CPSC 121 Sample Solutions to Midterm Exam #2
Wednesday, March 12, 2014, 17:30–19:30

[15] 1. **iSaks:** At the iSaks auction site, vendors offer up "lots" of identical items for sale. Each lot must have at least 1000 items. A bid is a dollar value **per item** d and a quantity q of items: (d, q).

The following definitions describe this domain, but unless we state them explicitly, the parts described as "intentions" are not known to hold.

- B is the set of bidders
- $\text{Lot}(i, n)$ means lot number i offers n (a positive integer) items for auction. i is a positive integer ID number. (The intention of i is that it uniquely identifies a particular lot and tells us what order lots were offered in; that is, no two lots should have the same ID number. For now, iSaks sells only lots of salt shakers; so, there's no need for a description of the item.)
- $\text{Bid}(j, b, d, q, i)$ means bidder b bid d (positive integer) dollars per item for q (positive integer) items on the lot with ID i. j is a positive integer ID number for the bid. (A lot and a bid may coincidentally have the same ID number, but the intention is that no two bids should have the same ID number.)
- $\text{Won}(b, d, q, i)$ means that bidder b won q items from lot i at the price d.

[6] a. **Defining Predicates:**

i. It would be handy to be able to refer to a bidder's "final bid". Define the predicate $\text{FBid}(b, d, q, i)$ to mean that the highest (by ID number) bid by bidder b on lot i is for d dollars per item and q items.
$\text{FBid}(b, d, q, i) \equiv$

Solution : We want to say "this is a bid (i.e., has a bid ID) and there's no other bid by the same bidder on the same lot with a higher bid ID".
$\exists j \in \mathbf{Z}^+, \text{Bid}(j, b, d, q, i) \wedge {\sim}(\exists j_2 \in \mathbf{Z}^+, \exists d_2 \in \mathbf{Z}^+, \exists q_2 \in \mathbf{Z}^+, \text{Bid}(j_2, b, d_2, q_2, i) \wedge j_2 > j)$
Note that because we know that bid IDs are unique, we might instead say "for every j where Bid(j, b, d, q, i) is true (domain restriction on a universal)..." as the start rather than "there is a j where Bid(j, b, d, q, i) is true".
We'll accept either, but not that if j is universally quantified, then the Bid part connects to the next subexpression with a $\rightarrow$ not a $\wedge$.

ii. Define a predicate $\text{Winner}(b, i)$ to mean that bidder b won lot i (for some dollar value and quantity).
$\text{Winner}(b, i) \equiv$

Solution : This one is pretty much exactly as the statement goes:
$\exists d \in \mathbf{Z}^+, \exists q \in \mathbf{Z}^+, \text{Won}(b, d, q, i)$

[7] b. **Making Statements:**

State the following facts that we want to assume true in predicate logic. **YOU MAY USE THE EXTRA PREDICATES YOU WERE ASKED TO DEFINE ABOVE.** (Assume they are correct.)

i. Each lot must have at least 1000 items.

Solution: So, all lot IDs and quantities *that represent actual lots* (domain restriction) have at least 1000 items in them.
$\forall i \in \mathbf{Z}^+, \forall n \in \mathbf{Z}^+, \text{Lot}(i, n) \rightarrow n \geq 1000$
(I went with a universal on this, but it might actually be easier to think about saying "there's no lot with fewer than 1000 items". The result is logically equivalent!)

ii. The final bid of a winning bidder on a lot is never for less money than the final bid of a losing bidder on the same lot.

Solution: This is by far the hardest predicate logic statement to write on the exam.
Fortunately, we have predicates for "final bid" and "winner" which help.
We want to establish the set of all final bids of winners with a domain restriction like:
$\forall b \in B, \forall d \in \mathbf{Z}^+, \forall q \in \mathbf{Z}^+, \forall i \in \mathbf{Z}^+, (\text{FBid}(b, d, q, i) \wedge \text{Winner}(b, i)) \rightarrow \ldots$
Now, we'd like to say that there's nobody else who's a winner on that lot and whose final bid is larger than the winner's final bid. (We didn't worry about saying the bidders are unequal explicitly, relying instead on $d < d_2$ and the fact that there should only be one final bid by a bidder.)
$\forall b \in B, \forall d \in \mathbf{Z}^+, \forall q \in \mathbf{Z}^+, \forall i \in \mathbf{Z}^+, (\text{FBid}(b, d, q, i) \wedge \text{Winner}(b, i)) \rightarrow {\sim}\exists b_2 \in B, \exists d_2 \in \mathbf{Z}^+, \exists q_2 \in \mathbf{Z}^+, \text{FBid}(b_2, d_2, q_2, i) \wedge {\sim}\text{Winner}(b_2, i) \wedge d < d_2$

[2] c. **Negation practice:**
Negate the following statement, moving the negations "inward" as far as they will go:
$\forall i \in \mathbf{Z}^+, \forall b_1 \in B, \text{Winner}(b_1, i) \rightarrow {\sim}\exists b_2 \in B, \text{Winner}(b_2, i) \wedge b_1 \neq b_2.$

Solution: We just put a negation on the outside and then move it in with rules like universal De Morgan's. When we move a negation across a universal, it becomes an existential. When we move a negation into a conditional, it works like ${\sim}(p \rightarrow q) \equiv {\sim}({\sim}p \vee q) \equiv p \wedge {\sim}q$. Finally, when we get something like ${\sim}{\sim}p$, it's just p, which in this case means there's not much work to do!
$\exists i \in \mathbf{Z}^+, \exists b_1 \in B, \text{Winner}(b_1, i) \wedge \exists b_2 \in B, \text{Winner}(b_2, i) \wedge b_1 \neq b_2.$
It's worth thinking about what this translates to, by the way!

[8] 2. **Who's on First (or the Equivalent)?**

For each of the following pairs of predicate logic statements, **CIRCLE ONE OF**: **EQUIV** (meaning they're logically equivalent), $1 \rightarrow 2$ (meaning the second one follows from the first but the first does not follow from the second), $2 \rightarrow 1$ (meaning the first one follows from the second but the second one does not follow from the first), or **NEITHER** meaning neither one implies the other.

(a) $\exists x \in X, R(x).$

$\forall x \in X, \sim R(x)$.

Solution : The second line is logically equivalent to $\sim\exists x \in X, R(x)$. So, these are negations of each other, and neither implies the other: **NEITHER**.

(b) $\exists x \in X, \exists y \in X, P(x,y) \rightarrow Q(x,y)$.
$\exists y \in X, \exists x \in X, \sim Q(x,y) \rightarrow \sim P(x,y)$.

Solution : If we apply the contrapositive rule to the second one, we get: $\exists y \in X, \exists x \in X, P(x,y) \rightarrow Q(x,y)$. At that point the only difference between these is the order of the two existentials, but we're free to rearrange the order of existentials; so, they're equivalent. (Don't remember why we can rearrange them? Think about the "challenge method": they're both the "same person's turn", anyway. More formally, this is related to commutativity and associativity.)
EQUIV

(c) $\forall x \in X, P(x) \wedge \sim\exists y \in Y, Q(x,y)$.
$\forall x \in X, \forall y \in Y, \sim Q(x,y)$.

Solution : Take a look at the first one and move the negation inward. You end up with something that's just like the second one, except it *also* states $P(x)$. Any time that first statement is true, the second is true (and also $P(x)$ is true for all x in X). Thus, **1** $\rightarrow$ **2**.

(d) $\sim\exists x \in X, \sim\exists y \in Y, \sim\exists z \in Z, P(x,y) \wedge P(y,z)$.
$\forall x \in X, \exists y \in Y, \forall z \in Z, P(y,z) \rightarrow \sim P(x,y)$.

Solution : Move the outermost negation inward on the first one over the first quantifier and cancel the next negation. Then, move the innermost negation inward as far as it goes. Next, convert the implication in the second one into an OR, use the commutative law, and they look the same. These are **EQUIV**.

[11] 3. **Sketchy Proofs:**

[6] a. **Pred Logic Statement to Proof:**
Sketch a proof in as much detail as possible for the theorem:
Theorem: $\forall x \in X, (\sim P(x,x) \vee (\forall y \in Y, P(x,y))) \rightarrow \exists z \in Z, Q(x,z) \vee Q(z,x)$.
(Do not use logical equivalences to alter the form of the theorem, which also means no proof by contradiction or contrapositive.)
NOTE: if your proof includes choosing a witness value, let us know which other variables' values its value can be based on.

Solution : Mostly, we can just read the solution off of the proof strategies sheet. For example, for a $\forall$, we'd use "without loss of generality".
We get:

WLOG, let x be an arbitrary element of X.

Assume $\sim P(x, x) \vee (\forall y \in Y, P(x, y))$. (Because of the $\vee$ in this, I wonder if a proof by cases might be appropriate, but it's not obvious that it is; so, we're OK with you having it or not.)

Choose a particular element of Z as the value of z. We cannot tell which to choose at this point, but we do know that our choice can be based on x if need be (because x has already been "chosen" earlier in our proof, which is to say its quantifier is outside z's quantifier).

Prove $Q(x, z)$ or prove $Q(z, x)$. (Possibly, you'd prove the $\vee$ of these two somehow.)

[5] b. **Proof to English Theorem:**

Consider the following fragment of a proof which is missing some details. (It was damaged during an ill-fated forest hike by algorithmicist R. R. Hood.) Write the theorem this was proving in predicate logic in as much detail as possible. Clearly define any sets or predicates you need.

Proof: Without loss of generality, let n be a non-negative integer (specifically a number of keys). We design the following "binomial queue" f of our choice: ... **SECTION OF PROOF LOST IN WOLF ATTACK** ...

Note that this binomial queue has exactly n nodes.

Without loss of generality, let g be a *different* binomial queue. We now show that g has some number of keys m that is not equal to n.

Solution: Again, we can largely read off our theorem in reverse using the proof strategies sheet.

We'll need a set of "binomial queues" B.

We'll need a predicate to tell us that a particular "binomial queue" has a particular number n of nodes: $\text{Has}(b, n)$. We didn't intend to use both the word "key" and "node", but if you made predicates for both like "HasKeys" and "HasNodes", that's totally fine.

(I figured this out as I went in solving the problem, just leaving blanks for sets or relation names I needed to come back to and filling them in after the fact.)

Now we can proceed. Note that designing an f **of our choice** suggests an existential.

$\forall n \in \mathbf{N}, \exists f \in B, \text{Has}(f, n) \wedge \forall g \in B, g \neq f \rightarrow \exists m \in \mathbf{N}, m \neq n \wedge \text{Has}(g, m)$.

(How did I know to use a $\rightarrow$ after $g \neq f$ but a $\wedge$ after $m \neq q$? Because it's domain restriction; on a universal, we use $\rightarrow$; on an existential, we use $\wedge$.)

[12] 4. **I Want the PROOF:**

Prove the following theorems using informal, English-language proofs. **Substantial partial credit** is available for well-formed proofs. Unclear, awkward, or unusual proof structures that are not precisely correct are likely to lose substantial credit.

[7] a. Prove the following theorem:

$\forall x \in X, (P(x, x) \vee (\forall y \in X, Q(x, y))) \rightarrow \exists z \in X, P(z, z) \vee Q(x, z)$.

Solution : Again, we can read off the form of our solution mostly from the proof strategies sheet. The big insights are that we need to proceed by cases on the $\vee$ on the left of the $\rightarrow$ and that we can then match the $P(x,x)$ part to the $P(z,z)$ part and the $Q(x,y)$ part to the $Q(x,z)$ part.

Here we go:

WLOG, let x be an arbitrary element of X.

Assume (for antecedent assumption) $P(x,x) \vee (\forall y \in X, Q(x,y))$. We proceed by cases (the two disjuncts of our assumption):

i. Assume $P(x,x)$. Now, choose $z = x$. Then, $P(z,z)$ is true. Therefore, by generalization, $P(z,z) \vee Q(x,z)$ is true.
ii. Assume $\forall y \in X, Q(x,y)$ is true. Then, we know whatever we choose z to equal, y could also be that and so $Q(x,z)$ will be true (but note the subtle point below). So, we choose z to be any element of X, and $Q(x,z)$ is true. Therefore, by generalization, $P(z,z) \vee Q(x,z)$ is true.

QED!

Subtle note: In fact, the proof falls apart if the set X is empty (and so there's *no* value we can choose z to equal in the second case). If you noticed that and stated clearly: full credit to you! If not, no problem. (We originally had extra text explaining that special case but decided that ignoring the trivial case was better. Sorry if it confused you!)

[5] b. We first give an example to illustrate some terms: The binary number 1001011 has four 1 bits in it. If we add 1 to it, we get 1001100, which has one fewer 1 bits than the original value.

Now, prove the following theorem: Adding 1 to an unsigned binary integer with b bits cannot decrease the number of 1 bits in that number by more than b nor increase the number of 1 bits by more than 1.

Solution : Without loss of generality, let b be an arbitrary number of bits in an unsigned binary number.

We must show that if we add 1 to the number, we cannot decrease the number of 1s by more than b nor increase it by more than 1.

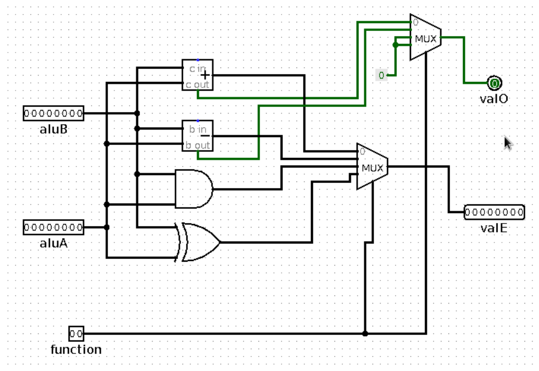
We begin with the former: The number cannot *have* more than b 1 bits; so, we clearly cannot decrease the number of 1s by more than b.

For the latter, note that when we add 1 to a binary number, we start from the right of the number "looking for" a 0. Each 1 we find changes to 0 and causes a carry. Eventually the first 0 becomes a 1. (If no 0 is found, depending on whether we increase the number of bits in the number or hold it fixed, we may or may not change a 0 to a 1, but this case does not change the argument.) So, up to one 0 becomes 1, which could increase the number of 1s by one but no more, and possibly some 1s become 0 (which cannot increase the number of 1s).

This completes our argument.

[4] 5. **Manifold Circuits:**

Consider the following diagram of an arithmetic logic unit (ALU) from lab.



The image contains two simple gates (AND and XOR), an addition module, a subtraction module, and two multiplexers. The two modules and the multiplexers are critical to the functioning of the ALU.

[2] a. Can the modules and the multiplexers be built from simple gates such as AND, OR, and NOT? **Briefly** justify your answer.

Solution : Yes, they can. We can describe the operation of any of the modules or the multiplexers with a truth table, and we have already established that AND, OR, and NOT are universal for truth tables. (In particular, but not needed for the exam, for each T row of the table, construct an "AND" clause where each variable whose value is T appears as a conjunct while the negation of each variable whose value is F appears as a conjunct. "OR" these clauses together, and we have a circuit for the truth table.)

[2] b. Briefly explain why the following statement is incorrect: "The `valO` output is only usable for `function` values of 0 and 1 because the MUX in the upper-right only has actual inputs for `function` values of 0 and 1."

Solution : The remaining two inputs (for `function` equal to 2 or 3) both have values; they just happen to be constant values. They're still perfectly legitimate and usable values.

[2] 6. **BONUS: I Can't Handle the Proof?**

Imagine we have a particular number of bits of memory available to represent an unsigned binary integer. We begin this memory at 0 and then repeatedly add 1 to it until we reach 0 again. So, for 3 bits, we get binary representations of 0, 1, 2, 3, 4, 5, 6, 7, and then 0.

Prove that the **average** number of bits that changes from 1 to 0 at each step is no greater than 1.

Some clever creation and manipulation of sums can solve the problem, particularly if you exploit an inequality in the problem to allow sums that are bounded (i.e., stop at some point) to instead go to infinity.

Most of the credit in the problem is available for **both** the proof structure and **substantial** progress on the proof itself, but we reserve the right to give little credit to messy, awkward, or otherwise ill-formed proofs. (It's a bonus problem!)

Solution : We do not provide a full solution, just a hint. What fraction of the bit patterns in a b bit number have the last bit equal to 0? Equal to 1? Of the ones where the last bit is 1, what fraction have the second bit from the right equal to 1? Of those where the last two bits are 1, what fraction have the third bit equal to 1? Etc.

So, in half of cases (ending in 0), we change NO 1s to 0s. (We change one 0 to 1.)

In half of cases (ending in 1), we change at least 1 bit from 1 to 0. That's $\frac{1}{2} \cdot 2^b$ (one half of the 2^b b-bit patterns). In a quarter of cases, we change *another* bit from 1 to 0. That's $\frac{1}{4} \cdot 1$. And so on.

Add up that sequence and what do you get? Divide it by n to get the average.

This fact turns out to be really handy for the occasional algorithm, by the way :)