Module Title: Informatics 2A
Exam Diet (Dec/April/Aug): August 2011
Brief notes on answers:

PART A

1. d
2. d
3. d
4. c
5. b
6. a
7. c
8. c
9. c
10. b
11. b - because the grammar is left recursive.
12. d - because the conversion algorithm may require the powerset of the original stateset.
13. b - many different derivations can lead to the same parse tree because derivations are different if they choose to expand non-terminal instances in different orders.
14. b - because this pairs letters occurring in the same position in the two copies of the string in the sentence.
15. d - the grammar is certainly unambiguous and locally unambiguous and the first two rules ensure we need arbitrary long lookahead to distinguish between uses of these two productions.
16. d - b and d both start productions for A, a starts a production for B (which starts a production for A) and A can derive ε
17. c- $L(S_1)$ has the same number of as and bs, $L(S_2)$ has the same number of as and cs so the intersection ensures equal numbers of as bs and cs.
18. d - the language recognised by the original machine is a^+.
19. b - state 1 copies the initial portion of the string onto the stack and state 2 checks the remaining symbols match the initial portion.
20. c- this is the definition.

PART B

1. (a) Anything reasonable will be accepted, for example:

$$\begin{array}{rcl}
\text{Cost(chicken)} & = & £3.00 \\
\text{Cost(mushroom)} & = & £1.00 \\
\text{Cost(pumpkin)} & = & £2.00 \\
\text{Cost(orange)} & = & £1.00 \\
\text{Cost}(\mathsf{Adj\ NP}) & = & \text{Factor}(\mathsf{Adj}) * \text{Cost}(\mathsf{NP}) \\
\text{Cost}(\mathsf{NP1}\ \text{and}\ \mathsf{NP2}) & = & \text{Cost}\ (\mathsf{NP1}) + \text{Cost}\ (\mathsf{NP2}) \\
\text{Cost}(\mathsf{NP1}\ \text{in}\ \mathsf{NP2}\ \text{sauce}) & = & \text{Cost}\ (\mathsf{NP1}) + (0.5 * \text{Cost}\ (\mathsf{NP2})) + £0.10 \\
\text{Factor(fresh)} & = & 1.5 \\
\text{Factor(roasted)} & = & 1.2 \\
\text{Factor(organic)} & = & 1.3
\end{array}$$

Of course, whether the numbers are realistic isn't the issue, but the definition should be compositional and should have something like the 'additivity' properties illustrated by the above. [3 marks for a definition of a cost function; 2 marks if it's compositional; 1 mark each for the stated characteristics.]

(b) There are 5 parse trees [2 marks]. The two indicated the following bracketings will have different costs:

(fresh (chicken and mushroom)) in a pumpkin sauce
((fresh chicken) and mushroom) in a pumpkin sauce

For the cost function given above, these come out at £7.10 and £6.60 respectively.

(c) It's not possible [1 mark], because all the parse trees involve exactly the same rule applications, albeit in different 'orders', and the probability of a parse tree is just the product of the probabilities for the rule applications [2 marks].

(d) The following equivalent grammar will do:

$$\begin{array}{rcl}
\mathsf{NP} & \rightarrow & \mathsf{Adj}\ \ \mathsf{NP}\ \mid\ \mathsf{N}\ \ \mathsf{Rest} \\
\mathsf{Rest} & \rightarrow & \epsilon\ \mid\ \text{and}\ \mathsf{NP}\ \mid\ \text{in}\ \mathsf{NP}\ \ \text{sauce}
\end{array}$$

This has the following LL(1) parse table:

	Adj	N	and	in	$
NP	Adj NP	N Rest			
Rest			and NP	in NP sauce	ϵ

4 marks for the grammar, 3 marks for the parse table.

(e) Any reasonable answer will be accepted. E.g. the user cannot be expected to know the disambiguation convention and might think they're ordering something different.

2. (a) Award three marks for constructing the system of equations and 3 marks for solving them. The equations are:

$$\begin{aligned} R_1 &= aR_2 + \varepsilon \\ R_2 &= bR_3 + \varepsilon R_1 \\ R_3 &= cR_4 \\ R_4 &= \varepsilon R_1 \end{aligned}$$

Of the three points for solving the equations, award one point for finding that $R_1 = abcR_1 + aR_1 + \varepsilon$ and two points for solving the equation $R_1 = (abc+a)R_1 + \varepsilon$ using the fact that $R = A^*B$ is the smallest solution to the equation $R = AR + B$. This gives us: $R_1 = (abc + a)^*$.

(b)

state	**a**	**b**	**c**
$\{q_1\}$	$\{q_2, q_1\}$		
$\{q_2, q_1\}$	$\{q_2, q_1\}$	$\{q_3\}$	
$\{q_3\}$			$\{q_4, q_1\}$
$\{q_4, q_1\}$	$\{q_2, q_1\}$		

Award 2 marks for an indication that the student knows the construction, three marks for the table, interpolated for inaccuracies, one mark for getting final states right.

(c) (i). The technician is wrong - one mark.

(ii). The language is not regular, this can be justified by looking at the dependencies and/or by outlining a pumping lemma proof. For example, if we choose $s = a^k(abc)^k\$a^{2k}$, where k is parameter in the pumping lemma then this is a suitable choice of string. I'd award up to two marks for the observation that it is not regular and that the dependencies show this. An additional mark for some discussion of the use of the pumping lemma.

(iii). The easiest way to construct this is to use the original FSM as a starting point, extending the transitions to manipulate the stack so a symbol is pushed every time there is an a transition, then adding a transition labelled \$ from the final state of the original FSM to a machine that checks the number of as read is equal to the number of symbols pushed onto the stack in the initial phase. Award 1 mark for an awareness of PDA diagrams, one mark for pushing the symbols and one mark for checking the number of symbols on the stack.

(d) We would need to use a context-free grammar (one mark). The grammar generates matched sequences a and a or sequence abc with a and does this iteratively (two marks). This is an example (award up to three marks):

$$S \rightarrow \$ \mid abcSa \mid aSa$$

3. (a) One mark for the definitions of ambiguity. Two marks for a short discussion of diferent parsing approaches, their applicability to ambiguous grammars and the time efficiency of the different approaches.

   (b) The two parses differ on how often the S → A S is used or not. The use of this production gives rise to the less likely parse. Three marks for the parses and one marks for the explanation.

   (c) The probabilities for the parts of the parse that differ are in the production of the two central bs, in one case this has a probability of 0.64, in the other the probability is 0.01.

   (d) The approach here would be to reverse the probabilities on the first two productions. Allocate 4 marks for the redesign and two marks for the recalculation.

   (e) The language generated is $(b + c)(a + b + c)^*(a + b)$.

   (f) Any grammar that generates the language. For example:

$$S \to CAB, \quad C \to b \mid c, \quad B \to a \mid b, \quad A \to aA \mid bA \mid cA \mid \varepsilon$$

   Allocate 4 marks for a fully correct answer, deduct 1 mark for each distinct defect in the answer.