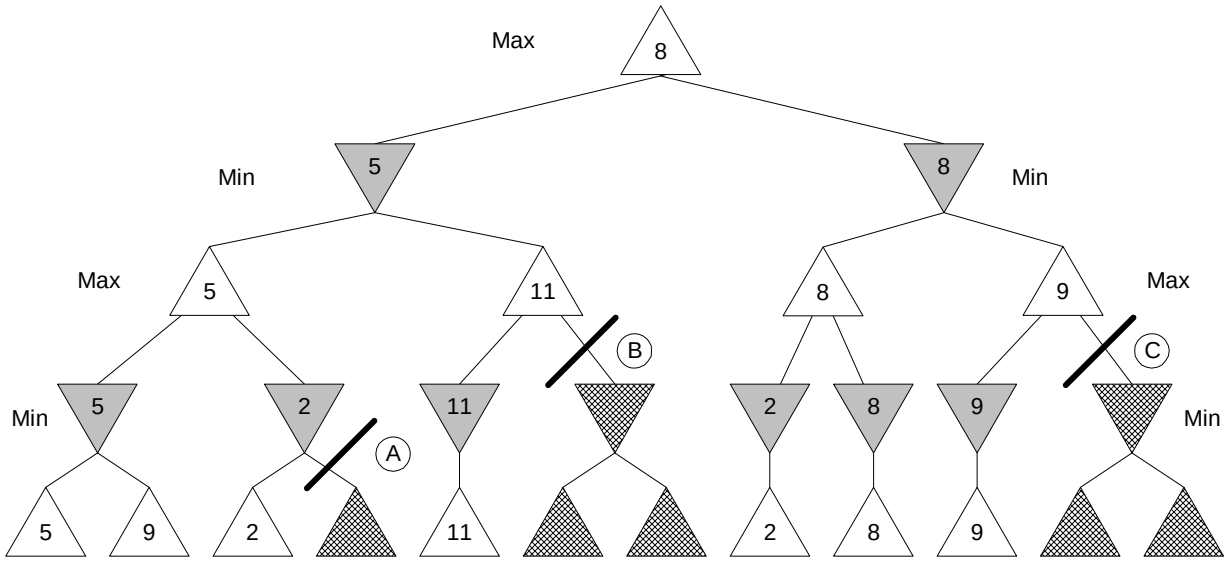
Answer Key
CSci 5551 Artificial Intelligence 1
April 8, 2008

Answer Key – 2nd Midterm

75 minutes == 75 points
Open book and notes

Question 1 – 15 points

Show the backed-up values for all the nodes, show the branches that are pruned and explain why they're pruned.



Above shows the backed up values for each node that is examined. No value is shown for nodes that were pruned since we did not need to look at those nodes.

White nodes are the results of your actions where, presumably, you picked the best (highest valued) option available to you. Grey nodes are the results of actions made by your opponent who, presumably, picked the option that was best for him (and, consequently, worst for you). So the white nodes should show the largest number directly under it and the gray nodes should show the smallest value directly below it.

There were three branches that were pruned, marked A, B and C.

A is pruned because 2 is less than 5. This is worth explaining in more detail. The leftmost max node at level three has two choices, 5 and 2. Given these two choices, Max will pick 5. We did not look at the node on level five below cut-point A. Why?

Consider two examples. If that node was 1, that's even smaller so Min would have picked it instead of 2. But it doesn't matter to us – we already know one of our options is 5 so anything

less than that isn't worth thinking about. So if the node we didn't evaluate is less than 2, it doesn't change our decision because we'll pick the 5.

On the other hand, suppose the value in the pruned node is 10. That's a lot better than the 5 we currently have. But it doesn't matter – Min gets to pick first and would have to pick between 2 and 10, so they'll pick 2. Meaning our choice would just be between 2 and 5, of which we'd pick 5. So if the node we didn't evaluate is greater than 2, it doesn't change our decision because Min will pick 2.

The second prune point is at B, right below level three. This is Max's choice. Max can either pick 11 or whatever is in the branch we pruned. If the number is less than 11, we won't pick it. If it's higher than 11, we'll pick it but it won't matter because that would make Min's choice at level 2 between 5 and 11, and they'll pick 5. So if it's less than 11, it won't change the value at level two, and it's more, it still won't change it.

The reasoning for prune point C is the same. Min already has 8 as its first option. Its second option is either 9 or what's in the branch we pruned. If it's less than 9, we won't pick it, if it's more, Min will choose 8. So it doesn't matter what the values in the pruned branch are, the value at level two will be 8.

Most people got the values correct but several missed the pruning. To inflate everyone's grades, i made the value part worth 10 points and the pruning worth 5. 1 point was taken off for each of the three prunings that was missed.

3 points was taken off for each mistake in backing up values. If an error was caused by an earlier error (e.g., picking 9 rather than 8 and then picking the 9 again at the next highest level), the error was only counted once. If the error was because of incorrect pruning (e.g., picking a pruned 1 over a non-pruned 2), the error was ignored.

Question 2 – 15 points

Prove by resolution

1. $\neg P \vee \neg Q \vee R$
2. $\neg S \vee Z$
3. $\neg Z \vee P$
4. S
5. $\neg R$
6. $\neg S \vee U$
7. $\neg U \vee Q$

General strategy: We have two facts (S and ¬R) and five rules. Facts are your friends. We should use the ones we have and try to get more.

$$\frac{S, \neg S \vee Z}{Z} \quad \left(\frac{facts\ 4\ and\ 3,\ joined\ on\ S}{our\ new\ fact\ 8} \right)$$

$$\frac{S, \neg S \vee U}{U} \quad \left(\frac{4\ \&\ 6}{9} \right)$$

$$\frac{Z, \neg Z \vee P}{P} \quad \left(\frac{8\ \&\ 3}{10} \right)$$

$$\frac{U, \neg U \vee Q}{Q} \quad \left(\frac{9\ \&\ 7}{11} \right)$$

OK, now we know the value of every variable, R-Z. All of them are true except for R. Let's do a quick check through our statements, seeing if we can find a contradiction. To make it easy, look through the smallest sentences first, which is everything with two terms. Any contradictions? Nope. So let's go look through the three term sentences, of which there's only 1. Does that have a contradiction? You betcha. To prove it, let's strip off its terms, one by one.

$$\frac{P, \neg P \vee \neg Q \vee R}{\neg Q \vee R} \quad \left(\frac{10\ \&\ 1}{12} \right)$$

$$\frac{Q, \neg Q \vee R}{R} \quad \left(\frac{11\ \&\ 12}{13} \right)$$

$$\frac{\neg R, R}{FALSE} \quad \left(\frac{5\ \&\ 13}{contradiction} \right)$$

This wasn't a super easy question (it's resolution after all) but almost everyone did a fantastic job on this.

Where people did make mistakes, it was in rushing. You can only cancel out one complementary pair of literals at a time. PvQvR and –Pv-Qv-R does not cancel everything out, it just results in Qv-QvRv-R which is valid (a tautology). For more explanation, see the answer key for the previous homework.

10 points was taken off if you did not prove a contradiction by eliminating variables one step at a time, showing all steps. Since the question was worth 15 and there weren't many places to get or lose credit, failure to prove the resolution using the proper technique took a lot of points off. But, since i felt bad, there were a few cases where i took off less if i thought the answer demonstrated that the student probably knew the proper way to do it despite not doing so.

2 points was taken off if you said two terms contradicted to get an empty set but did not actually show the final step that led to a contradiction.

Question 3 – 10 points (5 points each)

3.1. When doing constraint propagation, is backtracking search more efficient than generating and testing each combination of values? Why or why not?

You are given two options. Option 1, search for a solution using a semi-intelligent depth-first search that looks at one or more nodes (backtracking search), stopping when it finds the solution. Option 2, look at every single node, stopping when there are no more nodes to look at (generate and test). The question asks, which of these is more efficient, the option where you look at some of the nodes or the one where you look at all of them.

The answer, unsurprisingly, is that it is more efficient to look at fewer values. Not that depth-first search, unguided, is extremely efficient, especially given that you can search a given solution more than once (see the discussion on page 141). But a backtracking search gives you the option to use heuristics such as the most constrained value (MRV heuristic, page 143). As a result, to quote page 155, "backtracking search is commonly used for solving CSPs." If you generate and test each combination of values (an exhaustive search), you obviously don't prune anything.

A few people mentioned that forward checking with constraint propagation is as fast as backtracking. Unfortunately, forward checking wasn't an option here.

3 points was taken off for discussing forward checking, constraint propagation or arc connectivity rather than the two available choices, backtracking and exhaustive generate and test.

3.2. What are the advantages and disadvantages of using a complete-state formulation for CSP instead of using an incremental formulation?

Complete and incremental states were discussed in chapter three (page 66), when we first started discussing search. The constraint satisfaction chapter discussed it (briefly) on page 139.

There are several reasons why a complete-state formulation makes finding a solution easier. For example, it is a good format to use with local search methods and heuristics such as min-conflict. For online problems (problems where you get your data one piece at a time at runtime rather than as a complete set of data ahead of time), you can find solutions by patching (repairing) an existing solution or state. Most of this is discussed on page 150.

The alternative to a complete-state formulations is an incremental formulation. The book mentions that, if you represent the problem in this manner, you can treat a constraint satisfaction problem as a normal search problem, allowing you to leverage all the knowledge and techniques you have in solving search problems.

A lot of people had a rough time on this question. I can understand the disadvantages piece, since the book doesn't discuss it a lot (although it does give one in the chapter), but i'm surprised more

people didn't get the local search/heuristics/repair portion since the book spent almost an entire page discussing it.

A common mistake was to say that one approach used more memory or was faster. Neither is intrinsically true. Consider the traveling sales person problem from the first test (and answer key). Some people said the state was a list of Booleans, one for each city, that were initially false and set to true when a city was visited. This is a complete-state representation. Others said the state was an empty list and as a city was visited, it was added to the list. This is an incremental formulation. In neither case was the entire search space populated ahead of time. The complete-state formulation used a few bytes more memory at the start but by the time a solution was found, both states used the same amount of memory. Branching factors, state space and complexity are not required to be significantly different. Different representations simply make different algorithms, approaches and optimizations available.

There were several people who said complete-state formulations can't hold infinite states. Constraint satisfaction problems do not have infinite states. If they did, they couldn't be solved, regardless of the representation or algorithm. Think about a sudoku or crossword puzzle with an infinite number of squares, all of which must be filled.

Some people mentioned that complete-state formulations don't work when it's important to know the path to a solution. This never happens. A CSP problem is about finding a solution. It does not matter what order you enter the words in crossword puzzle as long as the final answer is right.

Some people said that complete-state formulations will find an answer but not the best one. There is no best answer in a CSP. Either something is an answer or it isn't.

3 points was taken off if the advantages were incorrect.

2 points was taken off for forgetting to mention a disadvantage of complete-state formulation, which a surprising number of people did.

1 point was taken off for an incorrect disadvantage.

Question 4 – 15 points (5 points each)

There was one specific sentence, #3 (the one the test warns you about), that had to be done properly to get the fact Greyhound(G). Aside from Amigo and Bunny, it is the only fact (as opposed to rule) that you have to use in resolution. In much of the test, i was generous (or perhaps lax) in taking off for how you did your groupings and which side of the implication you put your statements on. But for this specific problem, if you weren't very precise in how you did #3, the CNF would come out wrong and you couldn't continue the remainder of the problem.

4.1. Write the following statements in predicate calculus:

1. Horses are faster than dogs.

With a function:
1. $\forall$d,h Horse(h) $\wedge$ Dog(d) $\Rightarrow$ Speed(h)>Speed(d)

With a predicate:
1. $\forall$d,h Horse(h) $\wedge$ Dog(d) $\Rightarrow$ Faster(h, d)

Credit is given for using either a function or a predicate but, because it makes writing the solution for the resolution part of this problem easier to follow, i'll use the predicate version for the remainder of this question (for example of how to use the function version, see the answers to question 5 below).

2. Greyhounds are dogs.

2. $\forall$g Greyhound(g) $\Rightarrow$ Dog(g)

3. There is a greyhound that is faster than every rabbit.

$\exists$g Greyhound(g) $\wedge$ **(**$\forall$r Rabbit(r) $\Rightarrow$ Faster(g, r)**)**

You shouldn't have any $\exists$s when doing resolution so we'll have to Skolemize this sooner or later. i'm going to do it now:

Greyhound(G) $\wedge$ ($\forall$r Rabbit(r) $\Rightarrow$ Faster(G, r))

This can be broken into two facts:

3. Greyhound(G)
4. $\forall$r Rabbit(r) $\Rightarrow$ Faster(G, r)

There was a little confusion on particular sentence. Little things but they had big impact on CNF which in turn had a big impact on resolution

Note the parentheses in the original answer. You don't want to do this:

$$\exists g \ \forall r \ \textbf{(}Greyhound(g) \wedge Rabbit(r)\textbf{)} \Rightarrow Faster(g, r) \qquad (E1)$$

The error might not be obvious here, but it has an impact on the CNF. Look at the correct sentences 3 and 4 above. 3 will stand alone, 4 will turn into the CNF

Greyhound(G)
¬Rabbit(r) ∨ Faster(G, r)

We can't get this from the grouping in E1 above. We'd get the following:

∀r (Greyhound(G) ∧ Rabbit(r)) ⇒ Faster(G, r)
¬(Greyhound(G) ∧ Rabbit(r)) ∨ Faster(G, r)
¬Greyhound(G) ∨ ¬Rabbit(r) ∨ Faster(G, r)

We no longer know if Greyhound(G) exists. And if we don't know that, we can't solve the resolution piece of the question (well, unless you make another error later that accidentally corrects this one).

You should never use ∃ with ⇒, so the ∃ joins two terms with an ∧ while the ⇒ connects the pieces of the ∀ (since you should always use ⇒ with ∀). Which also means you shouldn't do this:

∃g ∀r Greyhound(g) ∧ Rabbit(r) ∧ Faster(g, r) *(E2)*

The above breaks into three independent facts and they aren't very useful facts. What are you going to do with the single fact like Rabbit(r)? Also consider the CNF.

∀r Rabbit(r) ∧ Faster(g, r)

turns into the CNF

¬Rabbit(r) ∨ ¬Faster(g, r)

The correct answer

∀r Rabbit(r) ⇒ Faster(G, r)

turns into the CNF

¬Rabbit(r) ∨ Faster(g, r)

Compare the above two. They're different in a pretty significant way.

You also shouldn't do the following:

∀r Rabbit(r) ⇒ ∃g Greyhound(g) ∧ Faster(g, r) *(E3)*

The syntax is correct but the logic is not. This says that every rabbit has at least one greyhound that is faster than it but these do not have to be the same greyhound.

4. Amigo is a horse and Bunny is a rabbit.

5. Horse(Amigo)
6. Rabbit(Bunny)

5. Faster is transitive – if x is faster than y and y is faster than z than x is faster than z

7. $\forall x,y,z$ Faster(x, y) $\wedge$ Faster(y, z) $\Rightarrow$ Faster(x, z)

4.2. Convert them to CNF

1. $\neg$Horse(h) $\vee$ $\neg$Dog(d) $\vee$ Faster(h, d)
2. $\neg$Greyhound(g) $\vee$ Dog(g)
3. Greyhound(G)
4. $\neg$Rabbit(r) $\vee$ Faster(G, r)
5. Horse(Amigo)
6. Rabbit(Bunny)
7. $\neg$Faster(x, y) $\vee$ $\neg$Faster(y, z) $\vee$ Faster(x, z)

You should not have quantifiers in your CNF format. Universals $\forall$ can be dropped since everything will be assumed to be universal. Existentials $\exists$ need to be replaced by a Skolem constant. Statements connected by $\wedge$ needed to be split into the independent sentences.

As Maria mentioned when we did the last resolution example, it makes our life a lot easier if we use different variable names for the different variables in our knowledge base. If you don't, there is a strong risk during the unification step that you will unify the wrong the wrong things, causing errors in your logic. So above we have the three objects (Amigo, Bunny and our mysterious Skolemized G) and the variables h for horse, g for greyhound, d for dog, r for rabbit and x,y,z for use in a single generic sentence.

1 point was taken off if the sentence was not in CNF format. This included leaving the $\wedge$ in.

2 points was taken off if you did not replace the existential quantifiers with a Skolem constant.

2 points was taken off if you did not give each sentence unique variable names.

4.3. Prove by resolution that "Amigo is faster than Bunny"

First, let's add one more fact, the opposite of what we hope is true:

8. $\neg$Faster(Amigo, Bunny)

Now for resolution. Different people do this differently. The solution that i used below takes 7 steps which i think is the shortest number of steps you can do this in.

$$\frac{3.\,Greyhound(G),\ 2.\,\neg Greyhound(g) \vee \neg Dog(g)}{9.\,\neg Dog(G)}\ \{g/G\}$$

$$\frac{6.\,Rabbit(Bunny),\ 4.\,\neg Rabbit(r) \vee Faster(G,r)}{10.\,Faster(G,Bunny)}\ \{r/Bunny\}$$

$$\frac{5.\,Horse(Amigo),\ 1.\,\neg Horse(h) \vee \neg Dog(d) \vee Faster(h,d)}{11.\,\neg Dog(d) \vee Faster(Amigo,d)}\ \{h/Amigo\}$$

$$\frac{9.\,Dog(G),\ 11.\,\neg Dog(d) \vee Faster(Amigo,G)}{12.\,Faster(Amigo,G)}\ \{d/G\}$$

$$\frac{12.\,Faster(Amigo,G),\ 7.\,\neg Faster(x,y) \vee \neg Faster(y,z) \vee Faster(x,z)}{13.\,\neg Faster(G,z) \vee Faster(Amigo,z)}\ \{x/Amigo,\ y/G\}$$

$$\frac{10.\,Faster(G,Bunny),\ 13.\,\neg Faster(G,z) \vee Faster(Amigo,z)}{14.\,Faster(Amigo,Bunny)}\ \{z/Bunny\}$$

$$\frac{14.\,Faster(Amigo,Bunny),\ 8.\,\neg Faster(Amigo,Bunny)}{FALSE}$$

Good style dictates that you show the unifications used at each step. You should use the original variable names in the top portion, the substitutions in the result.

There were several common resolution errors. One was trying to unify sentences that had ∧s in them. You can't do that. One was trying to resolve three sentences at a time. You can't do that, it has to be two sentences. Another was canceling out multiple terms at once. A ∨ B and ¬A ∨ ¬B do not cancel out to the empty set. Another was trying to join two sentences on a literal where it had the same value in each sentence. You cannot join A ∨ B with A ∨ C. Another was adding two sentences together. You cannot take the sentences Dead ∨ ¬Dead and LikesPeanutButter and stick them together to make the sentence Dead ∨ ¬Dead ∨ LikesPeanutButter (although if you could, if you liked peanut butter you could then be both not dead and not not dead, proving peanut butter makes you a vampire).

There were a lot of resolutions that included solutions similar to:

Faster(x,y)
Faster(Amigo, Bunny) {x/Amigo, y/Bunny}
Faster(Bunny, Amigo) {x/Bunny, y/Amigo}

$$\frac{Faster(Amigo, Bunny),\ \neg Faster(Amigo, Bunny)}{FALSE}$$

If you think about it a second, i'm hoping you consider this particular error obvious. If you could plug any value into any variable under any circumstance, logic wouldn't be very logical would it? Everything would be true simply because you wanted it to be. But since all variables are considered universally instantiated (they're true for everyone), it appears you really can do the above. So how can logic work then? The answer is that you should never be able to get that first sentence, Faster(x,y). In every situation where i found this, it was because someone had not carried through a constant after unification (or made up sentences out of thin air thinking i wouldn't notice). Faster(G,y) is not the same thing as Faster(x,y). You cannot replace a constant with a variable. The sentence

Faster(SantasLittleHelper, Bunny)

is not the same as

Faster(x, Bunny) {SatansLittleHelper ,x}

Remember that the above is short hand for (the lazy way of writing)

$\forall x$ *Faster(x, Bunny)*

If the first sentence was equivalent to the second then SantasLittleHelper being faster than Bunny would mean that everyone was faster than Bunny, even Bunny.

There were numerous mistakes with unification, especially when Skolem constants were used. A common example:

$A(Horse)$
$\neg A(x) \vee B(x, y)$
$Z(Bunny)$
$\neg Z(m) \vee \neg B(m, q)$

$$\frac{A(Horse), \neg A(x) \vee B(x, y)}{B(x, y)}$$
$$\frac{Z(Bunny), \neg Z(m) \vee \neg B(m, q)}{\neg B(m, q)}$$
$$\frac{B(x, y), \neg B(m, q)}{FALSE}$$

If you unify ¬A(S) with A(x) ∨ B(x, y), you can do {x/S} (replace x with S) but you cannot do {S/x} (replace single instance with everybody). If you do the former, the resulting sentence is B(S, y), not B(x, y). The proper way to do the above example is:

$$\frac{A(Horse), \neg A(x) \vee B(x, y)}{B(Horse, y)} \{x / Horse\}$$
$$\frac{Z(Bunny), \neg Z(m) \vee \neg B(m, q)}{\neg B(Bunny, q)} \{m / Bunny\}$$
$$\frac{B(Horse, y), \neg B(Bunny, q)}{This\ is\ an\ illegal\ step}$$

You cannot turn a horse into a bunny. This is one of the many important lessons you can learn in AI.

Another common mistake was to not do the entire unification. Some people said that A(S), ¬A(x) ∨ B(x,y) ∨ C(x) resolve to B(S,y) ∨ C(x). They replaced one of the x's but not all of them. Most people probably knew they had to carry it through and just didn't notice the extra term. A timed test can be stressful but it's important to take the time to check the little details. In something like resolution, an error early will throw off the rest of your solution.

There were a ***lot*** of mistakes due to thinking a sentence said something it did not (e.g., thinking you had A ∨ B when you really had ¬A ∨ B or A ∨ B ∨ C). Many times people forgot about some of the predicates in a sentence (e.g., A ∨ B, ¬A ∨ B ∨ C turned into B rather than B ∨ C). In almost every case, it was someone who did not write the complete resolution formula out and instead tried to do it in their head. The same was true for many of the unification mistakes. Once again, i want to point out that writing these rules out not only makes your work significantly easier to grade, it also helps you catch mistakes. Make sure to write things out, including the substitutions, and that you carry all the parts of the sentence over.

1 point was taken off if you did not show the unification. I was otherwise fairly generous on this.

1-2 points were taken off if you wrote the resolution in an improper format (e.g., joining two independent clauses with $\vee$ rather than a comma, making it hard to see where one clause ended and the next began) that made grading more slow and difficult than normal. I know there was time pressure so i didn't take off for a lot of the sloppy formatting i saw but if an error made it significantly harder to grade and could have been done in the same amount of time it took to do it wrong, i took off.

3 points was taken off if you failed to properly do resolution because you didn't have the fact Greyhound(G) or hadn't used a Skolem constant but effort was shown and you were otherwise correct.

4 points were taken off if your resolution was completely wrong but you tried something.

4 points were taken off if you did not use resolution and instead tried to prove the answer by modus ponens or some other approach.

Question 5 – 20 points (4 points each)

For each of the following sentences, decide if the logic sentence given is a correct translation of the English sentence or not (hint: it's not). If not, explain why not and correct it:

1 point was taken off if your explanation of the problem was incorrect.
2 points were taken off if your corrected sentence was not correct.
3 points were taken off if you said the sentence was fine.

A lot of people seemed to struggle with this section. Feeling bad, i decided to give you ½ credit if you just guessed that sentence was false and made some attempt at explaining and correcting it, no matter how bad. Even if you made your decision by flipping a coin and had no idea what was wrong with the sentence or how to fix it, i gave you ½ credit. I wouldn't normally do this (and won't on the final) but a lot of people needed the points. ☹

If you did absolutely everything wrong (guessed the sentence was correct, no explanation or correction), i went ahead and gave you a point for at least reading the question and making a guess. So you got 5 points just for playing. The only way to lose the full 20 points was to skip this section of the test.

5.1. There is exactly one house in Minneapolis whose cost is $300,000.
$\exists x$ ***house(x)*** $\wedge$ ***in(x, Minneapolis)*** $\wedge$
$\forall y$ **[*house(y)* $\wedge$ *in(y, Minneapolis)* $\wedge$ *cost(y)*=300000)** $\Rightarrow$ ***x=y*****]**

Error: missing cost(x)=300000 in part with existential quantifier.

The first part of this sentence says there's at least one house in Minneapolis. The second part says that if there happens to be one that costs $300,000 then there's only one. Put together this means that there is between 0 and 1 houses in Minneapolis that cost $300,000. If we had wanted to say that there is exactly one we would have needed:

$\exists$x house(x) $\wedge$ in(x, Minneapolis) $\wedge$ **cost(x)=300000** $\wedge$
$\forall$y [house(y) $\wedge$ in(y, Minneapolis) $\wedge$ cost(y)=300000) $\Rightarrow$ x=y]

5.2. Any house in Minneapolis costs less than any apartment in New York.
$\forall x$ **[*house(x)* $\wedge$ *in(x, Minneapolis)*] $\Rightarrow$**
[$\exists y$ *apartment(y)* $\wedge$ *in(y, NewYork)* $\wedge$ *cost(x)<cost(y)*]

Error: *y* should be universally quantified.

This says that, if that are any houses in Minneapolis, every one of them is cheaper than at least one apartment in New York. In short, all houses in Minneapolis are cheaper than some apartments in New York. What we need to have said was:

$\forall$x,**y** house(x) $\wedge$ in(x, Minneapolis) $\wedge$ apartment(y) $\wedge$ in(y, NewYork) $\Rightarrow$ cost(x)<cost(y)

5.3. Some apartments in Minneapolis cost less than some houses in New York.
$\forall x$ **[*apartment(x)* $\wedge$ *in(x, Minneapolis)* $\wedge$ $\exists y$ *house(y)* $\wedge$ *in(y, NewYork)*] $\Rightarrow$**
cost(x)<cost(y)

Error: x should be $\exists$ instead of $\forall$ (which means $\wedge$ instead of $\Rightarrow$).

The above says that, if there are any apartments in Minneapolis and at least one house in New York, every single apartment in Minneapolis is cheaper than that New York house. In short, all apartments in Minneapolis are cheaper than all houses in New York.

$\exists$x apartment(x) $\wedge$ in(x, Minneapolis) $\wedge$ $\exists$y house(y) $\wedge$ in(y, NewYork) **$\wedge$** cost(x)<cost(y)

5.4. All houses have at least one bathroom.
$\forall x\ [house(x) \wedge \exists y\ bathroom(y)] \Rightarrow in(x, y)$

Error: $\wedge$ should be $\Rightarrow$ and $\Rightarrow$ should be $\wedge$.

The above says that, if there exists in the world a house and a bathroom, the bathroom is in the house. This can also be read as, if bathrooms exist, there is a specific one that is in every house. If everyone in the world gets to use that one bathroom, the line to get into it must be very long.

What we should have said was:

$\forall x$ house(x) $\underline{\Rightarrow}$ $\exists y$ bathroom(y) $\underline{\wedge}$ in(x, y)

5.5. There is a house in Minneapolis which costs more than any other house.
$\forall x\ [house(x) \wedge in(x, Minneapolis)] \Rightarrow [\exists y\ house(y) \wedge cost(y) > cost(x)]$

Error: $\exists$ should be outside of $\forall$, need to have $x \neq y$.

This says that if there are any houses in Minneapolis, they all cost more than some house. Which means that the most expensive house in Minneapolis must cost more than itself. What we should have said is:

$\underline{\exists \mathbf{y}}$ house(y) $\wedge$ in(y, Minneapolis) $\wedge$ [$\forall$x house(x) $\wedge$ $\underline{\mathbf{x \neq y}}$ $\underline{\Rightarrow}$ cost(y)>cost(x)]

This version says that there is a house in Minneapolis that is more expensive than any other house anywhere, whether in Minneapolis or not. The original sentence, like many in English, is a bit vague. One might assume it says there is a house in Minneapolis that is more expensive than any other house in Minneapolis. If this is how you interpreted the sentence, you would have had:

$\underline{\exists \mathbf{y}}$ house(y) $\wedge$ in(y, Minneapolis) $\wedge$ [$\forall$x house(x) $\wedge$ in(x, Minneapolis) $\underline{\mathbf{x \neq y}}$ $\underline{\Rightarrow}$ cost(y)>cost(x)]