$ Y (Y $\preceq$ X)
   (B) $\exists$ Y $\forall$ X (Y $\preceq$ X)

   under the following assumptions:

   - X and Y range over all the words in English (e.g. aardvark, banana, bison, cat, chocolate, dog, fruit) ;
   - the predicate $\preceq$ means "precedes alphabetically or is the same as" and is therefore reflexive and transitive.

   (a) Give an unambiguous English translation of each sentence, **(A)** and **(B)**. [5%]

   (b) Is **(A)** true under this interpretation in which X and Y range over English words? Explain why or why not. [5%]

   (c) Is **(B)** true under this same interpretation? Explain why or why not. [5%]

   (d) Using resolution refutation, try to prove that **(A)** follows from **(B)**. Show the unifying substitution for each resolution step. If the proof fails, explain where, how and why it breaks down. [10%]

   (e) Using resolution refutation, try to prove that **(B)** follows from **(A)**. Show the unifying substitution for each resolution step. If the proof fails, explain where, how and why it breaks down. [5%]

QUESTION 1 CONTINUES ON THE NEXT PAGE

(f) Answer these questions with 1-2 sentences each:

(i). What are the elements of a model for *propositional logic*?

(ii). Automated reasoning operates on representations. Both *propositional logic* (PL) and *first-order logic* (FOL) can be used to represent a domain such as digital circuits. Why is FOL preferable to PL for automated reasoning about this domain (and about other domains as well)?

(iii). What kind of information can *quantifiers* be used to convey in *first-order logic*? Include at least one kind of information for *universal quantifiers* and at least one kind of information for *existential quantifiers*.

(iv). For what kind of knowledge base is Generalised Modus Ponens an *incomplete* reasoning procedure?

(v). In unifying two terms, what does the *most general unifier* (mgu) refer to?

(vi). What is the *Chomsky hierarchy*?

(vii). When are two grammars said to be *weakly equivalent*?

(viii). What does it mean for a string to be *locally ambiguous* with respect to some grammar?

(ix). Describe one difference between what gets *memo-ised* in the *CKY parsing algorithm* and in the *Earley algorithm*.

(x). What does it mean for a language to have a *compositional semantics*?

[20%]

2. We saw in class that Definite Clause Grammars (DCGs) use arguments to enforce grammatical constraints like *number agreement*. Thus, the following tiny DCG for English avoids sentences such as "The boy see the girl" and "The girls sees the boy", by requiring the **subject** and **verb** to have the same number (`Num`):

```
Grammar

s --> np(Num), vp(Num).

vp(Num) --> tv(Num), np(_).
vp(Num) --> iv(Num).

np(Num) --> pro(Num).
np(Num) --> det(Num), n(Num).
np(Num) --> npr(Num).


Lexicon

tv(Num) --> [Word], {tv(Word,Num)}.
tv(sees,sg).
tv(see,pl).

iv(Num) --> [Word], {iv(Word,Num)}.
iv(smiles,sg).
iv(smile,pl).
```

```
Lexicon

pro(Num) --> [Word], {pro(Word,Num)}.
pro(he,sg).
pro(they,pl).

det(Num) --> [Word], {det(Word,Num)}.
det(a, sg).
det(the, _).


n(Num) --> [Word], {n(Word,Num)}.
n(boy,sg).
n(boys,pl).
n(girl,sg).
n(girls,pl).

npr(Num) --> [Word], {npr(Word,Num)}.
npr(john,sg).
```

Although English has *number agreement*, it differs from many other human languages in not having *case constraints* except in the case of 1^{st} person pronouns (I, me; we, us) and 3^{rd} person pronoun (he, him; she, her; they, them). That is, in English, different pronouns are required for the **subject** and the **object** of a transitive verb (`tv`), while other noun phrases (`np`) are the same:

[we, they] see the boy	(Pronouns OK in **subject** position)
∗The boy sees [we, they]	(Pronouns not OK in **object** position)
The boy sees [us, them]	(Pronouns OK in **object** position)
∗[us, them] see the boy	(Pronouns not OK in **subject** position)

(Here, ∗ at the start of a sentence indicates that it is not grammatical.)

(a) Create *lexical entries* for the 3^{rd} person pronouns `them` and `him` that will allow a grammar to enforce *case constraints* in the case of a pronoun **subject** and **object**. You will also need to modify the entries already given for the pronouns `he` and `they` in the above `Lexicon`.

[8%]

(b) Modify the grammar rules for rewriting a noun phrase (`np`) to enforce *case constraints* when `np` is re-written as a pronoun, but not when it is re-written in other ways.

[6%]

(c) Now modify the other grammar rules needed to enforce *case constraints* in the realisation of pronouns in subject position (e.g., “he likes the girls, “they smile”) and their realisation in object position (e.g., “the girls like him”).

The grammar must **not** require case agreement when the noun phrase is not a pronoun. That is, the grammar should still allow both “the girl sees the boy” and “the boy sees the girl”.

[6%]

(d) Using your modified DCG, show a proof of

```
s([the, boy, sees, them]).
```

[8%]

(e) The 2^{nd} person pronoun `you` doesn’t vary: it is the same in both singular and plural number and in both subject and object position. Create a lexical entry for `you` that is compatible with your grammar, but reflects the fact that it doesn’t vary.

[2%]

QUESTION 2 CONTINUES ON THE NEXT PAGE

(f) Answer these questions with 1-2 sentences each:

(i). What are the elements of a *model* for *first-order logic*?

(ii). Automated reasoning operates on representations. Both *propositional logic* (PL) and *first-order logic* (FOL) can be used to represent a domain such as digital circuits. Why is FOL preferable to PL for automated reasoning about this domain (and about other domains as well)?

(iii). What are **two** ways of using *Generalised Modus Ponens* in reasoning?

(iv). For what kind of knowledge base is Generalised Modus Ponens an *incomplete* reasoning procedure?

(v). Specify two conditions under which two terms would fail to unify.

(vi). When is a grammar *strongly adequate* for a language?

(vii). What are *variables* used for in a Definite Clause Grammar?

(viii). How does the CKY (chart) parser deal with *non-determinism*?

(ix). Explain what the "dot" indicates in the *dotted rules* used in the Earley Algorithm.

(x). How is the interpretation of an *idiom* like "hit the road" (as in "After drinking some coffee, he hit the road.") or like "without a shadow of a doubt" (as in "I'm convinced without a shadow of a doubt") **not** compositional?

[20%]

Specimen Answers

Part A

1. (a) Let flowers:b, ribbons:t, garland:s, wire:c, soft branches:h. Then the clauses are:
$b \wedge t \wedge c \supset s$
$b \wedge h \supset s$
$s \supset b$
$\neg t \vee \neg h$
The desired conclusion is s.

Marking guide: 2 pts for each term

(b) (1) convert to CNF: $\{s \vee \neg b \vee \neg t \vee \neg c, s \vee \neg b \vee \neg h, b \vee \neg s, \neg t \vee \neg h, s\}$.
(2) Use Davis-Putnam: $\neg t$ is unit, so set $t = false$. This reduces set to: $\{s \vee \neg b \vee \neg h, b \vee \neg s, s\}$.
(3) singleton $s = true$, reduces set to: $\{b\}$.
(4) Set $b = true$
This implies flowers(b), garland(s), soft branches(h - not constrained by this solution).

Marking guide: 4 pts for CNF, 4 pts for DP, 2 pts for extracted solution. If student uses depth-first search or truth-table, then maximum of 7 points.

(c) satisfiable if an assignment to variables exists satisfying the premises and making the conclusion true.

Marking guide: pass if recognise satisf. means something, 4 if can distinguish from validity

(d) % sloppy layout due to latex problems
searchbad(C,[Va,Xab,Vc,Xcd]) :-
member(poss(Va,Lista),C), % find a first variable
member(Xab,Lista), % find a first value
member(poss(Vc,Listc),C), % find a second variable
member(Xcd,Listc), % find a second value
bad(Va,Xab,Vc,Xcd). % test for constraint violation

Marking guide: need comments, good variable and clause names, layout: 2 pts
need approach to generate variables: 1 pt
need approach to generate values: 1 pt
elegant use of member to generate via backtracking: 2 pts

(e) i) A entails B if in every world where A is true, B is also true. This can be tested by constructing a truth table and either (1) calculating whether $A \supset B$ always comes out true or (2) checking whether in all worlds in which A is true, B is also true.
ii) initial state, general state, operators, end test, path cost function.
iii) $(A \wedge B) \supset C$ goes to $C \vee \neg(A \wedge B)$ goes to $C \vee \neg B \vee \neg A$.
iv) Local optima mean no single step improves the result, but something better is possible.
v) Depth first can go on to an infinite depth search and can also miss a closer solution that iterative deepening search would have found.
vi) No, because $p = T$ and $q = F$ invalidates the statement.

vii) By removing impossible values from the domain of a variable, it reduces the search problem (possibly to nothing).
viii) X in 7..10, Y in 1..4.
ix) Fan-out, because each of N possibilities at one level leads to N*N at the next, and N*N*N at the next etc.
x) It is guaranteed to find the best solution if there are several.

Marking guide: 2 pts each, weighted by normal marking scale

2. (a) The Variables can be the unfilled rows or columns, the Domains be can the list of allowable words and the Constraints can be the position X of the first word has to be the same as position Y of the second word. The constraint graph is (latex isn't allowing figures in the answers): 1-2, 1-3, 2-5H, 3-5H, 5H-5V, 5V-6, 3-6, 3-4.

   Marking guide: 1 pt each for Variables, Domains, Constraints and graph. +1 for all correct.

   (b) You will need to implement some sort of search process that will take the elements from the remaining domains of each variable, and see if these solve the problem. This is essentially a combinatorial problem but hopefully smaller than the original problem.

   Marking guide: 2 pts for recognising search is needed. 2 pts for a sensible search strategy. 1 pt for a perfect solution.

   (c) Anything sensible can be used, but some could be: 1) use most constrained slot first, 2) use most constraining slot first, 3) provide multiple lists with words indexed by letter in intersecting positions.

   Marking guide: 1 pt each for a useful heuristic. 2 pts for clear and valid justifications.

   (d) badly spaced because verbatim doesn't work properly

```
pure(Set,Term,Result) :-
member(Clause,Set),
member(Term,Clause), % backtrack to finding clauses
unique(Term,Set), % check for only in one form
simplify(Term,Set,Result). % simplify the set with this term

% see if in set in only 1 form
unique(Term,Set) :-
invert(Term,Inv), % find simplified form of not(Term)
member(Inv,Set),!,fail.

% find simplified form of not(Term)
invert(not(X),X) :- !.
invert(X,X).

% simplify by removing terms containing Term
simplify(_,[],[]).
simplify(Term,[Head—Tail],Result) :-
```

member(Term,Head),!, simplify(Term,Tail,Result). % remove Head because Term is found
simplify(Term,[Head—Tail],[Head—Result]) :-
simplify(Term,Tail,Result). % recurse when Term not in Head.

Marking guide: need comments, good variable and clause names, layout: 4 pts
need approach to simplify clauses: 4 pts
need approach that recognises uniqueness 4 pts
need approach that sorts not(not(a)): 3 pts

(e) i) A entails B if in every world where A is true, B is also true. This can be tested by constructing a truth table and either (1) calculating whether $A \supset B$ always comes out true or (2) checking whether in all worlds in which A is true, B is also true.
ii) No (infinite paths). Yes (if a solution at level N, eventually gets there).
iii) Davis-Putnam, Resolution.
iv) Keeps track of variables until a true result is found. But it can be less efficient than Davis-Putnam.
v) rewards: E:-100, D:+100, C:-20, B:+60, A:4
vi) correspondence - finding matching features/pixels. There may be many possibilities.
vii) Fan-out is the number of descendents of a state and a large fan-out means very time-consuming search.
viii) Validity because you want "safe flying" in all situations.
ix) Use most constrained variable first. Split into independent subproblems.
x) Look at the ratio of clauses to variables. If ratio distant from phase transition, then likely to work.

Marking guide: 2 pts each, weighted by normal marking scale

Part B

1. (a) **A**. Every English word has some English word that precedes it alphabetically or is the same as.

   B. There is some English word that every English word precedes alphabetically or is the same as.

   Marking guide: Count them equally. Quantifiers must be in the right order. Of course, even the given answers are really ambiguous, but their most common reading is the correct one.

   (b) It's true because every English word is the same as itself. Also because the first word in the dictionary precedes or is equal to every English word.

   Marking guide: Some combination of the above answers should also be marked correct, provided it covers the entire space.

   (c) It's true, but the only possible reason is that every English word follows or equal to the first word in the dictionary.

   Marking guide: Only one possible answer

   (d) 1. Negate (A) to get $\exists X \; \forall \; Y \; \neg(Y \preceq X)$.
   2. Skolemize to remove quantifiers: i.e., replace X with a Skolem constant and drop the universal quantifier to get $\neg(Y \preceq c1)$.
   3. Skolemize (B) to remove quantifiers: i.e., replace Y with a different Skolem constant and drop the universal quantifier to get $(c2 \preceq X)$.
   4. Unify the resulting clauses under the substitution: $\{X/c1,\ Y/c2\}$ to get the empty clause.
   success

   Marking guide: Make sure the student negates what s/he is trying to prove. Make sure it's a resolution proof, not a GMP proof. Make sure the substitutions are included. None of the assumptions is needed in the proof.

   (e) 1. Negate (B) to get $\forall Y \; \exists \; X \; \neg(Y \preceq X)$.
   2. Skolemize to remove quantifiers: i.e., replace X with a Skolem function of Y and drop the universal quantifier to get $\neg(Y \preceq f1(Y))$.
   3. Skolemize (A) to remove quantifiers: i.e., replace Y with a different Skolem function on X and drop the universal quantifier to get $(f2(X) \preceq X)$.
   4. Try to unify the resulting clauses. The needed substitution $\{X/f1(Y), Y/f2(X)\}$ is illegal.
   failure

   Marking guide: The student should realise before s/he starts that the proof will fail. But his/her explanation should be in terms of the proof, not simply in terms of what (A) and (B) mean. The student's grade on this part should only be >2 if s/he realises that the proof fails. The previous part will have already shown if s/he can set up a resolution refutation proof.

   (f) The answers can be found in module2PartB-ans.txt. I can't be wasting time with this stupid format.

   Marking guide: 1 point for a partial answer, 2 points for a good one

2. (a)
```
pro(Num,Case) --> [Word], {pro(Word,Num,Case)}.
pro(they,pl,subj).
```

```
pro(them,pl,obj).
pro(he,sg,subj).
pro(him,sg,obj).
```

Marking guide: No credit if the student doesn't include an argument for case. Give 2 points if there is an argument for case, but its value doesn't vary. Don't give full credit if the argument already given for number is omitted. But any labels can be used for the two cases (subject and object).

(b)
```
np(Num,Case) --> pro(Num,Case).
np(Num,Case) --> det(Num), n(Num).
np(Num,Case) --> npr(Num).
```

Marking guide: Give 3 points if case is correctly propagated to `pro`*, and another 3 if case is correctly omitted from the other two rules.*

(c) Only two rules need to be modified:
```
s --> np(Num,subj), vp(Num).
vp(Num) --> tv(Num), np(_,obj).
```
Marking guide: Give full credit only if these and only these rules are modified.

(d) See figure in dcgproof.eps

Marking guide: The student should remember to add the indices for start and end

(e)
```
pro(you,_,_).
```
Marking guide: Full credit only if two arguments and both are "don't care".

(f) Answer to be found in module2PartB-ans.txt

Marking guide: 1 point for a partial answer, 2 points for a good one