Logic

Chris Brown

Do NOT try to submit these assignments through Blackboard: you can't. Use the turn-in box outside Marty's office on 7th floor CSB.

If there is a lot of prose in your answers (as opposed to math), consider using a text-processor! Write mathematics correctly – if in doubt see the courses "Helper Tools" page in the "Resources" section of the course homepage.

Work is due at the end of the module week named in the title.

1 Logic

(RNX.Y) means that this is problem X.Y from Russell and Norvig.

1.1 Week 2

1.2.1 (RN7.5): Consider a vocabulary of four propositions, $A, B, C,$ and D. How many models (different possible truth assignments) are there for the these four propositions? How many models are there for the following three sentences (how many truth-assignments to the propositions make each one true?) Hint: does D's value affect the truth values of A, B, or C?
A) $A \vee B$.
B) $(A \Leftrightarrow B) \Leftrightarrow C$.
C) $(A \wedge B) \vee (B \wedge C)$

1.2.2. Prove or find a counterexample to assertions A and B (hint: use the definition of $\models$.)
A) if $\alpha \models \gamma$ or $\beta \models \gamma$ (or both) then $(\alpha \wedge \beta) \models \gamma$.
B) if $\alpha \models (\beta \wedge \gamma)$ then $\alpha \models \beta$ and $\alpha \models \gamma$.
Is it true that
$A \Leftrightarrow B \models \neg A \vee B$?

1.2.3 (RN7.6): We've defined four binary logical connectives: How many binary connectives can there be? Are there any others that might be useful? Why are some of them not too useful?

1.2.4: (RN7.7) Use truth tables to verify:
A) $(\alpha \Rightarrow \beta) \Leftrightarrow (\neg\alpha \vee \beta)$.
B) $\neg(\alpha \wedge \beta) \Leftrightarrow (\neg\alpha \vee \neg\beta)$.

1.2.5: (RN7.8) Using truth tables, establish whether these sentences are valid, unsatisfiable, or neither.
A) $(Smoke \Rightarrow Fire) \Rightarrow (\neg Smoke \Rightarrow \neg Fire)$
B) $Smoke \vee Fire \vee \neg Fire$
C) $Big \vee Dumb \vee (Big \Rightarrow Dumb)$

1.2.6: *The maid said that she saw the butler in the living room. The living room adjoins the kitchen. The shot was fired in the kitchen, and could be heard in all the nearby rooms. The butler, who had good hearing, said he did not hear the shot.* Prove (using a truth table if you want but resolution preferred): *if the maid told the truth, the butler lied.*

1.2 Week 3

1.3.1 (RN 8.6): Represent the following sentences in FOPC:
Some students took French in Spring 2006.
The best score in Greek is always higher than the best score in French.
Only one student took Greek in Spring 2006.
There is a barber who shaves all men in town who do not shave themselves.
There is an agent who sells policies to people who are not insured.
You can fool some people all of the time, and all the people some of the time, but not all the people all of the time.

1.3.2 (RN 8.7): Represent the sentence "All Germans speak the same languages." in FOPC. Use *Speaks(x,l)* to mean that person *x* speaks language *l*.

1.3.3 (RN 9.5): Give the most general unifier, if it exists:
A. $P(A, B, B), P(x, y, z)$.
B. $Q(y, G(A, B)), Q(G(x, x), y)$.
C. $Older(Father(y), y), Older(Father(x), John)$
D. $Knows(Father(y), y), Knows(x, x)$.

1.3.4. (Len Schubert CSC 244/444): Prove by resolution.

The domain is people. Use L(x,y) for "x likes y", A(x,y) for "x is afraid of y", F(x,y) for "x has friend y", P(x) for "x is paranoid".

No-one likes anyone of whom he is afraid. No-one likes anyone who does not like him. All paranoids are afraid of everyone except their friends. There is a paranoid, and all of his friends are also paranoid. To Prove: *There is someone whom all non-paranoids dislike.*

Hints: As always, the axiomatization must be right for the proof to work! The last premise and the denial each generates two clauses. If you construct an intuitive argument, one that seems convincing and uses all the premises and the conclusion, then it can guide you as to what clauses to resolve in what order (this is a capability computers do NOT have!). You can derive the null-clause in five resolutions.

Extra Credit:

1.3.5:
Restate enough of your Prolog family-tree definitions in FOPC to demonstrate you know

what's going on and then write the definition of "m-th cousin n-times removed" in FOPC, for a few small ms and ns.

How could you generalize the "Parent" and "Child" predicates to allow the *m-th cousin n'th removed* predicate to be defined for arbitrary *m,n*?

1.3 Week 4

For all the circuit problems, "gate" means two-input AND, OR, or one-input NOT. No XOR, NAND, no n-input ANDs, etc. Such solutions are not wrong or bad, but they really expand the space of possibly correct answers and run the risk of being hard to analyze systematically.

1.4.1. Design a circuit for the *majority* function of inputs *w,x,y,z* that is 1 if and only if most (i.e. three or more) of its inputs are 1.

1.4.2. Draw circuit diagrams for two different 2-bit adders (adders that take two 2-bit inputs and compute a 2-bit output and a carry bit): 1) a ripple-carry adder and 2) a divide-and-conquer adder. For the ripple carry adder, you may use the straight sum-of-products adder (derived from disjunctive-normal-form ("sum of minterms"), without trying to eliminate any literals or terms). OR, you can reduce the number of gates you have to draw by simplifying the 1-bit adder circuit with a Karnaugh map. For each of the 2-bit adders, give the total number of gates and the depth (the number of gates on the longest path).

1.4.3. Minimize the following expression using a Karnaugh map (i.e., remove as many literals and terms as possible): $(A \wedge B \wedge C \wedge D) \vee (\neg A \wedge B \wedge C \wedge D) \vee (A \wedge \neg B \wedge C \wedge D) \vee (A \wedge B \wedge \neg C \wedge D) \vee (\neg A \wedge B \wedge \neg C \wedge D)$

Express the result in sum of products notation. (Hint: each prime implicant is an OR (sum) of minterms (ANDs, or products). You can thus directly generate the sum of products notation from the Karnaugh map). Also, minimize the expression using algebraic laws with the result in product of sums notation. Show that the two are equivalent (derive one algebraically from the other).

1.4.4. (Extra Credit) For those of you who remember your recurrences...
Determine, for the ripple-carry and divide-and-conquer adders mentioned in 1.4.1, a general formula for the total number of gates and the depth, as a function of the number of bits N in each operand. (This is not a big-O question: I'm looking for an exact answer.)