Homework 7

Assigned on: Friday, November 1st, 2019.
Due: Friday, November 15th, 2019.

Points: 100, plus a potential 20 bonus points in the main tract. Additionally, you have the option of implementing a SAT solver for 100 additional bonus points.

Contents

1. AIMA, Exercise 7.1, page 279. (16 points) 1
2. AIMA, Exercise 7.7, page 281. (6 points) 1
3. Truth Tables (8 points) 1
4. AIMA, Exercise 7.10, page 281. (16 points) 1
5. Logical Equivalences (8 points) 2
6. AIMA, Exercise 7.22, page 284. (18 points + 20 bonus) 2
7. Proofs (28 points) 2
8. Bonus: Implementation, Solving SAT (100 points) 3

Alert: If you submit your homework handwritten, it must be absolutely neat or it will not be corrected. If you type your homework (preferable), submit using webhandin.

1. AIMA, Exercise 7.1, page 279. (16 points)
2. AIMA, Exercise 7.7, page 281. (6 points)
3. Truth Tables (8 points)

Use truth tables to show that each of the following is a tautology.

1. \((p \land q) \rightarrow \neg(p \lor \neg q)\)
2. \([\text{Mary} \land (\text{Mary} \rightarrow \text{Susy})] \rightarrow \text{Susy}\)
3. \(\alpha \rightarrow [\beta \rightarrow (\alpha \land \beta)]\)
4. \((a \rightarrow b) \rightarrow [(b \rightarrow c) \rightarrow (a \rightarrow c)]\)

4. AIMA, Exercise 7.10, page 281. (16 points)

Only b, c, d, e, f, and g.
5 Logical Equivalences

Using a method of your choice, verify:

1. \((\alpha \rightarrow \beta) \equiv (\neg \beta \rightarrow \neg \alpha)\) contraposition
2. \(\neg(\alpha \land \beta) \equiv (\neg \alpha \lor \neg \beta)\) de Morgan
3. \((\alpha \land (\beta \lor \gamma)) \equiv ((\alpha \land \gamma) \lor (\alpha \land \beta))\) distributivity of \(\land\) over \(\lor\)

6 AIMA, Exercise 7.22, page 284. (18 points + 20 bonus)

Parts a, b, and c are required. Parts d, e, and f are bonus.

7 Proofs (28 points)

Give the explanations of each step if the steps are given, and give both the explanation and step if they are not.

- If \(q \land (r \land p), t \rightarrow v, v \rightarrow \neg p,\) then \(\neg t \land r.\)

  **Proof**
  
  1. \(q \land (r \land p)\)  
  2. \(t \rightarrow v\)  
  3. \(v \rightarrow \neg p\)  
  4. \(t \rightarrow \neg p\)  
  5. \((r \land p)\)  
  6. \(r\)  
  7. \(p\)  
  8. \(\neg p\)  
  9. \(\neg t\)  
  10. \(\neg t \land r\)

- If \(p \rightarrow (q \land r), q \rightarrow s,\) and \(r \rightarrow t,\) then \(p \rightarrow (s \land t).\)

  **Proof**
  
  1. \(\)  
  2. \(\)  
  3. \(\)  
  4. \(\)  
  5. \(\)  
  6. \(\)  
  7. \(\)

- Prove by contradiction.

  If \(\neg(\neg p \land q), p \rightarrow (\neg t \lor r), q,\) and \(t,\) then \(r.\)

  **Proof**
  
  1. \(\neg(\neg p \land q)\)  
  2. \(p \rightarrow (\neg t \lor r)\)  
  3. \(\)  
  4. \(\)  
  5. \(\)  
  6. \(\)  
  7. \(\)
3. \( q \) \quad \text{Given}
4. \( t \) \quad \text{Given}
5. \( \neg r \) \quad \text{Negation of Conclusion}

8 Bonus: Implementation, Solving SAT (100 points)

Write a search algorithm to determine the satisfiability of a SAT instance. You can either write:

- A DPLL procedure (backtrack search),
- A local search procedure.

You must

- Clearly describe, in addition to your code, your data structures, how your search algorithm operates, and the improvements, if any, that you have included in your code.
- We recommend that you use the standard file for input files known as the ‘simplified version of the DIMACS format’:
- Test the performance of your algorithm on some non trivial uniform random instances taken from the SAT Competition. For example:
  http://www.cs.ubc.ca/~hoos/SATLIB/benchm.html

Alert: many implementations exist in the literature and on the web. We expect you to do your own implementation.