Joseph Morgan

Homework 10.42

CISP440

1 Source Code

```
#include <cstdio>

bool implication (bool a, bool b);
bool bicondition (bool a, bool b);
bool maybe (bool a, bool b);
bool because (bool a, bool b);
bool logic_and (bool a, bool b);
bool logic_or (bool a, bool b);
bool logic_xor (bool a, bool b);
bool logic_not (bool a);

int main()
{
  bool r1;
  bool r2;
  bool equiv = true;
  // a
  printf("| p q | p or q | ~(~p and ~q) |\n");
  printf("-------------------------------\n");
  for (int p = 0; p < 2; ++p) {
    for (int q = 0; q < 2; ++q) {
      r1 = logic_or (p, q);
      r2 = logic_not(logic_and(logic_not(p), logic_not(q)));
      if (r1 != r2)
        equiv = false;
      printf ("| %i %i | %6i | %12i |\n", p, q, r1, r2);
    }
  }
  equiv ? printf("\nExpressions are equivalent\n") : printf("\nExpressions are
     not equivalent\n");
  printf("\n\n");
  equiv = true;

  // b

  printf("| p q r | p && (q -> r) | (p && q) ^ r |\n");
  printf("-----------------------------------------\n");
  for (int p = 0; p < 2; ++p) {
    for (int q = 0; q < 2; ++q) {
      for (int r = 0; r < 2; ++r) {
        r1 = logic_and (p, implication (q, r));
        r2 = logic_xor (logic_and (p, q), r);
        if (r1 != r2)
          equiv = false;
        printf ("| %i %i %i | %12i | %11i |\n", p, q, r, r1, r2);
      }
    }
  }
  equiv ? printf("\nExpressions are equivalent\n") : printf("\nExpressions are
     not equivalent\n");
  printf("\n\n");
```

```
50    equiv = true;
51
52    // c
53
54    printf("| p q | p bicondition q | ~(p xor q) |\n");
55    printf("-------------------------------------\n");
56    for (int p = 0; p < 2; ++p) {
57      for (int q = 0; q < 2; ++q) {
58        r1 = bicondition (p, q);
59        r2 = logic_not (logic_xor (p, q));
60        if (r1 != r2)
61          equiv = false;
62        printf ("| %i %i | %15i | %10i |\n", p, q, r1, r2);
63      }
64    }
65    equiv ? printf("\nExpressions are equivalent\n") : printf("\nExpressions are
          not equivalent\n");
66    printf("\n\n");
67    equiv = true;
68
69    // d
70
71    printf("| p q r | (p <-> q) -> r | p & (~q | r) |\n");
72    printf("-----------------------------------------\n");
73    for (int p = 0; p < 2; ++p) {
74      for (int q = 0; q < 2; ++q) {
75        for (int r = 0; r < 2; ++r) {
76          r1 = implication (bicondition (p, q), r);
77          r2 = logic_and (p, logic_or (logic_not(q), r));
78          if (r1 != r2)
79            equiv = false;
80          printf ("| %i %i %i | %14i | %12i |\n", p, q, r, r1, r2);
81        }
82      }
83    }
84    equiv ? printf("\nExpressions are equivalent\n") : printf("\nExpressions are
          not equivalent\n");
85    printf("\n\n");
86    equiv = true;
87
88    // e
89
90    printf("| p q r | ~(p -> (q & r)) | p & ~(q & r) |\n");
91    printf("---------------------------------------------\n");
92    for (int p = 0; p < 2; ++p) {
93      for (int q = 0; q < 2; ++q) {
94        for (int r = 0; r < 2; ++r) {
95          r1 = logic_not (implication (p, logic_and (q, r)));
96          r2 = logic_and (p, logic_not (logic_and (q, r)));
97          if (r1 != r2)
98            equiv = false;
99          printf ("| %i %i %i | %15i | %12i |\n", p, q, r, r1, r2);
100       }
101     }
```

```
102   }
103   equiv ? printf("\nExpressions are equivalent\n") : printf("\nExpressions are
          not equivalent\n");
104   printf("\n\n");
105   equiv = true;
106
107   // f p | (p ? q)      p & (p @ q)k
108
109   printf("| p q | p | (p ? q) | p & (p @ q) |\n");
110   printf("-------------------------------\n");
111   for (int p = 0; p < 2; ++p) {
112     for (int q = 0; q < 2; ++q) {
113       r1 = logic_or (p, maybe (p, q));
114       r2 = logic_and (p, because (p, q));
115       if (r1 != r2)
116         equiv = false;
117       printf ("| %i %i | %11i | %11i |\n", p, q, r1, r2);
118     }
119   }
120   equiv ? printf("\nExpressions are equivalent\n") : printf("\nExpressions are
          not equivalent\n");
121   printf("\n\n");
122   equiv = true;
123 }
124
125 bool implication (bool a, bool b)
126 {
127   return ((!a) || b);
128 }
129
130 bool bicondition (bool a, bool b)
131 {
132   return logic_and(implication (a, b), implication (b, a));
133 }
134
135 bool maybe (bool a, bool b)
136 {
137   if (!a && b)
138     return true;
139   return false;
140 }
141
142 bool because (bool a, bool b)
143 {
144   b = b % 2;
145   if (a)
146     return true;
147   return false;
148 }
149
150 bool logic_and (bool a, bool b)
151 {
152   return (a && b);
153 }
```

```
154
155 bool logic_or (bool a, bool b)
156 {
157   return (a || b);
158 }
159
160 bool logic_xor (bool a, bool b)
161 {
162   return (a != b);
163 }
164
165 bool logic_not (bool a)
166 {
167   return (!a);
168 }
```

2 Output

```
1  | p q | p or q | ~(~p and ~q) |
2  ------------------------------
3  | 0 0 |      0 |            0 |
4  | 0 1 |      1 |            1 |
5  | 1 0 |      1 |            1 |
6  | 1 1 |      1 |            1 |
7
8  Expressions are equivalent
9
10
11 | p q r | p & (q -> r) | (p & q) ^ r |
12 ---------------------------------------
13 | 0 0 0 |            0 |           0 |
14 | 0 0 1 |            0 |           1 |
15 | 0 1 0 |            0 |           0 |
16 | 0 1 1 |            0 |           1 |
17 | 1 0 0 |            1 |           0 |
18 | 1 0 1 |            1 |           1 |
19 | 1 1 0 |            0 |           1 |
20 | 1 1 1 |            1 |           0 |
21
22 Expressions are not equivalent
23
24
25 | p q | p bicondition q | ~(p xor q) |
26 -------------------------------------
27 | 0 0 |               1 |          1 |
28 | 0 1 |               0 |          0 |
29 | 1 0 |               0 |          0 |
30 | 1 1 |               1 |          1 |
31
32 Expressions are equivalent
33
34
35 | p q r | (p <-> q) -> r | p & (~q | r) |
```

p q r	~(p -> (q & r))	p & ~(q & r)
0 0 0	0	0
0 0 1	1	0
0 1 0	1	0
0 1 1	1	0
1 0 0	1	1
1 0 1	1	1
1 1 0	0	0
1 1 1	1	1

Expressions are **not** equivalent

p q r	~(p -> (q & r))	p & ~(q & r)
0 0 0	0	0
0 0 1	0	0
0 1 0	0	0
0 1 1	0	0
1 0 0	1	1
1 0 1	1	1
1 1 0	1	1
1 1 1	0	0

Expressions are equivalent

p q	p	(p ? q)	p & (p @ q)
0 0	0	0	0
0 1	1	1	0
1 0	1	1	1
1 1	1	1	1

Expressions are **not** equivalent