LABORATORIUM PEMBELAJARAN ILMU KOMPUTER

FAKULTAS ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

BAB	: AGEN LOGIKA
NAMA	: REZA AZZUBAIR WIJONARKO
NIM	: 155150200111182
TANGGAL	: 03/05/2017
JENIS	: LATIHAN
ASISTEN	: - ANNISA FITRIANI NUR - RISKI PUSPA DEWI D. P..

ACC

A. DEFINISI MASALAH

1. Selesaikan contoh tabel kebenaran berikut:

Let $\alpha = A \vee B$ and $KB = (A \vee C) \wedge (B \vee \neg C)$

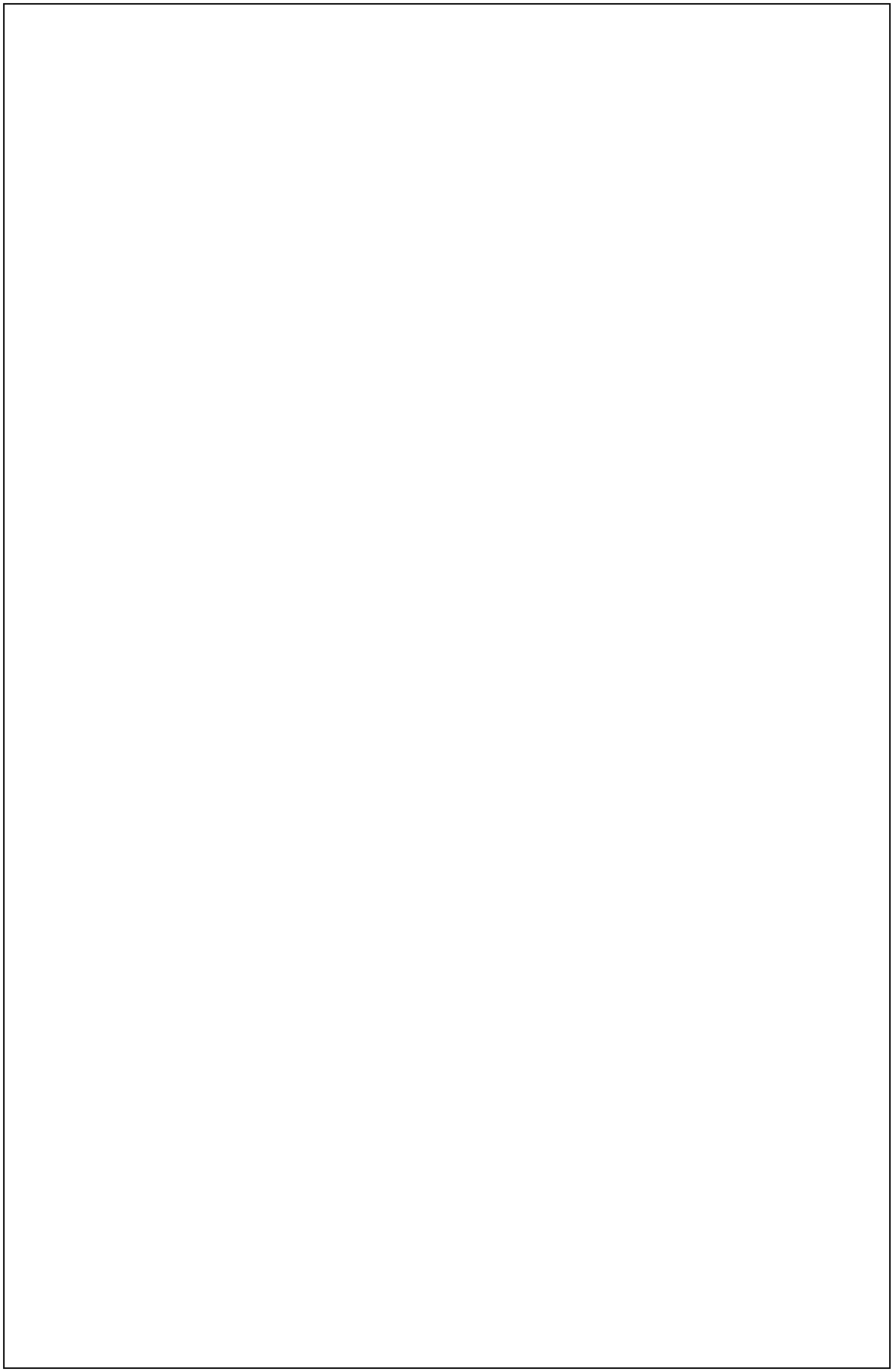
Is it the case that $KB \models \alpha$?
Check all possible models—α must be true wherever KB is *true*

A	B	C	$A \vee C$	$B \vee \neg C$	KB	α
False	*False*	*False*				
False	*False*	*True*				
False	*True*	*False*				
False	*True*	*True*				
True	*False*	*False*				
True	*False*	*True*				
True	*True*	*False*				
True	*True*	*True*				

2. Jika diberikan fakta seperti di bawah ini
   - Jika banyak diskon dan tanggal muda maka Gramedia ramai pengunjung
   - Jika mau Tahun Ajaran Baru sekolah maka Gramedia juga melakukan diskon
   - Gramedia Sepi
   - Mau Tahun Ajaran Baru sekolah

   Pertanyaan:
   a. Nyatakan setiap fakta pada setiap statement di atas dengan simbol proposisi (1 statement 1 simbol). Misalay: tanggal muda disimbolkan dengan T
   b. Buat KB (*knowledge base*) mengenai fakta di atas (dari hasil jawaban a. yakni dengan menggunakan simbol yang telah dibuat)
   c. Dari b., lakukan penalaran dengan *entailment*: “Apakah sekarang lagi tanggal tua?” penalaran tersebut bisa dijawab dengan menggunakan Tabel kebenaran (*truth table*)
   d. Coba cek apakah KB | = α, di mana α = tanggal tua

B. JAWAB



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

BAB	: AGEN LOGIKA
NAMA	: REZA AZZUBAIR WIJONARKO
NIM	: 155150200111182
TANGGAL	: 03/05/2017
JENIS	: TUGAS
ASISTEN	: - ANNISA FITRIANI NUR - RISKI PUSPA DEWI D. P..

ACC

A. DEFINISI MASALAH

1. Jelaskan konsep dasar dari agen berbasis pengetahuan dan hal-hal apa saja yang harus dipenuhi ketika membuat agen tersebut
2. Berilah contoh kasus entailment dari permainan Wumpus World berdasarkan aturan yang diberikan
3. Dari soal nomor 2, tunjukkan dengan metoda pembuktian baik model checking ataupun menggunakan tabel kebenaran. (minimal 3 kalimat dan maksimal 6 kalimat)

   Contoh pengerjaan :

   Diketahui KB = $(B_{1,1} \Leftrightarrow (P_{2,1} \vee \neg P_{3,1}) \vee \neg B_{1,1})$ dan $\sigma = \neg B_{1,1} \vee P_{2,1}$
   Buktikan dengan tabel kebenaran berikut, apa saja kondisi yang memenuhi KB|=σ

	$B_{1,1}$	$P_{2,1}$	$P_{3,1}$	$\neg B_{1,1} \vee P_{2,1}$	$B_{1,1} \Leftrightarrow (P_{2,1} \vee \neg P_{3,1})$	KB	σ
1	T	F	F				
2	T	T	F				
3	T	T	T				
4	F	T	F				
5	F	F	F				
6	F	T	T				

4. Carilah kode program dari library atau lainnya dan kemudian modifikasi yang digunakan untuk mengimplementasi konsep agen berbasis logika dengan menerapkan pula proses inferensi dengan cara entailment yang kemudian dibuktikan dengan model checking dan tabel kebenaran.

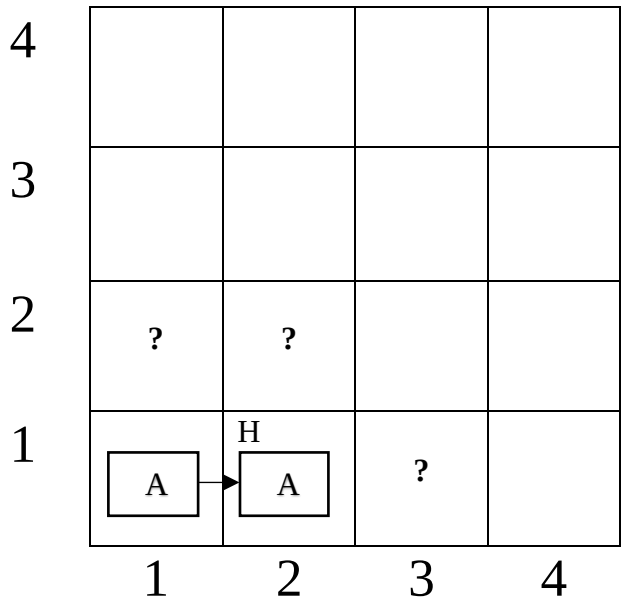
B. JAWAB

1. Konsep dasar agen berbasis pengetahuan:
   Mengetahui hal-hal tentang dunia dan dapat melakukan *reasoning* (berpikir, bernalar) mengenai:
   - Hal – hal yang tidak diketahui sebelumnya (*imperfect*/*partial information*)
   - Tindakan yang paling baik untuk diambil (*best action*)

   Maka sebuah agen berbasis pengetahuan harus bisa:
   - Merepresentasikan world, state, action, dst.
   - Menerima informasi baru (dan meng-udpate representasinya).
   - Menyimpulkan pengetahuan lain yang tidak eksplisit (*hidden property*).
   - Menyimpulkan action apa yang perlu diambil
2. Aturan permainan Wumpus World
   - Perfomance measure : emas +1000, mati -1000, gerak -1, panah-10
   - Environment : Matriks 4 x 4 ruang dengan initial state [1,1]. Ada emas, wumpus, dan lubang yang lokasinya dipilih secara acak.
   - Percept terdiri dari :
     - Hembusan angin : kamar disamping lubang jebakan ada hembusan angin.
     - Kilauan : kamar dimana ada emas ada kilauan/ sinar.
     - Bau : kamar disamping Wumpus berbau busuk (stench).
   - Action : maju, belok kiri 90°, belok kanan 90°, tembak panah (hanya 1!), ambil benda.

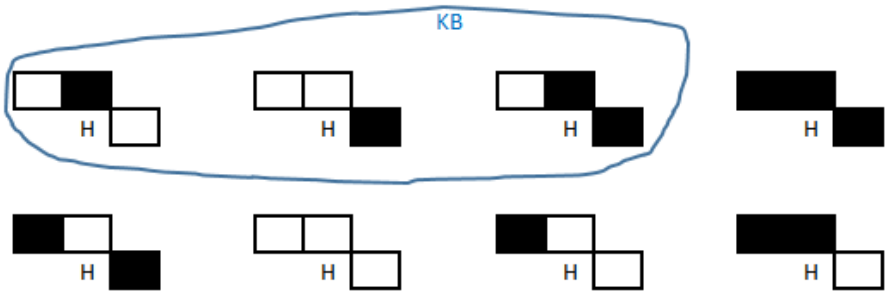
   Contoh kasus entailment

   Pada kotak [1,1] aman, kemudian agen berpindah ke kotak [1,2] dan terdapat H (hembusan angin) pada kotak tersebut yang menandakan adanya L (Lubang) disekitar kotak.

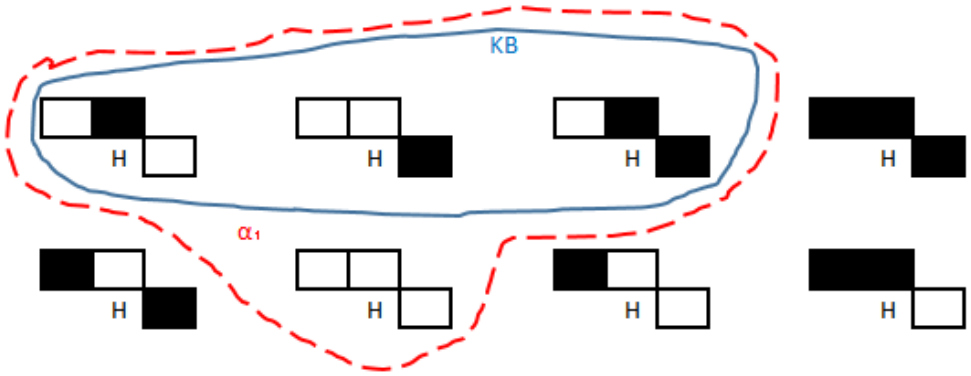


Keterangan
A : Agen
H : Hembusan Angin
E : Emas
L : Lubang
B : Bau Busuk
T : Terkunjungi
W : Wumpus
K : Kilauan

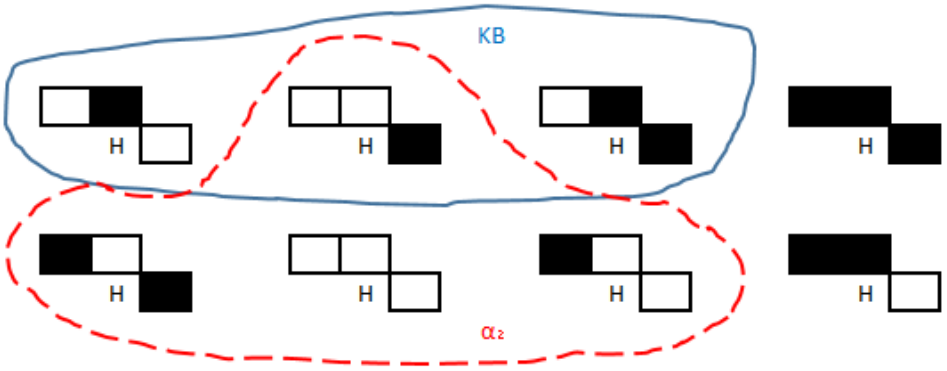
Model jebakan terdapat 3 pilihan boolean yaitu [1,2], [2,2], dan [1,3] dengan 8 kemungkinan model yaitu :



3. Model-model checking
   - KB = pengamatan (percept) + aturan-aturan Wumpus World
     Untuk menyatakan apakah kamar [2,1] aman atau tidak aman, maka dilakukan entailment yang dibuktikan dengan menggunakan model checking, yakni memeriksa semua kemungkinan M(KB), M(α_1).



     Dari ilustrasi ini maka dapat dilihat bahwa M(KB) subset dari M(α_1), sehingga disimpulkan bahwa KB |= α_1, dengan kata lain kamar [2,1] aman.
   - Untuk pengamatan selanjutnya mengecek apakah kamar [2,2] aman. Ilustrasi α_2 dapat digambarkan sebagai berikut:



Ilustrasi di atas menunjukkan M(KB) bukan subset dari M(α_2) sehingga bisa disimpulkan bahwa KB| $\neq \alpha_2$, atau dengan kata lain kamar [2,2] tidak aman.

Tabel Kebenaran

KB = $B_{1,2} \wedge (P_{3,1} \vee P_{2,2})$

α_1 = Menyatakan kamar di [1,2] aman

$B_{1,1}$	$B_{1,2}$	$P_{1,2}$	$P_{2,2}$	$P_{3,1}$	$P_{2,1}$	KB	α_1
False	False	False	False	False	False	F	T
False	False	False	False	True	False	F	T
False	True	False	False	False	False	F	T
False	True	False	True	False	False	T	T
False	True	False	False	True	False	T	T
False	True	False	True	True	False	T	T

Karena KB |= α_1, maka kamar [2,1] aman

Menyatakan kamar di [2,2] tidak aman

$B_{1,1}$	$B_{1,2}$	$P_{1,2}$	$P_{2,2}$	$P_{3,1}$	$P_{2,1}$	KB	α_2
False	False	False	False	False	False	F	T
False	False	True	False	False	False	F	T
False	True	False	False	False	False	F	T
False	True	True	False	False	False	F	T
False	True	False	False	True	False	T	T
False	False	True	False	True	False	F	T

Karena KB|≠ α_1, maka kamar [2,2] tidak aman

4. Implementasi agen berbasis logika dengan contoh Wumpus World

Source code:

Environment.java

```
1   package praktikumkb5;
2   import java.util.Scanner;
3   public class Environment {
4       Scanner scr = new Scanner(System.in);
5       int np;
6       int wp, gp;
7       int pos[];
8       int b_pos[] = new int[20];
9       int s_pos[] = new int[20];
10      String w[][];
11      int sp;
12      public Environment(int row, int column) {
13          w = new String[5][5];
14          sp = 13;
15          for (int i = 0; i < 20; ++i) {
16              b_pos[i] = -1;
17              s_pos[i] = -1;
18          }
19          for (int i = 0; i < 5; ++i) {
20              for (int j = 0; j < 5; ++j) {
21                  w[i][j] = "";
22              }
23          }
24      }
25      public void createProblem() {
26          display(0);
27          System.out.println("\nAgent start position from "
28        + sp);
29          w[4][1] = "A";
30
31          System.out.print("Enter the number of pits: ");
32          np = scr.nextInt();
33          pos = new int[np];
34
35          System.out.println("Note: positions of pit, gold
36        and wumpus should not overlap.");
37          System.out.print("Enter the position of pits: ");
38          for (int i = 0; i < np; ++i) {
39              pos[i] = scr.nextInt();
40              showSense(pos[i], 1);
41          }
42
43          System.out.print("Enter the position of wumpus:
44        ");
45          wp = scr.nextInt();
46          showSense(wp, 2);
47
48          System.out.print("Enter the position of gold: ");
49          gp = scr.nextInt();
50          System.out.println();
51          insert();
```

```
52      }
53      public void insert() {
54          int temp = 0;
55          int count = 0;
56          int flag1 = 0, flag2 = 0;
57          for (int i = 0; i < np; ++i) {
58              temp = pos[i];
59              count = 0;
60              for (int j = 1; j <= 4; ++j) {
61                  for (int k = 1; k <= 4; ++k) {
62                      ++count;
63                      if (count == temp) {
64                          w[j][k] += "P";
65                      } else {
66                          if (count == gp && flag1 == 0) {
67                              w[j][k] += "G";
68                              flag1 = 1;
69                          } else {
70                              if (count == wp && flag2 ==
71        0) {
72                                  w[j][k] += "W";
73                                  flag2 = 1;
74                              }
75                          }
76                      }
77                  }
78              }
79          }
80          display(1);
81      }
82      public void showSense(int a, int b) {
83          int left, right, bottom, up;
84          left = a - 1;
85          right = a + 1;
86          bottom = a + 4;
87          up = a - 4;
88          if (a == 5 || a == 9) {
89              left = 0;
90          }
91          if (a == 8 || a == 12) {
92              right = 0;
93          }
94          if (a == 4) {
95              right = 0;
96          }
97          if (a == 13) {
98              left = 0;
99          }
100         if (bottom > 16) {
101             bottom = 0;
102         }
103         if (bottom < 0) {
104             up = 0;
105         }
106         if (b == 1) {
107             b_pos[0] = left;
108             b_pos[1] = right;
109             b_pos[2] = bottom;
110             b_pos[3] = up;
```

```
111         } else {
112             if (b == 2) {
113                 s_pos[0] = left;
114                 s_pos[1] = right;
115                 s_pos[2] = bottom;
116                 s_pos[3] = up;
117             }
118         }
119         int temp1, count;
120         for (int i = 0; i < 4; ++i) {
121             if (b == 1) {
122                 temp1 = b_pos[i];
123             } else {
124                 temp1 = s_pos[i];
125             }
126             count = 0;
127             for (int j = 1; j <= 4; ++j) {
128                 for (int k = 1; k <= 4; ++k) {
129                     ++count;
130                     if (count == temp1 && b == 1
131       && !w[j][k].contains("B")) {
132                         w[j][k] += "B";
133                     } else {
134                         if (count == temp1 && b == 2
135       && !w[j][k].contains("S")) {
136                             w[j][k] += "S";
137                         }
138                     }
139                 }
140             }
141         }
142     }
143     public void display(int status) {
144         int count = 1;
145         if (status == 0) {
146             System.out.println("The positions are as
147       follows.");
148         } else {
149             System.out.println("The environment for
150       problem is as follows.");
151         }
152         System.out.println(" ----------------------------
153       ---------------------------------- ");
154         for (int i = 1; i <= 4; ++i) {
155             System.out.print("|\t");
156             for (int j = 1; j <= 4; ++j) {
157                 if (status == 0) {
158                     System.out.print((count++) +
159       "\t|\t");
160                 } else {
161                     System.out.print(w[i][j] + "\t|\t");
162                 }
163             }
164             System.out.println();
165             if (i < 4) {
166                 for (int x = 0; x <= 64; x++) {
167                     if (x == 0 || x == 16 || x == 32 || x
168       == 48 || x == 64) {
169                         System.out.print("|");
```

```
170                         } else {
171                             System.out.print("-");
172                         }
173                     }
174                     System.out.println();
175                 }
176             }
177             System.out.println(" ----------------------------
178           ---------------------------------- ");
179         }
180 }
```

Tiles.java

```
1   package praktikumkb5;
2   public class Tiles {
3       int safe = 0;
4       int unsafe = 0;
5       int wump = 0;
6       int pit = 0;
7       int gold = 0;
8       int doubt_pit = 0;
9       int doubt_wump = 0;
10      String env;
11      int num = 0;
12      int br = 0;
13      int bl = 0;
14      int bu = 0;
15      int bd = 0;
16      int visited = 0;
17      int l, r, u, d;
18      String back = "";
19      public Tiles(String s, int n) {
20          env = s;
21          num = n;
22          l = r = u = d = 0;
23          if (n == 9 || n == 5) {
24              bl = 1;
25          }
26          if (n == 8 || n == 12) {
27              br = 1;
28          }
29          if (n == 1) {
30              bu = 1;
31              bl = 1;
32          }
33          if (n == 13) {
34              bd = 1;
35              bl = 1;
36          }
37          if (n == 4) {
38              bu = 1;
39              br = 1;
40          }
41          if (n == 16) {
42              bd = 1;
43              br = 1;
44          }
```

```
45      }
46      public int sense() {
47          if (env.contains("B")) {
48              return 1;
49          } else {
50              if (env.contains("S")) {
51                  return 2;
52              } else {
53                  if (env.contains("G")) {
54                      return 3;
55                  }
56              }
57          }
58          if (env.contains("W")) {
59              return 4;
60          } else {
61              return 0;
62          }
63      }
64  }
```

Wumpus.java

```
1   package praktikumkb5;
2   public class Wumpus {
3       static int scream = 0;
4       static int score = 0;
5       static int complete = 0;
6       Environment e;
7       public Wumpus(Environment e) {
8           this.e = e;
9       }
10      public boolean check(Tiles t) {
11          int temp = t.sense();
12          return !(temp == 1 || temp == 2);
13      }
14      public void findSolution() {
15          Tiles t[] = new Tiles[17];
16          int c = 1;
17          out:
18          for (int i = 1; i < 5; ++i) {
19              for (int j = 1; j < 5; ++j) {
20                  if (c > 16) {
21                      break out;
22                  }
23                  t[c] = new Tiles(e.w[i][j], c);
24                  ++c;
25              }
26          }
27          t[13].safe = 1;
28          t[13].visited = 1;
29          int pos = 13;
30          int condition;
31          int limit = 0;
32          String temp1, temp2;
33          System.out.println("\nStart from position\t: " +
34        e.sp);
35          do {
```

```
36              ++limit;
37              if (t[pos].env.contains("G")) {
38                  complete = 1;
39                  System.out.println("Finish, Gold
40          Found!!\n");
41                  break;
42              }
43              if (t[pos].br != 1 && t[pos].r != 1 && t[pos
44          + 1].doubt_pit < 1 && t[pos + 1].doubt_wump < 1 &&
45          t[pos + 1].pit != 1 && t[pos + 1].wump != 1
46          && !(t[pos].back.contains("r") && (t[pos].l != 1 ||
47          t[pos].u != 1 || t[pos].d != 1) && check(t[pos])))
48          {
49                  temp1 = "l";
50                  t[pos].r = 1;
51                  ++pos;
52                  System.out.println("→ Move right,
53          position\t: " + pos);
54                  ++score;
55                  t[pos].back += temp1;
56                  condition = t[pos].sense();
57                  if (condition == 3) {
58                      complete = 1;
59                      break;
60                  } else {
61                      if (condition == 1 && t[pos].visited
62          == 0) {
63                          if (t[pos].br != 1 && t[pos +
64          1].safe != 1) {
65                              t[pos + 1].doubt_pit += 1;
66                          }
67                          if (t[pos].bu != 1 && (pos -
68          4) >= 1 && t[pos - 4].safe != 1) {
69                              t[pos - 4].doubt_pit += 1;
70                          }
71                          if (t[pos].bl != 1 && t[pos -
72          1].safe != 1) {
73                              t[pos - 1].doubt_pit += 1;
74                          }
75                          if (t[pos].bd != 1 && (pos + 4)
76          <= 16 && t[pos + 4].safe != 1) {
77                              t[pos + 4].doubt_pit += 1;
78                          }
79                          t[pos].safe = 1;
80                      } else {
81                          if (condition == 2 &&
82          t[pos].visited == 0) {
83                              if (t[pos].br != 1 && t[pos +
84          1].safe != 1) {
85                                  t[pos + 1].doubt_wump +=
86          1;
87                              }
88                              if (t[pos].bu != 1 && (pos -
89          4) >= 1 && t[pos - 4].safe != 1) {
90                                  t[pos - 4].doubt_wump +=
91          1;
92                              }
93                              if (t[pos].bl != 1 && t[pos -
94          1].safe != 1) {
```

```
                                    t[pos - 1].doubt_wump +=
1;
                                }
                                if (t[pos].bd != 1 && (pos +
4) <= 16 && t[pos + 4].safe != 1) {
                                    t[pos + 4].doubt_wump +=
1;
                                }
                                t[pos].safe = 1;
                            } else {
                                if (condition == 0) {
                                    t[pos].safe = 1;
                                }
                            }
                        }
                    }
                    t[pos].visited = 1;
                } else {
                    if (t[pos].bl != 1 && t[pos].l != 1 &&
t[pos - 1].doubt_pit < 1 && t[pos - 1].doubt_wump <
1 && t[pos - 1].pit != 1 && t[pos - 1].wump != 1
&& !(t[pos].back.contains("1") && (t[pos].r != 1 ||
t[pos].u != 1 || t[pos].d != 1) && check(t[pos])))
{
                        temp1 = "r";
                        t[pos].l = 1;
                        pos = pos - 1;
                        System.out.println("← Move left,
position\t: " + pos);
                        ++score;
                        t[pos].back += temp1;
                        condition = t[pos].sense();
                        if (condition == 3) {
                            complete = 1;
                            break;
                        } else {
                            if (condition == 1 &&
t[pos].visited == 0) {
                                if (t[pos].br != 1 && t[pos +
1].safe != 1) {
                                    t[pos + 1].doubt_pit +=
1;
                                }
                                if (t[pos].bu != 1 && (pos -
4) >= 1 && t[pos - 4].safe != 1) {
                                    t[pos - 4].doubt_pit +=
1;
                                }
                                if (t[pos].bl != 1 && t[pos -
1].safe != 1) {
                                    t[pos - 1].doubt_pit +=
1;
                                }
                                if (t[pos].bd != 1 && (pos +
4) <= 16 && t[pos + 4].safe != 1) {
                                    t[pos + 4].doubt_pit +=
1;
                                }
                                t[pos].safe = 1;
```

```
154                         } else {
155                             if (condition == 2 &&
156     t[pos].visited == 0) {
157                                 if (t[pos].br != 1 &&
158     t[pos + 1].safe != 1) {
159                                     t[pos + 1].doubt_wump
160     += 1;
161                                 }
162                                 if (t[pos].bu != 1 &&
163     (pos - 4) >= 1 && t[pos - 4].safe != 1) {
164                                     t[pos - 4].doubt_wump
165     += 1;
166                                 }
167                                 if (t[pos].bl != 1 &&
168     t[pos - 1].safe != 1) {
169                                     t[pos - 1].doubt_wump
170     += 1;
171                                 }
172                                 if (t[pos].bd != 1 &&
173     (pos + 4) <= 16 && t[pos + 4].safe != 1) {
174                                     t[pos + 4].doubt_wump
175     += 1;
176                                 }
177                                 t[pos].safe = 1;
178                             } else {
179                                 if (condition == 0) {
180                                     t[pos].safe = 1;
181                                 }
182                             }
183                         }
184                     }
185                     t[pos].visited = 1;
186                 } else {
187                     if (t[pos].bu != 1 && t[pos].u != 1
188     && (pos - 4) >= 1 && t[pos - 4].doubt_pit < 1 &&
189     t[pos - 4].doubt_wump < 1 && t[pos - 4].pit != 1 &&
190     t[pos - 1].wump != 1 && !(t[pos].back.contains("u")
191     && (t[pos].l != 1 || t[pos].r != 1 || t[pos].d !=
192     1) && check(t[pos]))) {
193                         temp1 = "d";
194                         t[pos].u = 1;
195                         pos = pos - 4;
196                         System.out.println("↑ Move up,
197     position\t: " + pos);
198                         ++score;
199                         t[pos].back += temp1;
200                         condition = t[pos].sense();
201                         if (condition == 3) {
202                             complete = 1;
203                             break;
204                         } else {
205                             if (condition == 1 &&
206     t[pos].visited == 0) {
207                                 if (t[pos].br != 1 &&
208     t[pos + 1].safe != 1) {
209                                     t[pos + 1].doubt_pit
210     += 1;
211                                 }
212                                 if (t[pos].bu != 1 &&
```

```
213         (pos - 4) >= 1 && t[pos - 4].safe != 1) {
214                                     t[pos - 4].doubt_pit
215         += 1;
216                                 }
217                                 if (t[pos].bl != 1 &&
218         t[pos - 1].safe != 1) {
219                                     t[pos - 1].doubt_pit
220         += 1;
221                                 }
222                                 if (t[pos].bd != 1 &&
223         (pos + 4) <= 16 && t[pos + 4].safe != 1) {
224                                     t[pos + 4].doubt_pit
225         += 1;
226                                 }
227                                 t[pos].safe = 1;
228                             } else {
229                                 if (condition == 2 &&
230         t[pos].visited == 0) {
231                                     if (t[pos].br != 1 &&
232         t[pos + 1].safe != 1) {
233                                         t[pos +
234         1].doubt_wump += 1;
235                                     }
236                                     if (t[pos].bu != 1 &&
237         (pos - 4) >= 1 && t[pos - 4].safe != 1) {
238                                         t[pos -
239         4].doubt_wump += 1;
240                                     }
241                                     if (t[pos].bl != 1 &&
242         t[pos - 1].safe != 1) {
243                                         t[pos -
244         1].doubt_wump += 1;
245                                     }
246                                     if (t[pos].bd != 1 &&
247         (pos + 4) <= 16 && t[pos + 4].safe != 1) {
248                                         t[pos +
249         4].doubt_wump += 1;
250                                     }
251                                     t[pos].safe = 1;
252                                 } else {
253                                     if (condition == 0) {
254                                         t[pos].safe = 1;
255                                     }
256                                 }
257                             }
258                         }
259                         t[pos].visited = 1;
260                     } else {
261                         if (t[pos].bd != 1 && t[pos].d !=
262         1 && (pos + 4) <= 16 && t[pos + 4].doubt_pit < 1 &&
263         t[pos + 4].doubt_wump < 1 && t[pos + 4].pit != 1 &&
264         t[pos + 4].wump != 1) {
265                             temp1 = "u";
266
267                             t[pos].d = 1;
268                             pos = pos + 4;
269                             System.out.println("↓ Move
270         down, position\t: " + pos);
271                             ++score;
```

```
272
273                             t[pos].back += temp1;
274                             condition = t[pos].sense();
275                             if (condition == 3) {
276                                 complete = 1;
277                                 break;
278                             } else {
279                                 if (condition == 1 &&
280       t[pos].visited == 0) {
281                                     if (t[pos].br != 1 &&
282       t[pos + 1].safe != 1) {
283                                         t[pos +
284       1].doubt_pit += 1;
285                                     }
286                                     if (t[pos].bu != 1 &&
287       (pos - 4) >= 1 && t[pos - 4].safe != 1) {
288                                         t[pos -
289       4].doubt_pit += 1;
290                                     }
291                                     if (t[pos].bl != 1 &&
292       t[pos - 1].safe != 1) {
293                                         t[pos -
294       1].doubt_pit += 1;
295                                     }
296                                     if (t[pos].bd != 1 &&
297       (pos + 4) <= 16 && t[pos + 4].safe != 1) {
298                                         t[pos +
299       4].doubt_pit += 1;
300                                     }
301                                     t[pos].safe = 1;
302                                 } else {
303                                     if (condition == 2 &&
304       t[pos].visited == 0) {
305                                         if (t[pos].br !=
306       1 && t[pos + 1].safe != 1) {
307                                             t[pos +
308       1].doubt_wump += 1;
309                                         }
310                                         if (t[pos].bu !=
311       1 && (pos - 4) >= 1 && t[pos - 4].safe != 1) {
312                                             t[pos -
313       4].doubt_wump += 1;
314                                         }
315                                         if (t[pos].bl !=
316       1 && t[pos - 1].safe != 1) {
317                                             t[pos -
318       1].doubt_wump += 1;
319                                         }
320                                         if (t[pos].bd !=
321       1 && (pos + 4) <= 16 && t[pos + 4].safe != 1) {
322                                             t[pos +
323       4].doubt_wump += 1;
324                                         }
325                                         t[pos].safe = 1;
326                                     } else {
327                                         if (condition ==
328       0) {
329                                             t[pos].safe =
330       1;
```

```
                                        }
                                    }
                                }
                            }
                            t[pos].visited = 1;
                        } else {
                            if (limit > 50) {
                                int temp3 = pos;
                                int flag_1 = 0, flag2 =
    0, flag3 = 0, flag4 = 0;

    System.out.println("\nCurrently at position " +
    temp3 + ".\nThinking....");
                                while (t[pos].visited ==
    1 && t[pos].br != 1) {
                                    ++pos;
                                    ++score;
                                }
                                if (t[pos].pit == 1 ||
    t[pos].wump == 1 || (t[pos].br == 1 &&
    t[pos].visited == 1 && t[pos].safe != 1)) {
                                    pos = temp3;
                                    flag_1 = 1;
                                }
                                if (flag_1 == 0) {
                                    t[pos].back += "l";
                                }
                                while (pos + 4 >= 1 &&
    t[pos].bu != 1 && t[pos].visited == 1) {
                                    pos -= 4;
                                    ++score;
                                }
                                if (t[pos].pit == 1 ||
    t[pos].wump == 1 || (t[pos].bu == 1 &&
    t[pos].visited == 1 && t[pos].safe != 1)) {
                                    pos = temp3;
                                    flag3 = 1;
                                }
                                if (flag3 == 0) {
                                    t[pos].back += "d";
                                }
                                while (t[pos].visited ==
    1 && t[pos].bl != 1) {
                                    --pos;
                                    ++score;
                                }
                                if (t[pos].pit == 1 ||
    t[pos].wump == 1 || (t[pos].bl == 1 &&
    t[pos].visited == 1 && t[pos].safe != 1)) {
                                    pos = temp3;
                                    flag2 = 1;
                                }
                                if (flag2 == 0) {
                                    t[pos].back += "r";
                                }
                                while (pos + 4 <= 16 &&
    t[pos].bd != 1 && t[pos].visited == 1) {
                                    pos += 4;
                                    ++score;
```

```
390                                     }
391                                     if (t[pos].pit == 1 ||
392         t[pos].wump == 1 || (t[pos].bd == 1 &&
393         t[pos].visited == 1 && t[pos].safe != 1)) {
394                                         pos = temp3;
395                                         flag4 = 1;
396                                     }
397                                     if (flag4 == 0) {
398                                         t[pos].back += "u";
399                                     }
400                                     t[pos].safe = 1;
401                                     t[pos].visited = 1;
402 
403         System.out.println("reached at position " + pos);
404                                     limit = 0;
405                                 }
406                             }
407                         }
408                     }
409                 }
410                 if (t[pos].env.contains("W") && scream != 1)
411         {
412                     score += 100;
413                     scream = 1;
414                     t[pos].safe = 1;
415                     System.out.println("\n\nWumpus killed >--
416         0-->");
417                     t[pos].env.replace("W", " ");
418                     for (int l = 1; l <= 16; ++l) {
419                         t[l].doubt_wump = 0;
420                         t[l].env.replace("S", " ");
421                     }
422                 }
423                 if (t[pos].env.contains("P")) {
424                     score += 50;
425                     t[pos].pit = 1;
426                     System.out.println("\n\nFallen in pit of
427         position " + pos + ".");
428                 }
429                 for (int k = 1; k <= 16; ++k) {
430                     if (t[k].doubt_pit == 1 &&
431         t[k].doubt_wump == 1) {
432                         t[k].doubt_pit = 0;
433                         t[k].doubt_wump = 0;
434                         t[k].safe = 1;
435                     }
436                 }
437                 for (int y = 1; y <= 16; ++y) {
438                     if (t[y].doubt_wump > 1) {
439                         t[y].wump = 1;
440                         for (int h = 1; h <= 16; ++h) {
441                             if (h != y) {
442                                 t[h].doubt_wump = 0;
443                                 t[h].env.replace("S", " ");
444                             }
445                         }
446                     }
447                 }
448                 for (int y = 1; y <= 16; ++y) {
```

```
449                     if (t[y].doubt_pit > 1) {
450                         t[y].pit = 1;
451                     }
452                 }
453                 try {
454                     Thread.sleep(100);
455                 } catch (Exception p) {
456                 }
457 
458             } while (complete == 0);
459             if (complete == 1) {
460                 score *= -1;
461                 score += 1000;
462             }
463             System.out.println("The score of the agent till
464           he reaches gold is "
465                     + score + ".\nNow he will return back
466           following the best explored path.");
467             System.out.println("");
468         }
469 }
```

Main.java

```
1  package praktikumkb5;
2  public class Main {
3      public static void main(String[] args) {
4          Environment e = new Environment(5, 5);
5          e.createProblem();
6          Wumpus w = new Wumpus(e);
7          w.findSolution();
8      }
9  }
```

Penjelasan

Environment.java

- Baris 12-24: constructor Environment() digunakan untuk menginisialisasi field mulai dari kotak masing-masing Wumpus World sampai posisi awal dari agent
- Baris 25-52: method createProblem() digunakan untuk menampilkan kondisi awal Wumpus World sebelum dimasukkan masalahnya kemudian memasukkan semua kondisi Wumpus World, mulai dari jumlah PIT, posisi PIT, posisi GOLD, dan posisi WUMPUS dalam Wumpus World
- Baris 53-81: method insert() digunakan untuk memasukkan kondisi PIT, GOLD, WUMPUS ke tile yang ada pada Wumpus World. Dan menampilkan hasil Wumpus World setelah terisi kondisi-kondisinya
- Baris 82-142: method showSense() digunakan untuk meletakan posisi Breeze dan Smell disekeliling posisi PIT dan WUMPUS. Kemudian posisi yang sudah ditentukan akan diberi nilai B untuk Breeze dan S untuk Smell
- Baris 143-179: method display() digunakan untuk menampilkan kondisi Wumpus World sebelum diisi kondisi permasalahan dan sesudah diisi permasalahan

Tiles.java

- Baris 19-45: constructor Tiles() digunakan untuk menginisialisasi semua field yang ada pada class Tiles. Penginisialisasian ini juga berdasarkan metode pengecekan pada kondisi tiles
- Baris 46-63: method sense() digunakan untuk merasakan apakah dalam suatu tile terdapat kondisi Breeze, Smell, Gold dan Wumpus dan akan mengembalikan nilai masing-masing sesuai kondisi

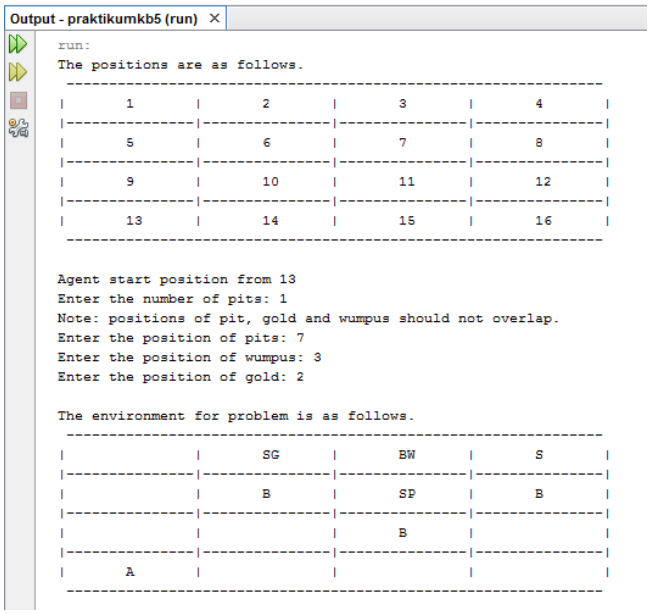
Wumpus.java

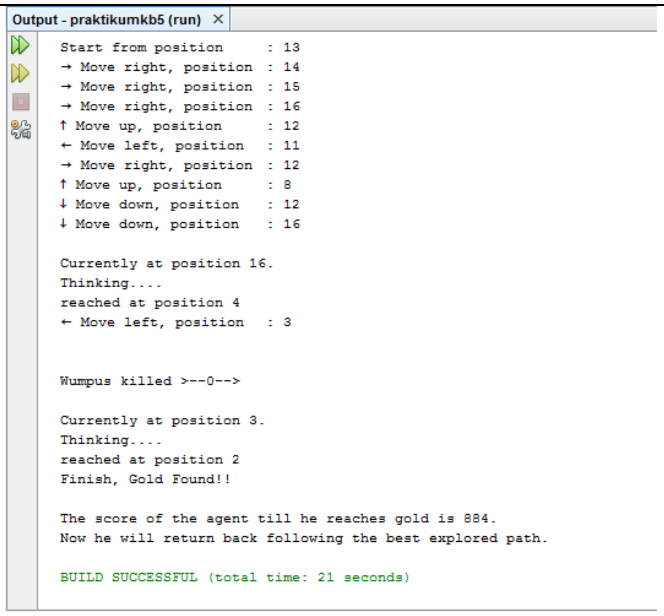
- Baris 7-9: constructor Wumpus() digunakan untuk menginisialisasi lingkungan Wumpus World yang akan dicari solusinya
- Baris 10-13: method check() digunakan untuk mengecek apakah dalam suatu kotak (tile) tedapat gold atau wumpus. Bernilai benar jika tidak ada keduanya
- Baris 14-468: method findSolution() digunakan untuk menerapkan algoritma agent logika di mana dari lingkungan Wumpus World yang ada dicari solusinya dengan menjalankan metode-metode preposisi. Dari lingkungan yang ada kemudian akan menggunakan inferensi untuk menentukan langkah selanjutnya. Langkah inferensi ini menggunakan entailment untuk mempertimbangkan langkah yang dipilih. Di method ini juga akan mencetak solusi sekaligus ketika dioperasikan dan juga skor yang diperoleh

Main.java

Method main() digunakan untuk membuat objek lingkungan dan membuat masalah wumpus world kemudian akan dicari solusinya.

Screenshot:




Output - praktikumkb5 (run)
Start from position : 13
→ Move right, position : 14
→ Move right, position : 15
→ Move right, position : 16
↑ Move up, position : 12
← Move left, position : 11
→ Move right, position : 12
↑ Move up, position : 8
↓ Move down, position : 12
↓ Move down, position : 16
Currently at position 16.
Thinking....
reached at position 4
← Move left, position : 3
Wumpus killed >--0-->
Currently at position 3.
Thinking....
reached at position 2
Finish, Gold Found!!
The score of the agent till he reaches gold is 884.
Now he will return back following the best explored path.
BUILD SUCCESSFUL (total time: 21 seconds)

C. KESIMPULAN

1. Sebutkan perbedaan definisi dan konsep dasar agen berbasis logika dengan agen berbasis pengetahuan!
   - Definisi
     - Agen logika merupakan agen yang memiliki kemampuan bernalar secara logis. Ketika beberapa solusi tidak diketahui secara eksplisit, maka diperlukan suatu agen berbasis logika.
     - Agen berbasis pengetahuan menyatakan tentang apa yang diketahui oleh agent. Agent dapat dipandang dari knowledge level (informasi apa saja yang diketahui)
   - Konsep dasar
     - Problem solving agent: memilih solusi di antara kemungkinan yang ada. Apa yang ia "ketahui" tentang dunia, pengetahuannya tidak berkembang untuk mencapai problem solution (initial state, successor function, goal test)
     - Knowledge-based agent : lebih "pintar". Ia "mengetahui" hal-hal tentang dunia dan dapat melakukan reasoning (berpikir, bernalar) mengenai hal-hal yang tidak diketahui sebelumnya (imperfect/partial information) dan tindakan yang paling baik untuk diambil (best action)
2. Jelaskan konsep dasar logika proporsisi baik secara sintaks atau maupun semantic!
   - Sintaks: aturan yang diperlukan untuk mengkombinasikan antara propositions dan propositional connectives dalam menghasilkan kalimat logika
   - Semantic: aturan yang digunakan untuk menentukan "truth value" dari suatu kalimat
3. Jelaskan metode pembuktian dari logika proporsisi
   Ada 3 cara yang bisa dilakukan dalam pembuktian logika:
   a. Menggunakan Tabel Kebenaran
      Pembuktian dengan menggunakan Tabel Kebenaran adalah metode pembuktian yang paling mudah. Teknis pelaksanaan pembuktian, pertama tentukan formula dari premis yang ada, kemudian baru tentukan metode apa yang hendak digunakan dalam pembuktian misalnya pembuktian langsung, setelah itu baru dibuat tabel kebenarannya.
   b. Menggunakan Normalisasi
      Pembuktian dengan Normalisasi adalah menyambungkan premis-premis dengan mengunakan operator logika kemudian disederhanakan lagi menggunakan bentuk normal konjungsi atau disjungsi. Teknis pelaksanaan pembuktian, lakukan penyederhanaan formula dengan melakukan penggantian(substitusi) terhadap suatu formula dengan formula lainnya. Manfaatkan Tabel Ekuivalensi Logis.
   c. Menggunakan Aturan Inferensi
      Pembuktian dengan Aturan Inferensi adalah pembuktian berdasarkan asumsi bahwa suatu himpunan proposisi terdiri dari sejumlah premis (P) dan kesimpulan (C). Selajutnya pembuktian kesimpulan berdasarkan premis tersebut dirangkai menjadi sebuah tree/pohon. Manfaatkan Tabel Aturan Inferensi.