Bhargavi Poyekar: CH33454

Homework 2: Hunt the Wampa

Part 1: Implement Agent

1. Knowledge Base Function:

   I've employed two functions: KnowledgeBaseQuery and KnowledgeBaseStore. When I use KnowledgeBaseQuery, I retrieve the specific rule I require from my self.KB. On the other hand, with KnowledgeBaseStore, I'm responsible for incorporating the freshly generated rules stemming from the inference algorithm into the knowledge base.

```
def knowledgeBaseQuery(self, rule):
  #### YOUR CODE HERE ####
  # This function is for picking out relevant information to do deductions/inferences
  return self.KB[rule]


def knowledgeBaseStore(self, rules):
  #### YOUR CODE HERE ####
  # This function is for storing new inferences in your KB

    # Storing the New Infered Rule
    self.KB[rules[0]]=rules[1]

    # If the rule is of a single cell, put True and False for correct rule and it's inverse.
    if rules[2]==1 and '~' not in rules[0]:
      self.KB['~'+rules[0]]=not rules[1]
      # print('~'+rules[0],':', self.KB['~'+rules[0]])
    elif rules[2]==1 and '~' in rules[0]:
      self.KB[rules[0][1:]]=not rules[1]
      # print(rules[0][1:],':', self.KB[rules[0][1:]])
```

   In addition to these functions, my Initialize KB plays a crucial role in setting up the initial Wampa rules within the knowledge base. My KB is structured as a dictionary, where each rule is defined as either True or False. For example, if there's no Wampa in a cell, then ~W11 is marked as True, and W11 is marked as False. When I refer to the initial Wampa rules, I mean the rules that establish a connection between the presence of Stench and the existence of a Wampa in an adjacent cell, as well as the absence of Stench and the absence of a Wampa in an adjacent cell. These initial rules were too extensive to input manually, so I wrote a code to automate the initialization process. Subsequently, I applied the double implication inference rule to convert all ⇔ to => for greater clarity. As a final step, I ensured that the initial KB, including these rules, was printed out in the output for all scenarios. I've attached a screenshot of one of the initial KBs for reference.

```
def initializeKB(self,w):
  # Initialize the Knowledge Base

  #Grid width and height
  grid_Row=len(w.grid)
  grid_Col=len(w.grid[0])

  # For every cell, updating basic wampa rules
  for i in range(1,grid_Row+1):
    for j in range(1,grid_Col+1):
      # Current cell
      location_str=str(i)+str(j)
      basic_rules=[]

      x,y=i,j

      # Stench <=> Wampa in Neighboring cell
      stenchlogic1='S'+location_str+' <=> '

      # No Stench <=> No Wampa in Neighboring cell
      stenchlogic2='~S'+location_str+' <=> '

      # Breeze <=> Pit in Neighboring cell
      breezelogic1='B'+location_str+' <=> '

      # No Breeze <=> No Pit in Neighboring cell
      breezelogic2='~B'+location_str+' <=> '
```

```
      for dx,dy in [(0,-1),(0,1),(1,0),(-1,0)]:
        new_x,new_y=x+dx,y+dy
        if w.agent.valid_cell(new_x,new_y,grid_Row,grid_Col):
          nghbr=str(new_x)+str(new_y)
          stenchlogic1+='W'+ nghbr+' or '
          stenchlogic2+='~W'+ nghbr+' and '
          breezelogic1+='P'+ nghbr+' or '
          breezelogic2+='~P'+ nghbr+' and '

      # Removing last 'or' and 'and'
      stenchlogic1=stenchlogic1[:-4]
      stenchlogic2=stenchlogic2[:-5]
      breezelogic1=breezelogic1[:-4]
      breezelogic2=breezelogic2[:-5]

      basic_rules.extend([stenchlogic1,stenchlogic2,breezelogic1,breezelogic2])

      # Update every rule to knowledge base
      for rule in basic_rules:
        w.agent.knowledgeBaseStore([rule,True,0])

        # Perform double elimination inference rule
        w.agent.double_elim(rule)

  print("Initialize Knowledge Base")
  for i in self.KB:
    print(i)

  # return the knowledge Base
  return self.KB
```

```
Initialize Knowledge Base
S11 <=> W12 or W21
S11 => W12 or W21
W12 or W21 => S11
~S11 <=> ~W12 and ~W21
~S11 => ~W12 and ~W21
~W12 and ~W21 => ~S11
B11 <=> P12 or P21
B11 => P12 or P21
P12 or P21 => B11
~B11 <=> ~P12 and ~P21
~B11 => ~P12 and ~P21
~P12 and ~P21 => ~B11
S12 <=> W11 or W13 or W22
S12 => W11 or W13 or W22
W11 or W13 or W22 => S12
~S12 <=> ~W11 and ~W13 and ~W22
~S12 => ~W11 and ~W13 and ~W22
~W11 and ~W13 and ~W22 => ~S12
B12 <=> P11 or P13 or P22
B12 => P11 or P13 or P22
P11 or P13 or P22 => B12
~B12 <=> ~P11 and ~P13 and ~P22
~B12 => ~P11 and ~P13 and ~P22
~P11 and ~P13 and ~P22 => ~B12
S13 <=> W12 or W14 or W23
S13 => W12 or W14 or W23
W12 or W14 or W23 => S13
~S13 <=> ~W12 and ~W14 and ~W23
~S13 => ~W12 and ~W14 and ~W23
~W12 and ~W14 and ~W23 => ~S13
B13 <=> P12 or P14 or P23
B13 => P12 or P14 or P23
P12 or P14 or P23 => B13
~B13 <=> ~P12 and ~P14 and ~P23
~B13 => ~P12 and ~P14 and ~P23
~P12 and ~P14 and ~P23 => ~B13
S14 <=> W13 or W24
S14 => W13 or W24
W13 or W24 => S14
```

2. Inference Algorithm Function:

```python
# Double Elimination
def double_elim(self,rule):
  # print(f'Rule: {rule}')
  new_rules=[]
  if '<=>' in rule:
    # print(f'\nRule: {rule}')
    parts=rule.split(' <=> ')
    # print("By Double Elimination: ")
    # print(parts[0]+ ' => '+ parts[1])
    # print(parts[1]+ ' => '+ parts[0])

    w.agent.knowledgeBaseStore([parts[0]+ ' => '+ parts[1],True,0])
    w.agent.knowledgeBaseStore([parts[1]+ ' => '+ parts[0],True,0])


# Modus Ponens
def modus_ponens(self, alpha, beta, w):
  if (alpha+' => '+beta) in self.KB and self.KB[alpha+' => '+beta] and alpha in self.KB and self.KB[alpha]:
    print(f'\nRule: {alpha} => {beta}, {alpha}')
    print("By Modus Ponens: ")
    print(beta)
    w.agent.knowledgeBaseStore([beta,True,0])

# And Introduction
def and_intro(self, alpha, beta, w):
  if alpha in self.KB and self.KB[alpha] and beta in self.KB and self.KB[beta]:
    rule=alpha+' and '+beta
    w.agent.knowledgeBaseStore([rule,True,0])
```

```python
# And Elimination
def and_elimitation(self, rule, w):
  if rule in self.KB and self.KB[rule]:
    print(f'\nRule: {rule}')
    rule_split=rule.split(' and ')
    print("By And Elimination: ")
    for r in rule_split:
      print(r)
      w.agent.knowledgeBaseStore([r,True,1])

# Resolution
def unit_resol(self, rule, w):
  remove=[]

  if rule in self.KB and self.KB[rule]:
    print(f'\nRule: {rule}')
    for r in rule.split(' or '):
      if r in self.KB and not self.KB[r]:
        print('~'+r)
        remove.append(r)

  final_rule=''
  for r in rule.split(' or '):
    if r not in remove:
      final_rule+= r+ ' or '

  final_rule=final_rule[:-4]
  print("By Resolution: ")
  print(final_rule)
  if 'or' not in final_rule:
    w.agent.knowledgeBaseStore([final_rule,True,1])
  else:
    w.agent.knowledgeBaseStore([final_rule,True,0])
```

In the Inference Algorithm, my initial step involves incorporating two key rules: one indicating the absence of a Wampa in the current location (~W11: True) and the other specifying the absence of a Pit in the current location (~P11: True). To optimize the process, I also included corresponding rules denoting the presence of Wampa (W11: False) and Pit (P11: False). This practice of including both the negation and the standard rule proved to be immensely helpful, particularly in handling rules stored as strings, which would have otherwise posed challenges in managing negations.

I started by creating logic strings based on the agent's current location to check neighboring cells in my Knowledge Base, similar to how I set up the initial rules.

When I observed the presence of stench, I marked S(xy) as True, which triggered the implication to become True as well through modus ponens. This led me to use the modus ponens function.

Once the implication was confirmed as True, I applied unit resolution to see if I could simplify the rule and draw meaningful conclusions for any of the nearby cells. I followed the same approach for Breeze and Pit in this inferencing process.

If there's no stench, it means there are no Wampas in neighboring cells. So, I apply modus ponens and And elimination to derive rules like ~W12: true and ~W21: true, assuming these are neighboring cells.

Additionally, I check whether the current location's W11 or its negation (~W11) has any impact on previous rules. I use 'and' or 'or' functions and then apply resolution and And Elimination for this purpose.

Ultimately, I categorize all the rules for neighboring cells into three groups: safe (where we're confident there are no Wampas or Pits), unsafe (where we've inferred the presence of a Pit or Wampa), and unknown (where the presence of a Pit, Wampa, or nothing is uncertain, as it's not fully inferred yet). As a result, my inference algorithm generates three lists, each containing neighbors grouped into these categories.

3. Successful Paths and Output:

Scenario W1:

Current Direction : right
Current Location: [1, 1]
Current Percepts: ['stench', None, None, None, None]

Rule: S11 => W12 or W21, S11
By Modus Ponens:
W12 or W21

Rule: W12 or W21
By Resolution:
W12 or W21

Rule: ~B11 => ~P12 and ~P21, ~B11
By Modus Ponens:
~P12 and ~P21

Rule: ~P12 and ~P21
By And Elimination:
~P12, ~P21

Safe Neighbours: []
Unknown Neighbours: ['12', '21']
Unsafe Neighbours: []

Action: Left
(True, [1, 1], 'up')
Action: Forward
(True, [1, 2], [None, 'breeze', None, None, None])

Current Percepts: [None, 'breeze', None, None, None]

Rule: ~S12 => ~W11 and ~W13 and ~W22, ~S12
By Modus Ponens:
~W11 and ~W13 and ~W22

Rule: ~W11 and ~W13 and ~W22
By And Elimination:
~W11, ~W13, ~W22

Rule: B12 => P11 or P13 or P22, B12
By Modus Ponens:
P11 or P13 or P22

Rule: P11 or P13 or P22, ~P11

By Resolution:
P13 or P22

Rule: W12 or W21
~W12
By Resolution:
W21

Rule: ~P12 and ~P21
By And Elimination:
~P12, ~P21

Safe Neighbours: ['11']
Unknown Neighbours: ['13', '22']
Unsafe Neighbours: []

Action: Forward
(True, [1, 3], [None, 'breeze', None, None, None])

Current Percepts: [None, 'breeze', None, None, None]

Rule: ~S13 => ~W12 and ~W14 and ~W23, ~S13
By Modus Ponens:
~W12 and ~W14 and ~W23

Rule: ~W12 and ~W14 and ~W23
By And Elimination:
~W12, ~W14, ~W23

Rule: B13 => P12 or P14 or P23, B13
By Modus Ponens:
P12 or P14 or P23

Rule: P12 or P14 or P23
~P12
By Resolution:
P14 or P23

Rule: ~W11 and ~W13 and ~W22
By And Elimination:
~W11, ~W13, ~W22

Rule: P11 or P13 or P22
~P11, ~P13
By Resolution:
P22

Rule: P13 or P22
~P13

By Resolution:
P22

Safe Neighbours: ['12']
Unknown Neighbours: ['14', '23']
Unsafe Neighbours: []

Action: Right
(True, [1, 3], 'right')
Action: Forward
(True, [2, 3], [None, 'breeze', None, None, None])

Rule: ~S23 => ~W22 and ~W24 and ~W33 and ~W13, ~S23
By Modus Ponens:
~W22 and ~W24 and ~W33 and ~W13

Rule: ~W22 and ~W24 and ~W33 and ~W13
By And Elimination:
~W22, ~W24, ~W33, ~W13

Rule: B23 => P22 or P24 or P33 or P13, B23
By Modus Ponens:
P22 or P24 or P33 or P13

Rule: P22 or P24 or P33 or P13
~P13
By Resolution:
P22 or P24 or P33

Rule: ~W12 and ~W14 and ~W23
By And Elimination:
~W12, ~W14, ~W23

Rule: P12 or P14 or P23
~P12, ~P23
By Resolution:
P14

Rule: P14 or P23
~P23
By Resolution:
P14

Safe Neighbours: ['13']
Unknown Neighbours: ['24', '33']
Unsafe Neighbours: ['P22']

Action: Forward
(True, [3, 3], [None, None, None, None, None])

Rule: ~S33 => ~W32 and ~W34 and ~W43 and ~W23, ~S33
By Modus Ponens:
~W32 and ~W34 and ~W43 and ~W23

Rule: ~W32 and ~W34 and ~W43 and ~W23
By And Elimination:
~W32, ~W34, ~W43, ~W23

Rule: ~B33 => ~P32 and ~P34 and ~P43 and ~P23, ~B33
By Modus Ponens:
~P32 and ~P34 and ~P43 and ~P23

Rule: ~P32 and ~P34 and ~P43 and ~P23
By And Elimination:
~P32, ~P34, ~P43, ~P23

Rule: ~W22 and ~W24 and ~W33 and ~W13
By And Elimination:
~W22, ~W24, ~W33, ~W13

Rule: P22 or P24 or P33 or P13
~P33, ~P13
By Resolution:
P22 or P24

Rule: P22 or P24 or P33
~P33
By Resolution:
P22 or P24

Safe Neighbours: ['32', '34', '43', '23']
Unknown Neighbours: []
Unsafe Neighbours: []

Right
Action: Left
(True, [3, 3], 'up')
Action: Forward
(True, [3, 4], [None, None, 'gasp', None, None])
Action: Grab
(True, [3, 4], 'R2-D2 has picked up Luke')

Rule: ~S34 => ~W33 and ~W44 and ~W24, ~S34
By Modus Ponens:
~W33 and ~W44 and ~W24

Rule: ~W33 and ~W44 and ~W24
By And Elimination:

~W33, ~W44, ~W24

Rule: ~B34 => ~P33 and ~P44 and ~P24, ~B34
By Modus Ponens:
~P33 and ~P44 and ~P24

Rule: ~P33 and ~P44 and ~P24
By And Elimination:
~P33, ~P44, ~P24

Rule: ~W32 and ~W34 and ~W43 and ~W23
By And Elimination:
~W32, ~W34, ~W43, ~W23

Rule: ~P32 and ~P34 and ~P43 and ~P23
By And Elimination:
~P32, ~P34, ~P43, ~P23

Safe Neighbours: ['33', '44', '24']
Unknown Neighbours: []
Unsafe Neighbours: []

up
Action: Right
(True, [3, 4], 'right')
Action: Right
(True, [3, 4], 'down')
Action: Forward
(True, [3, 3], [None, None, None, None, None])

Rule: ~S33 => ~W32 and ~W34 and ~W43 and ~W23, ~S33
By Modus Ponens:
~W32 and ~W34 and ~W43 and ~W23

Rule: ~W32 and ~W34 and ~W43 and ~W23
By And Elimination:
~W32, ~W34, ~W43, ~W23

Rule: ~B33 => ~P32 and ~P34 and ~P43 and ~P23, ~B33
By Modus Ponens:
~P32 and ~P34 and ~P43 and ~P23

Rule: ~P32 and ~P34 and ~P43 and ~P23
By And Elimination:
~P32, ~P34, ~P43, ~P23

Rule: ~W22 and ~W24 and ~W33 and ~W13
By And Elimination:
~W22, ~W24, ~W33, ~W13

Rule: P22 or P24 or P33 or P13
~P24, ~P33, ~P13
By Resolution:
P22

Rule: P22 or P24 or P33
~P24, ~P33
By Resolution:
P22

Rule: ~W33 and ~W44 and ~W24
By And Elimination:
~W33, ~W44, ~W24

Rule: ~P33 and ~P44 and ~P24
By And Elimination:
~P33, ~P44, ~P24

Safe Neighbours: ['32', '34', '43', '23']
Unknown Neighbours: []
Unsafe Neighbours: []

Action: Right
(True, [3, 3], 'left')
Action: Forward
(True, [2, 3], [None, 'breeze', None, None, None])

Rule: ~S23 => ~W22 and ~W24 and ~W33 and ~W13, ~S23
By Modus Ponens:
~W22 and ~W24 and ~W33 and ~W13

Rule: ~W22 and ~W24 and ~W33 and ~W13
By And Elimination:
~W22, ~W24, ~W33, ~W13

Rule: B23 => P22 or P24 or P33 or P13, B23
By Modus Ponens:
P22 or P24 or P33 or P13

Rule: P22 or P24 or P33 or P13
~P24
~P33
~P13
By Resolution:
P22

Rule: ~W12 and ~W14 and ~W23
By And Elimination:

~W12
~W14
~W23

Rule: P12 or P14 or P23
~P12
~P23
By Resolution:
P14

Rule: P14 or P23
~P23
By Resolution:
P14

Rule: ~W32 and ~W34 and ~W43 and ~W23
By And Elimination:
~W32
~W34
~W43
~W23

Rule: ~P32 and ~P34 and ~P43 and ~P23
By And Elimination:
~P32
~P34
~P43
~P23

Safe Neighbours: ['24', '33', '13']
Unknown Neighbours: []
Unsafe Neighbours: ['P22']

Action: Forward
(True, [1, 3], [None, 'breeze', None, None, None])

Rule: ~S13 => ~W12 and ~W14 and ~W23, ~S13
By Modus Ponens:
~W12 and ~W14 and ~W23

Rule: ~W12 and ~W14 and ~W23
By And Elimination:
~W12
~W14
~W23

Rule: B13 => P12 or P14 or P23, B13
By Modus Ponens:
P12 or P14 or P23

Rule: P12 or P14 or P23
~P12
~P23
By Resolution:
P14

Rule: ~W11 and ~W13 and ~W22
By And Elimination:
~W11
~W13
~W22

Rule: P11 or P13 or P22
~P11
~P13
By Resolution:
P22

Rule: P13 or P22
~P13
By Resolution:
P22

Rule: ~W22 and ~W24 and ~W33 and ~W13
By And Elimination:
~W22
~W24
~W33
~W13

Rule: P22 or P24 or P33 or P13
~P24
~P33
~P13
By Resolution:
P22

Safe Neighbours: ['12', '23']
Unknown Neighbours: []
Unsafe Neighbours: ['P14']

Action: Left
(True, [1, 3], 'down')
Action: Forward
(True, [1, 2], [None, 'breeze', None, None, None])

Rule: ~S12 => ~W11 and ~W13 and ~W22, ~S12
By Modus Ponens:

~W11 and ~W13 and ~W22

Rule: ~W11 and ~W13 and ~W22
By And Elimination:
~W11
~W13
~W22

Rule: B12 => P11 or P13 or P22, B12
By Modus Ponens:
P11 or P13 or P22

Rule: P11 or P13 or P22
~P11
~P13
By Resolution:
P22

Rule: W12 or W21
~W12
By Resolution:
W21

Rule: ~P12 and ~P21
By And Elimination:
~P12
~P21

Rule: ~W12 and ~W14 and ~W23
By And Elimination:
~W12
~W14
~W23

Rule: P12 or P14 or P23
~P12
~P23
By Resolution:
P14

Safe Neighbours: ['11', '13']
Unknown Neighbours: []
Unsafe Neighbours: ['P22']

Action: Forward
(True, [1, 1], ['stench', None, None, None, None])
Action: Climb
(True, [1, 1], 'Congrats! R2 has saved Luke! +1000 points! Your final score: 981')

Scenario W2:

Current Direction : right
Current Location: [1, 1]
Current Percepts: [None, None, None, None, None]

Rule: ~S11 => ~W12 and ~W21, ~S11
By Modus Ponens:
~W12 and ~W21

Rule: ~W12 and ~W21
By And Elimination:
~W12
~W21

Rule: ~B11 => ~P12 and ~P21, ~B11
By Modus Ponens:
~P12 and ~P21

Rule: ~P12 and ~P21
By And Elimination:
~P12
~P21

Safe Neighbours: ['12', '21']
Unknown Neighbours: []
Unsafe Neighbours: []

Action: Forward
(True, [2, 1], [None, 'breeze', None, None, None])

Rule: ~S21 => ~W22 and ~W31 and ~W11, ~S21
By Modus Ponens:
~W22 and ~W31 and ~W11

Rule: ~W22 and ~W31 and ~W11
By And Elimination:
~W22
~W31
~W11

Rule: B21 => P22 or P31 or P11, B21
By Modus Ponens:
P22 or P31 or P11

Rule: P22 or P31 or P11
~P11
By Resolution:

P22 or P31

Rule: ~W12 and ~W21
By And Elimination:
~W12
~W21

Rule: ~P12 and ~P21
By And Elimination:
~P12
~P21

Safe Neighbours: ['11']
Unknown Neighbours: ['22', '31']
Unsafe Neighbours: []

Action: Right
(True, [2, 1], 'down')
Action: Right
(True, [2, 1], 'left')
Action: Forward
(True, [1, 1], [None, None, None, None, None])

Rule: ~S11 => ~W12 and ~W21, ~S11
By Modus Ponens:
~W12 and ~W21

Rule: ~W12 and ~W21
By And Elimination:
~W12
~W21

Rule: ~B11 => ~P12 and ~P21, ~B11
By Modus Ponens:
~P12 and ~P21

Rule: ~P12 and ~P21
By And Elimination:
~P12
~P21

Rule: ~W22 and ~W31 and ~W11
By And Elimination:
~W22
~W31
~W11

Rule: P22 or P31 or P11
~P11

By Resolution:
P22 or P31

Safe Neighbours: ['12', '21']
Unknown Neighbours: []
Unsafe Neighbours: []

Action: Forward
(False, [1, 1], [None, None, None, 'bump', None])

Rule: ~S11 => ~W12 and ~W21, ~S11
By Modus Ponens:
~W12 and ~W21

Rule: ~W12 and ~W21
By And Elimination:
~W12
~W21

Rule: ~B11 => ~P12 and ~P21, ~B11
By Modus Ponens:
~P12 and ~P21

Rule: ~P12 and ~P21
By And Elimination:
~P12
~P21

Rule: ~W22 and ~W31 and ~W11
By And Elimination:
~W22
~W31
~W11

Rule: P22 or P31 or P11
~P11
By Resolution:
P22 or P31

Safe Neighbours: ['12', '21']
Unknown Neighbours: []
Unsafe Neighbours: []

left
Action: Right
(True, [1, 1], 'up')
Action: Forward
(True, [1, 2], ['stench', None, None, None, None])

Rule: S12 => W11 or W13 or W22, S12
By Modus Ponens:
W11 or W13 or W22

Rule: W11 or W13 or W22
~W11
~W22
By Resolution:
W13

Rule: ~B12 => ~P11 and ~P13 and ~P22, ~B12
By Modus Ponens:
~P11 and ~P13 and ~P22

Rule: ~P11 and ~P13 and ~P22
By And Elimination:
~P11
~P13
~P22

Rule: ~W12 and ~W21
By And Elimination:
~W12
~W21

Rule: ~P12 and ~P21
By And Elimination:
~P12
~P21

Safe Neighbours: ['11', '22']
Unknown Neighbours: []
Unsafe Neighbours: ['W13']

up
Action: Right
(True, [1, 2], 'right')
Action: Forward
(True, [2, 2], [None, None, None, None, None])

Rule: ~S22 => ~W21 and ~W23 and ~W32 and ~W12, ~S22
By Modus Ponens:
~W21 and ~W23 and ~W32 and ~W12

Rule: ~W21 and ~W23 and ~W32 and ~W12
By And Elimination:
~W21
~W23
~W32

~W12

Rule: ~B22 => ~P21 and ~P23 and ~P32 and ~P12, ~B22
By Modus Ponens:
~P21 and ~P23 and ~P32 and ~P12

Rule: ~P21 and ~P23 and ~P32 and ~P12
By And Elimination:
~P21
~P23
~P32
~P12

Rule: ~W22 and ~W31 and ~W11
By And Elimination:
~W22
~W31
~W11

Rule: P22 or P31 or P11
~P22
~P11
By Resolution:
P31

Rule: P22 or P31
~P22
By Resolution:
P31

Rule: W11 or W13 or W22
~W11
~W22
By Resolution:
W13

Rule: ~P11 and ~P13 and ~P22
By And Elimination:
~P11
~P13
~P22

Safe Neighbours: ['21', '23', '32', '12']
Unknown Neighbours: []
Unsafe Neighbours: []

Action: Forward
(True, [3, 2], [None, 'breeze', None, None, None])

Rule: ~S32 => ~W31 and ~W33 and ~W42 and ~W22, ~S32
By Modus Ponens:
~W31 and ~W33 and ~W42 and ~W22

Rule: ~W31 and ~W33 and ~W42 and ~W22
By And Elimination:
~W31
~W33
~W42
~W22

Rule: B32 => P31 or P33 or P42 or P22, B32
By Modus Ponens:
P31 or P33 or P42 or P22

Rule: P31 or P33 or P42 or P22
~P22
By Resolution:
P31 or P33 or P42

Rule: ~W21 and ~W23 and ~W32 and ~W12
By And Elimination:
~W21
~W23
~W32
~W12

Rule: ~P21 and ~P23 and ~P32 and ~P12
By And Elimination:
~P21
~P23
~P32
~P12

Safe Neighbours: ['22']
Unknown Neighbours: ['33', '42']
Unsafe Neighbours: ['P31']

right
Action: Right
(True, [3, 2], 'down')
Action: Right
(True, [3, 2], 'left')
Action: Forward
(True, [2, 2], [None, None, None, None, None])

Rule: ~S22 => ~W21 and ~W23 and ~W32 and ~W12, ~S22
By Modus Ponens:
~W21 and ~W23 and ~W32 and ~W12

Rule: ~W21 and ~W23 and ~W32 and ~W12
By And Elimination:
~W21
~W23
~W32
~W12

Rule: ~B22 => ~P21 and ~P23 and ~P32 and ~P12, ~B22
By Modus Ponens:
~P21 and ~P23 and ~P32 and ~P12

Rule: ~P21 and ~P23 and ~P32 and ~P12
By And Elimination:
~P21
~P23
~P32
~P12

Rule: ~W22 and ~W31 and ~W11
By And Elimination:
~W22
~W31
~W11

Rule: P22 or P31 or P11
~P22
~P11
By Resolution:
P31

Rule: P22 or P31
~P22
By Resolution:
P31

Rule: W11 or W13 or W22
~W11
~W22
By Resolution:
W13

Rule: ~P11 and ~P13 and ~P22
By And Elimination:
~P11
~P13
~P22

Rule: ~W31 and ~W33 and ~W42 and ~W22

By And Elimination:
~W31
~W33
~W42
~W22

Rule: P31 or P33 or P42 or P22
~P22
By Resolution:
P31 or P33 or P42

Safe Neighbours: ['21', '23', '32', '12']
Unknown Neighbours: []
Unsafe Neighbours: []

Action: Right
(True, [2, 2], 'up')
Action: Forward
(True, [2, 3], ['stench', 'breeze', 'gasp', None, None])

Rule: S23 => W22 or W24 or W33 or W13, S23
By Modus Ponens:
W22 or W24 or W33 or W13

Rule: W22 or W24 or W33 or W13
~W22
~W33
By Resolution:
W24 or W13

Rule: B23 => P22 or P24 or P33 or P13, B23
By Modus Ponens:
P22 or P24 or P33 or P13

Rule: P22 or P24 or P33 or P13
~P22
~P13
By Resolution:
P24 or P33

Rule: ~W21 and ~W23 and ~W32 and ~W12
By And Elimination:
~W21
~W23
~W32
~W12

Rule: ~P21 and ~P23 and ~P32 and ~P12
By And Elimination:

~P21
~P23
~P32
~P12

Safe Neighbours: ['22']
Unknown Neighbours: ['24', '33']
Unsafe Neighbours: ['W13']

Action: Grab
(True, [2, 3], 'R2-D2 has picked up Luke')
up
Action: Right
(True, [2, 3], 'right')
Action: Right
(True, [2, 3], 'down')
Action: Forward
(True, [2, 2], [None, None, None, None, None])

Rule: ~S22 => ~W21 and ~W23 and ~W32 and ~W12, ~S22
By Modus Ponens:
~W21 and ~W23 and ~W32 and ~W12

Rule: ~W21 and ~W23 and ~W32 and ~W12
By And Elimination:
~W21
~W23
~W32
~W12

Rule: ~B22 => ~P21 and ~P23 and ~P32 and ~P12, ~B22
By Modus Ponens:
~P21 and ~P23 and ~P32 and ~P12

Rule: ~P21 and ~P23 and ~P32 and ~P12
By And Elimination:
~P21
~P23
~P32
~P12

Rule: ~W22 and ~W31 and ~W11
By And Elimination:
~W22
~W31
~W11

Rule: P22 or P31 or P11
~P22

~P11
By Resolution:
P31

Rule: P22 or P31
~P22
By Resolution:
P31

Rule: W11 or W13 or W22
~W11
~W22
By Resolution:
W13

Rule: ~P11 and ~P13 and ~P22
By And Elimination:
~P11
~P13
~P22

Rule: ~W31 and ~W33 and ~W42 and ~W22
By And Elimination:
~W31
~W33
~W42
~W22

Rule: P31 or P33 or P42 or P22
~P22
By Resolution:
P31 or P33 or P42

Rule: W22 or W24 or W33 or W13
~W22
~W33
By Resolution:
W24 or W13

Rule: P22 or P24 or P33 or P13
~P22
~P13
By Resolution:
P24 or P33

Safe Neighbours: ['21', '23', '32', '12']
Unknown Neighbours: []
Unsafe Neighbours: []

Action: Forward
(True, [2, 1], [None, 'breeze', None, None, None])

Rule: ~S21 => ~W22 and ~W31 and ~W11, ~S21
By Modus Ponens:
~W22 and ~W31 and ~W11

Rule: ~W22 and ~W31 and ~W11
By And Elimination:
~W22
~W31
~W11

Rule: B21 => P22 or P31 or P11, B21
By Modus Ponens:
P22 or P31 or P11

Rule: P22 or P31 or P11
~P22
~P11
By Resolution:
P31

Rule: ~W12 and ~W21
By And Elimination:
~W12
~W21

Rule: ~P12 and ~P21
By And Elimination:
~P12
~P21

Rule: ~W21 and ~W23 and ~W32 and ~W12
By And Elimination:
~W21
~W23
~W32
~W12

Rule: ~P21 and ~P23 and ~P32 and ~P12
By And Elimination:
~P21
~P23
~P32
~P12

Safe Neighbours: ['22', '11']
Unknown Neighbours: []

Unsafe Neighbours: ['P31']

Action: Right
(True, [2, 1], 'left')
Action: Forward
(True, [1, 1], [None, None, None, 'bump', None])
Action: Climb
(True, [1, 1], 'Congrats! R2 has saved Luke! +1000 points! Your final score: 977')

Scenario W3:

Current Direction : right
Current Location: [1, 1]
Current Percepts: [None, None, None, None, None]

Rule: ~S11 => ~W12 and ~W21, ~S11
By Modus Ponens:
~W12 and ~W21

Rule: ~W12 and ~W21
By And Elimination:
~W12
~W21

Rule: ~B11 => ~P12 and ~P21, ~B11
By Modus Ponens:
~P12 and ~P21

Rule: ~P12 and ~P21
By And Elimination:
~P12
~P21

Safe Neighbours: ['12', '21']
Unknown Neighbours: []
Unsafe Neighbours: []

Action: Forward
(True, [2, 1], [None, None, None, None, None])

Rule: ~S21 => ~W22 and ~W31 and ~W11, ~S21
By Modus Ponens:
~W22 and ~W31 and ~W11

Rule: ~W22 and ~W31 and ~W11
By And Elimination:
~W22
~W31

~W11

Rule: ~B21 => ~P22 and ~P31 and ~P11, ~B21
By Modus Ponens:
~P22 and ~P31 and ~P11

Rule: ~P22 and ~P31 and ~P11
By And Elimination:
~P22
~P31
~P11

Rule: ~W12 and ~W21
By And Elimination:
~W12
~W21

Rule: ~P12 and ~P21
By And Elimination:
~P12
~P21

Safe Neighbours: ['22', '31', '11']
Unknown Neighbours: []
Unsafe Neighbours: []

Action: Forward
(True, [3, 1], [None, 'breeze', None, None, None])

Rule: ~S31 => ~W32 and ~W41 and ~W21, ~S31
By Modus Ponens:
~W32 and ~W41 and ~W21

Rule: ~W32 and ~W41 and ~W21
By And Elimination:
~W32
~W41
~W21

Rule: B31 => P32 or P41 or P21, B31
By Modus Ponens:
P32 or P41 or P21

Rule: P32 or P41 or P21
~P21
By Resolution:
P32 or P41

Rule: ~W22 and ~W31 and ~W11

By And Elimination:
~W22
~W31
~W11

Rule: ~P22 and ~P31 and ~P11
By And Elimination:
~P22
~P31
~P11

Safe Neighbours: ['21']
Unknown Neighbours: ['32', '41']
Unsafe Neighbours: []

Action: Right
(True, [3, 1], 'down')
Action: Right
(True, [3, 1], 'left')
Action: Forward
(True, [2, 1], [None, None, None, None, None])
Action: Right
(True, [2, 1], 'up')
Action: Forward
(True, [2, 2], [None, 'breeze', None, None, None])

Rule: ~S22 => ~W21 and ~W23 and ~W32 and ~W12, ~S22
By Modus Ponens:
~W21 and ~W23 and ~W32 and ~W12

Rule: ~W21 and ~W23 and ~W32 and ~W12
By And Elimination:
~W21
~W23
~W32
~W12

Rule: B22 => P21 or P23 or P32 or P12, B22
By Modus Ponens:
P21 or P23 or P32 or P12

Rule: P21 or P23 or P32 or P12
~P21
~P12
By Resolution:
P23 or P32

Rule: ~W22 and ~W31 and ~W11
By And Elimination:

~W22
~W31
~W11

Rule: ~P22 and ~P31 and ~P11
By And Elimination:
~P22
~P31
~P11

Safe Neighbours: ['21', '12']
Unknown Neighbours: ['23', '32']
Unsafe Neighbours: []

Action: Left
(True, [2, 2], 'left')
Action: Forward
(True, [1, 2], [None, None, None, None, None])

Rule: ~S12 => ~W11 and ~W13 and ~W22, ~S12
By Modus Ponens:
~W11 and ~W13 and ~W22

Rule: ~W11 and ~W13 and ~W22
By And Elimination:
~W11
~W13
~W22

Rule: ~B12 => ~P11 and ~P13 and ~P22, ~B12
By Modus Ponens:
~P11 and ~P13 and ~P22

Rule: ~P11 and ~P13 and ~P22
By And Elimination:
~P11
~P13
~P22

Rule: ~W12 and ~W21
By And Elimination:
~W12
~W21

Rule: ~P12 and ~P21
By And Elimination:
~P12
~P21

Rule: ~W21 and ~W23 and ~W32 and ~W12
By And Elimination:
~W21
~W23
~W32
~W12

Rule: P21 or P23 or P32 or P12
~P21
~P12
By Resolution:
P23 or P32

Safe Neighbours: ['11', '13', '22']
Unknown Neighbours: []
Unsafe Neighbours: []

Action: Right
(True, [1, 2], 'up')
Action: Forward
(True, [1, 3], [None, 'breeze', None, None, None])

Rule: ~S13 => ~W12 and ~W14 and ~W23, ~S13
By Modus Ponens:
~W12 and ~W14 and ~W23

Rule: ~W12 and ~W14 and ~W23
By And Elimination:
~W12
~W14
~W23

Rule: B13 => P12 or P14 or P23, B13
By Modus Ponens:
P12 or P14 or P23

Rule: P12 or P14 or P23
~P12
By Resolution:
P14 or P23

Rule: ~W11 and ~W13 and ~W22
By And Elimination:
~W11
~W13
~W22

Rule: ~P11 and ~P13 and ~P22
By And Elimination:

~P11
~P13
~P22

Safe Neighbours: ['12']
Unknown Neighbours: ['14', '23']
Unsafe Neighbours: []

Action: Forward
(True, [1, 4], [None, 'breeze', None, None, None])

Rule: ~S14 => ~W13 and ~W15 and ~W24, ~S14
By Modus Ponens:
~W13 and ~W15 and ~W24

Rule: ~W13 and ~W15 and ~W24
By And Elimination:
~W13
~W15
~W24

Rule: B14 => P13 or P15 or P24, B14
By Modus Ponens:
P13 or P15 or P24

Rule: P13 or P15 or P24
~P13
By Resolution:
P15 or P24

Rule: ~W12 and ~W14 and ~W23
By And Elimination:
~W12
~W14
~W23

Rule: P12 or P14 or P23
~P12
~P14
By Resolution:
P23

Rule: P14 or P23
~P14
By Resolution:
P23

Safe Neighbours: ['13']
Unknown Neighbours: ['15', '24']

Unsafe Neighbours: []

Action: Right
(True, [1, 4], 'right')
Action: Forward
(True, [2, 4], [None, 'breeze', None, None, None])

Rule: ~S24 => ~W23 and ~W25 and ~W34 and ~W14, ~S24
By Modus Ponens:
~W23 and ~W25 and ~W34 and ~W14

Rule: ~W23 and ~W25 and ~W34 and ~W14
By And Elimination:
~W23
~W25
~W34
~W14

Rule: B24 => P23 or P25 or P34 or P14, B24
By Modus Ponens:
P23 or P25 or P34 or P14

Rule: P23 or P25 or P34 or P14
~P14
By Resolution:
P23 or P25 or P34

Rule: ~W13 and ~W15 and ~W24
By And Elimination:
~W13
~W15
~W24

Rule: P13 or P15 or P24
~P13
~P24
By Resolution:
P15

Rule: P15 or P24
~P24
By Resolution:
P15

Safe Neighbours: ['14']
Unknown Neighbours: ['25', '34']
Unsafe Neighbours: ['P23']

right

Action: Left
(True, [2, 4], 'up')
Action: Forward
(True, [2, 5], [None, 'breeze', 'gasp', None, None])

Rule: ~S25 => ~W24 and ~W35 and ~W15, ~S25
By Modus Ponens:
~W24 and ~W35 and ~W15

Rule: ~W24 and ~W35 and ~W15
By And Elimination:
~W24
~W35
~W15

Rule: B25 => P24 or P35 or P15, B25
By Modus Ponens:
P24 or P35 or P15

Rule: P24 or P35 or P15
~P24
By Resolution:
P35 or P15

Rule: ~W23 and ~W25 and ~W34 and ~W14
By And Elimination:
~W23
~W25
~W34
~W14

Rule: P23 or P25 or P34 or P14
~P25
~P14
By Resolution:
P23 or P34

Rule: P23 or P25 or P34
~P25
By Resolution:
P23 or P34

Safe Neighbours: ['24']
Unknown Neighbours: ['35']
Unsafe Neighbours: ['P15']

Action: Grab
(True, [2, 5], 'R2-D2 has picked up Luke')
Action: Right

(True, [2, 5], 'right')
Action: Right
(True, [2, 5], 'down')
Action: Forward
(True, [2, 4], [None, 'breeze', None, None, None])

Rule: ~S24 => ~W23 and ~W25 and ~W34 and ~W14, ~S24
By Modus Ponens:
~W23 and ~W25 and ~W34 and ~W14

Rule: ~W23 and ~W25 and ~W34 and ~W14
By And Elimination:
~W23
~W25
~W34
~W14

Rule: B24 => P23 or P25 or P34 or P14, B24
By Modus Ponens:
P23 or P25 or P34 or P14

Rule: P23 or P25 or P34 or P14
~P25
~P14
By Resolution:
P23 or P34

Rule: ~W13 and ~W15 and ~W24
By And Elimination:
~W13
~W15
~W24

Rule: P13 or P15 or P24
~P13
~P24
By Resolution:
P15

Rule: P15 or P24
~P24
By Resolution:
P15

Rule: ~W24 and ~W35 and ~W15
By And Elimination:
~W24
~W35
~W15

Rule: P24 or P35 or P15
~P24
By Resolution:
P35 or P15

Safe Neighbours: ['25', '14']
Unknown Neighbours: ['34']
Unsafe Neighbours: ['P23']

Action: Right
(True, [2, 4], 'left')
Action: Forward
(True, [1, 4], [None, 'breeze', None, None, None])

Rule: ~S14 => ~W13 and ~W15 and ~W24, ~S14
By Modus Ponens:
~W13 and ~W15 and ~W24

Rule: ~W13 and ~W15 and ~W24
By And Elimination:
~W13
~W15
~W24

Rule: B14 => P13 or P15 or P24, B14
By Modus Ponens:
P13 or P15 or P24

Rule: P13 or P15 or P24
~P13
~P24
By Resolution:
P15

Rule: ~W12 and ~W14 and ~W23
By And Elimination:
~W12
~W14
~W23

Rule: P12 or P14 or P23
~P12
~P14
By Resolution:
P23

Rule: P14 or P23
~P14

By Resolution:
P23

Rule: ~W23 and ~W25 and ~W34 and ~W14
By And Elimination:
~W23
~W25
~W34
~W14

Rule: P23 or P25 or P34 or P14
~P25
~P14
By Resolution:
P23 or P34

Safe Neighbours: ['13', '24']
Unknown Neighbours: []
Unsafe Neighbours: ['P15']

Action: Left
(True, [1, 4], 'down')
Action: Forward
(True, [1, 3], [None, 'breeze', None, None, None])

Rule: ~S13 => ~W12 and ~W14 and ~W23, ~S13
By Modus Ponens:
~W12 and ~W14 and ~W23

Rule: ~W12 and ~W14 and ~W23
By And Elimination:
~W12
~W14
~W23

Rule: B13 => P12 or P14 or P23, B13
By Modus Ponens:
P12 or P14 or P23

Rule: P12 or P14 or P23
~P12
~P14
By Resolution:
P23

Rule: ~W11 and ~W13 and ~W22
By And Elimination:
~W11
~W13

~W22

Rule: ~P11 and ~P13 and ~P22
By And Elimination:
~P11
~P13
~P22

Rule: ~W13 and ~W15 and ~W24
By And Elimination:
~W13
~W15
~W24

Rule: P13 or P15 or P24
~P13
~P24
By Resolution:
P15

Safe Neighbours: ['12', '14']
Unknown Neighbours: []
Unsafe Neighbours: ['P23']

Action: Forward
(True, [1, 2], [None, None, None, None, None])

Rule: ~S12 => ~W11 and ~W13 and ~W22, ~S12
By Modus Ponens:
~W11 and ~W13 and ~W22

Rule: ~W11 and ~W13 and ~W22
By And Elimination:
~W11
~W13
~W22

Rule: ~B12 => ~P11 and ~P13 and ~P22, ~B12
By Modus Ponens:
~P11 and ~P13 and ~P22

Rule: ~P11 and ~P13 and ~P22
By And Elimination:
~P11
~P13
~P22

Rule: ~W12 and ~W21
By And Elimination:

~W12
~W21

Rule: ~P12 and ~P21
By And Elimination:
~P12
~P21

Rule: ~W21 and ~W23 and ~W32 and ~W12
By And Elimination:
~W21
~W23
~W32
~W12

Rule: P21 or P23 or P32 or P12
~P21
~P12
By Resolution:
P23 or P32

Rule: ~W12 and ~W14 and ~W23
By And Elimination:
~W12
~W14
~W23

Rule: P12 or P14 or P23
~P12
~P14
By Resolution:
P23

Safe Neighbours: ['11', '13', '22']
Unknown Neighbours: []
Unsafe Neighbours: []

Action: Forward
(True, [1, 1], [None, None, None, None, None])
Action: Climb
(True, [1, 1], 'Congrats! R2 has saved Luke! +1000 points! Your final score: 973')

Scenario W4:

Current Direction : right
Current Location: [1, 1]
Current Percepts: [None, 'breeze', None, None, None]

Rule: ~S11 => ~W12 and ~W21, ~S11
By Modus Ponens:
~W12 and ~W21

Rule: ~W12 and ~W21
By And Elimination:
~W12
~W21

Rule: B11 => P12 or P21, B11
By Modus Ponens:
P12 or P21

Rule: P12 or P21
By Resolution:
P12 or P21

Safe Neighbours: []
Unknown Neighbours: ['12', '21']
Unsafe Neighbours: []

Action: Forward
(True, [2, 1], [None, 'breeze', None, None, None])

Rule: ~S21 => ~W22 and ~W31 and ~W11, ~S21
By Modus Ponens:
~W22 and ~W31 and ~W11

Rule: ~W22 and ~W31 and ~W11
By And Elimination:
~W22
~W31
~W11

Rule: B21 => P22 or P31 or P11, B21
By Modus Ponens:
P22 or P31 or P11

Rule: P22 or P31 or P11
~P11
By Resolution:
P22 or P31

Rule: ~W12 and ~W21
By And Elimination:
~W12
~W21

Rule: P12 or P21
~P21
By Resolution:
P12

Safe Neighbours: ['11']
Unknown Neighbours: ['22', '31']
Unsafe Neighbours: []

Action: Forward
(True, [3, 1], ['stench', None, None, None, None])

Rule: S31 => W32 or W41 or W21, S31
By Modus Ponens:
W32 or W41 or W21

Rule: W32 or W41 or W21
~W21
By Resolution:
W32 or W41

Rule: ~B31 => ~P32 and ~P41 and ~P21, ~B31
By Modus Ponens:
~P32 and ~P41 and ~P21

Rule: ~P32 and ~P41 and ~P21
By And Elimination:
~P32
~P41
~P21

Rule: ~W22 and ~W31 and ~W11
By And Elimination:
~W22
~W31
~W11

Rule: P22 or P31 or P11
~P31
~P11
By Resolution:
P22

Rule: P22 or P31
~P31
By Resolution:
P22

Safe Neighbours: ['21']

Unknown Neighbours: ['32', '41']
Unsafe Neighbours: []

Action: Left
(True, [3, 1], 'up')
Action: Forward
(True, [3, 2], [None, 'breeze', None, None, None])

Rule: ~S32 => ~W31 and ~W33 and ~W42 and ~W22, ~S32
By Modus Ponens:
~W31 and ~W33 and ~W42 and ~W22

Rule: ~W31 and ~W33 and ~W42 and ~W22
By And Elimination:
~W31
~W33
~W42
~W22

Rule: B32 => P31 or P33 or P42 or P22, B32
By Modus Ponens:
P31 or P33 or P42 or P22

Rule: P31 or P33 or P42 or P22
~P31
By Resolution:
P33 or P42 or P22

Rule: W32 or W41 or W21
~W32
~W21
By Resolution:
W41

Rule: W32 or W41
~W32
By Resolution:
W41

Rule: ~P32 and ~P41 and ~P21
By And Elimination:
~P32
~P41
~P21

Safe Neighbours: ['31']
Unknown Neighbours: ['33', '42']
Unsafe Neighbours: ['P22']

Action: Right
(True, [3, 2], 'right')
Action: Forward
(True, [4, 2], ['stench', None, None, None, None])

Rule: S42 => W41 or W43 or W52 or W32, S42
By Modus Ponens:
W41 or W43 or W52 or W32

Rule: W41 or W43 or W52 or W32
~W32
By Resolution:
W41 or W43 or W52

Rule: ~B42 => ~P41 and ~P43 and ~P52 and ~P32, ~B42
By Modus Ponens:
~P41 and ~P43 and ~P52 and ~P32

Rule: ~P41 and ~P43 and ~P52 and ~P32
By And Elimination:
~P41
~P43
~P52
~P32

Rule: ~W31 and ~W33 and ~W42 and ~W22
By And Elimination:
~W31
~W33
~W42
~W22

Rule: P31 or P33 or P42 or P22
~P31
~P42
By Resolution:
P33 or P22

Rule: P33 or P42 or P22
~P42
By Resolution:
P33 or P22

Safe Neighbours: ['32']
Unknown Neighbours: ['43', '52']
Unsafe Neighbours: ['W41']

Action: Left
(True, [4, 2], 'up')

Action: Forward
(True, [4, 3], [None, 'breeze', None, None, None])

Rule: ~S43 => ~W42 and ~W44 and ~W53 and ~W33, ~S43
By Modus Ponens:
~W42 and ~W44 and ~W53 and ~W33

Rule: ~W42 and ~W44 and ~W53 and ~W33
By And Elimination:
~W42
~W44
~W53
~W33

Rule: B43 => P42 or P44 or P53 or P33, B43
By Modus Ponens:
P42 or P44 or P53 or P33

Rule: P42 or P44 or P53 or P33
~P42
By Resolution:
P44 or P53 or P33

Rule: W41 or W43 or W52 or W32
~W43
~W32
By Resolution:
W41 or W52

Rule: W41 or W43 or W52
~W43
By Resolution:
W41 or W52

Rule: ~P41 and ~P43 and ~P52 and ~P32
By And Elimination:
~P41
~P43
~P52
~P32

Safe Neighbours: ['42']
Unknown Neighbours: ['44', '53', '33']
Unsafe Neighbours: []

up
Action: Left
(True, [4, 3], 'right')
(True, [4, 3], 'down')

(True, [4, 3], 'left')
Action: Forward
(True, [3, 3], [None, 'breeze', None, None, None])

Rule: ~S33 => ~W32 and ~W34 and ~W43 and ~W23, ~S33
By Modus Ponens:
~W32 and ~W34 and ~W43 and ~W23

Rule: ~W32 and ~W34 and ~W43 and ~W23
By And Elimination:
~W32
~W34
~W43
~W23

Rule: B33 => P32 or P34 or P43 or P23, B33
By Modus Ponens:
P32 or P34 or P43 or P23

Rule: P32 or P34 or P43 or P23
~P32
~P43
By Resolution:
P34 or P23

Rule: ~W31 and ~W33 and ~W42 and ~W22
By And Elimination:
~W31
~W33
~W42
~W22

Rule: P31 or P33 or P42 or P22
~P31
~P33
~P42
By Resolution:
P22

Rule: P33 or P42 or P22
~P33
~P42
By Resolution:
P22

Rule: P33 or P22
~P33
By Resolution:
P22

Rule: ~W42 and ~W44 and ~W53 and ~W33
By And Elimination:
~W42
~W44
~W53
~W33

Rule: P42 or P44 or P53 or P33
~P42
~P33
By Resolution:
P44 or P53

Rule: P44 or P53 or P33
~P33
By Resolution:
P44 or P53

Safe Neighbours: ['32', '43']
Unknown Neighbours: ['34', '23']
Unsafe Neighbours: []

Action: Forward
(True, [2, 3], [None, 'breeze', None, None, None])

Rule: ~S23 => ~W22 and ~W24 and ~W33 and ~W13, ~S23
By Modus Ponens:
~W22 and ~W24 and ~W33 and ~W13

Rule: ~W22 and ~W24 and ~W33 and ~W13
By And Elimination:
~W22
~W24
~W33
~W13

Rule: B23 => P22 or P24 or P33 or P13, B23
By Modus Ponens:
P22 or P24 or P33 or P13

Rule: P22 or P24 or P33 or P13
~P33
By Resolution:
P22 or P24 or P13

Rule: ~W32 and ~W34 and ~W43 and ~W23
By And Elimination:
~W32

~W34
~W43
~W23

Rule: P32 or P34 or P43 or P23
~P32
~P43
~P23
By Resolution:
P34

Rule: P34 or P23
~P23
By Resolution:
P34

Safe Neighbours: ['33']
Unknown Neighbours: ['24', '13']
Unsafe Neighbours: ['P22']

Action: Right
(True, [2, 3], 'up')
Action: Forward
(True, [2, 4], [None, 'breeze', None, None, None])

Rule: ~S24 => ~W23 and ~W25 and ~W34 and ~W14, ~S24
By Modus Ponens:
~W23 and ~W25 and ~W34 and ~W14

Rule: ~W23 and ~W25 and ~W34 and ~W14
By And Elimination:
~W23
~W25
~W34
~W14

Rule: B24 => P23 or P25 or P34 or P14, B24
By Modus Ponens:
P23 or P25 or P34 or P14

Rule: P23 or P25 or P34 or P14
~P23
By Resolution:
P25 or P34 or P14

Rule: ~W22 and ~W24 and ~W33 and ~W13
By And Elimination:
~W22
~W24

~W33
~W13

Rule: P22 or P24 or P33 or P13
~P24
~P33
By Resolution:
P22 or P13

Rule: P22 or P24 or P13
~P24
By Resolution:
P22 or P13

Safe Neighbours: ['23']
Unknown Neighbours: ['25', '14']
Unsafe Neighbours: ['P34']

Action: Forward
(True, [2, 5], [None, None, None, None, None])

Rule: ~S25 => ~W24 and ~W26 and ~W35 and ~W15, ~S25
By Modus Ponens:
~W24 and ~W26 and ~W35 and ~W15

Rule: ~W24 and ~W26 and ~W35 and ~W15
By And Elimination:
~W24
~W26
~W35
~W15

Rule: ~B25 => ~P24 and ~P26 and ~P35 and ~P15, ~B25
By Modus Ponens:
~P24 and ~P26 and ~P35 and ~P15

Rule: ~P24 and ~P26 and ~P35 and ~P15
By And Elimination:
~P24
~P26
~P35
~P15

Rule: ~W23 and ~W25 and ~W34 and ~W14
By And Elimination:
~W23
~W25
~W34
~W14

Rule: P23 or P25 or P34 or P14
~P23
~P25
By Resolution:
P34 or P14

Rule: P25 or P34 or P14
~P25
By Resolution:
P34 or P14

Safe Neighbours: ['24', '26', '35', '15']
Unknown Neighbours: []
Unsafe Neighbours: []

Action: Right
(True, [2, 5], 'right')
Action: Forward
(True, [3, 5], [None, 'breeze', None, None, None])

Rule: ~S35 => ~W34 and ~W36 and ~W45 and ~W25, ~S35
By Modus Ponens:
~W34 and ~W36 and ~W45 and ~W25

Rule: ~W34 and ~W36 and ~W45 and ~W25
By And Elimination:
~W34
~W36
~W45
~W25

Rule: B35 => P34 or P36 or P45 or P25, B35
By Modus Ponens:
P34 or P36 or P45 or P25

Rule: P34 or P36 or P45 or P25
~P25
By Resolution:
P34 or P36 or P45

Rule: ~W24 and ~W26 and ~W35 and ~W15
By And Elimination:
~W24
~W26
~W35
~W15

Rule: ~P24 and ~P26 and ~P35 and ~P15

By And Elimination:
~P24
~P26
~P35
~P15

Safe Neighbours: ['25']
Unknown Neighbours: ['36', '45']
Unsafe Neighbours: ['P34']

Action: Forward
(True, [4, 5], [None, 'breeze', None, None, None])

Rule: ~S45 => ~W44 and ~W46 and ~W55 and ~W35, ~S45
By Modus Ponens:
~W44 and ~W46 and ~W55 and ~W35

Rule: ~W44 and ~W46 and ~W55 and ~W35
By And Elimination:
~W44
~W46
~W55
~W35

Rule: B45 => P44 or P46 or P55 or P35, B45
By Modus Ponens:
P44 or P46 or P55 or P35

Rule: P44 or P46 or P55 or P35
~P35
By Resolution:
P44 or P46 or P55

Rule: ~W34 and ~W36 and ~W45 and ~W25
By And Elimination:
~W34
~W36
~W45
~W25

Rule: P34 or P36 or P45 or P25
~P45
~P25
By Resolution:
P34 or P36

Rule: P34 or P36 or P45
~P45
By Resolution:

P34 or P36

Safe Neighbours: ['35']
Unknown Neighbours: ['44', '46', '55']
Unsafe Neighbours: []

Action: Forward
(True, [5, 5], [None, None, None, None, None])

Rule: ~S55 => ~W54 and ~W56 and ~W65 and ~W45, ~S55
By Modus Ponens:
~W54 and ~W56 and ~W65 and ~W45

Rule: ~W54 and ~W56 and ~W65 and ~W45
By And Elimination:
~W54
~W56
~W65
~W45

Rule: ~B55 => ~P54 and ~P56 and ~P65 and ~P45, ~B55
By Modus Ponens:
~P54 and ~P56 and ~P65 and ~P45

Rule: ~P54 and ~P56 and ~P65 and ~P45
By And Elimination:
~P54
~P56
~P65
~P45

Rule: ~W44 and ~W46 and ~W55 and ~W35
By And Elimination:
~W44
~W46
~W55
~W35

Rule: P44 or P46 or P55 or P35
~P55
~P35
By Resolution:
P44 or P46

Rule: P44 or P46 or P55
~P55
By Resolution:
P44 or P46

Safe Neighbours: ['54', '56', '65', '45']
Unknown Neighbours: []
Unsafe Neighbours: []

Action: Right
(True, [5, 5], 'down')
Action: Forward
(True, [5, 4], [None, 'breeze', 'gasp', None, None])

Rule: ~S54 => ~W53 and ~W55 and ~W64 and ~W44, ~S54
By Modus Ponens:
~W53 and ~W55 and ~W64 and ~W44

Rule: ~W53 and ~W55 and ~W64 and ~W44
By And Elimination:
~W53
~W55
~W64
~W44

Rule: B54 => P53 or P55 or P64 or P44, B54
By Modus Ponens:
P53 or P55 or P64 or P44

Rule: P53 or P55 or P64 or P44
~P55
By Resolution:
P53 or P64 or P44

Rule: ~W54 and ~W56 and ~W65 and ~W45
By And Elimination:
~W54
~W56
~W65
~W45

Rule: ~P54 and ~P56 and ~P65 and ~P45
By And Elimination:
~P54
~P56
~P65
~P45

Safe Neighbours: ['55']
Unknown Neighbours: ['53', '64', '44']
Unsafe Neighbours: []

Action: Grab
(True, [5, 4], 'R2-D2 has picked up Luke')

Action: Right
(True, [5, 4], 'left')
Action: Right
(True, [5, 4], 'up')
Action: Forward
(True, [5, 5], [None, None, None, None, None])

Rule: ~S55 => ~W54 and ~W56 and ~W65 and ~W45, ~S55
By Modus Ponens:
~W54 and ~W56 and ~W65 and ~W45

Rule: ~W54 and ~W56 and ~W65 and ~W45
By And Elimination:
~W54
~W56
~W65
~W45

Rule: ~B55 => ~P54 and ~P56 and ~P65 and ~P45, ~B55
By Modus Ponens:
~P54 and ~P56 and ~P65 and ~P45

Rule: ~P54 and ~P56 and ~P65 and ~P45
By And Elimination:
~P54
~P56
~P65
~P45

Rule: ~W44 and ~W46 and ~W55 and ~W35
By And Elimination:
~W44
~W46
~W55
~W35

Rule: P44 or P46 or P55 or P35
~P55
~P35
By Resolution:
P44 or P46

Rule: P44 or P46 or P55
~P55
By Resolution:
P44 or P46

Rule: ~W53 and ~W55 and ~W64 and ~W44
By And Elimination:

~W53
~W55
~W64
~W44

Rule: P53 or P55 or P64 or P44
~P55
By Resolution:
P53 or P64 or P44

Safe Neighbours: ['54', '56', '65', '45']
Unknown Neighbours: []
Unsafe Neighbours: []

Action: Left
(True, [5, 5], 'left')
Action: Forward
(True, [4, 5], [None, 'breeze', None, None, None])

Rule: ~S45 => ~W44 and ~W46 and ~W55 and ~W35, ~S45
By Modus Ponens:
~W44 and ~W46 and ~W55 and ~W35

Rule: ~W44 and ~W46 and ~W55 and ~W35
By And Elimination:
~W44
~W46
~W55
~W35

Rule: B45 => P44 or P46 or P55 or P35, B45
By Modus Ponens:
P44 or P46 or P55 or P35

Rule: P44 or P46 or P55 or P35
~P55
~P35
By Resolution:
P44 or P46

Rule: ~W34 and ~W36 and ~W45 and ~W25
By And Elimination:
~W34
~W36
~W45
~W25

Rule: P34 or P36 or P45 or P25
~P45

~P25
By Resolution:
P34 or P36

Rule: P34 or P36 or P45
~P45
By Resolution:
P34 or P36

Rule: ~W54 and ~W56 and ~W65 and ~W45
By And Elimination:
~W54
~W56
~W65
~W45

Rule: ~P54 and ~P56 and ~P65 and ~P45
By And Elimination:
~P54
~P56
~P65
~P45

Safe Neighbours: ['55', '35']
Unknown Neighbours: ['44', '46']
Unsafe Neighbours: []

Action: Forward
(True, [3, 5], [None, 'breeze', None, None, None])

Rule: ~S35 => ~W34 and ~W36 and ~W45 and ~W25, ~S35
By Modus Ponens:
~W34 and ~W36 and ~W45 and ~W25

Rule: ~W34 and ~W36 and ~W45 and ~W25
By And Elimination:
~W34
~W36
~W45
~W25

Rule: B35 => P34 or P36 or P45 or P25, B35
By Modus Ponens:
P34 or P36 or P45 or P25

Rule: P34 or P36 or P45 or P25
~P45
~P25
By Resolution:

P34 or P36

Rule: ~W24 and ~W26 and ~W35 and ~W15
By And Elimination:
~W24
~W26
~W35
~W15

Rule: ~P24 and ~P26 and ~P35 and ~P15
By And Elimination:
~P24
~P26
~P35
~P15

Rule: ~W44 and ~W46 and ~W55 and ~W35
By And Elimination:
~W44
~W46
~W55
~W35

Rule: P44 or P46 or P55 or P35
~P55
~P35
By Resolution:
P44 or P46

Safe Neighbours: ['45', '25']
Unknown Neighbours: ['36']
Unsafe Neighbours: ['P34']

Action: Forward
(True, [2, 5], [None, None, None, None, None])

Rule: ~S25 => ~W24 and ~W26 and ~W35 and ~W15, ~S25
By Modus Ponens:
~W24 and ~W26 and ~W35 and ~W15

Rule: ~W24 and ~W26 and ~W35 and ~W15
By And Elimination:
~W24
~W26
~W35
~W15

Rule: ~B25 => ~P24 and ~P26 and ~P35 and ~P15, ~B25
By Modus Ponens:

~P24 and ~P26 and ~P35 and ~P15

Rule: ~P24 and ~P26 and ~P35 and ~P15
By And Elimination:
~P24
~P26
~P35
~P15

Rule: ~W23 and ~W25 and ~W34 and ~W14
By And Elimination:
~W23
~W25
~W34
~W14

Rule: P23 or P25 or P34 or P14
~P23
~P25
By Resolution:
P34 or P14

Rule: P25 or P34 or P14
~P25
By Resolution:
P34 or P14

Rule: ~W34 and ~W36 and ~W45 and ~W25
By And Elimination:
~W34
~W36
~W45
~W25

Rule: P34 or P36 or P45 or P25
~P45
~P25
By Resolution:
P34 or P36

Safe Neighbours: ['24', '26', '35', '15']
Unknown Neighbours: []
Unsafe Neighbours: []

Action: Left
(True, [2, 5], 'down')
Action: Forward
(True, [2, 4], [None, 'breeze', None, None, None])

Rule: ~S24 => ~W23 and ~W25 and ~W34 and ~W14, ~S24
By Modus Ponens:
~W23 and ~W25 and ~W34 and ~W14

Rule: ~W23 and ~W25 and ~W34 and ~W14
By And Elimination:
~W23
~W25
~W34
~W14

Rule: B24 => P23 or P25 or P34 or P14, B24
By Modus Ponens:
P23 or P25 or P34 or P14

Rule: P23 or P25 or P34 or P14
~P23
~P25
By Resolution:
P34 or P14

Rule: ~W22 and ~W24 and ~W33 and ~W13
By And Elimination:
~W22
~W24
~W33
~W13

Rule: P22 or P24 or P33 or P13
~P24
~P33
By Resolution:
P22 or P13

Rule: P22 or P24 or P13
~P24
By Resolution:
P22 or P13

Rule: ~W24 and ~W26 and ~W35 and ~W15
By And Elimination:
~W24
~W26
~W35
~W15

Rule: ~P24 and ~P26 and ~P35 and ~P15
By And Elimination:
~P24

~P26
~P35
~P15

Safe Neighbours: ['23', '25']
Unknown Neighbours: ['14']
Unsafe Neighbours: ['P34']

Action: Forward
(True, [2, 3], [None, 'breeze', None, None, None])

Rule: ~S23 => ~W22 and ~W24 and ~W33 and ~W13, ~S23
By Modus Ponens:
~W22 and ~W24 and ~W33 and ~W13

Rule: ~W22 and ~W24 and ~W33 and ~W13
By And Elimination:
~W22
~W24
~W33
~W13

Rule: B23 => P22 or P24 or P33 or P13, B23
By Modus Ponens:
P22 or P24 or P33 or P13

Rule: P22 or P24 or P33 or P13
~P24
~P33
By Resolution:
P22 or P13

Rule: ~W32 and ~W34 and ~W43 and ~W23
By And Elimination:
~W32
~W34
~W43
~W23

Rule: P32 or P34 or P43 or P23
~P32
~P43
~P23
By Resolution:
P34

Rule: P34 or P23
~P23
By Resolution:

P34

Rule: ~W23 and ~W25 and ~W34 and ~W14
By And Elimination:
~W23
~W25
~W34
~W14

Rule: P23 or P25 or P34 or P14
~P23
~P25
By Resolution:
P34 or P14

Safe Neighbours: ['24', '33']
Unknown Neighbours: ['13']
Unsafe Neighbours: ['P22']

Action: Left
(True, [2, 3], 'right')
Action: Forward
(True, [3, 3], [None, 'breeze', None, None, None])

Rule: ~S33 => ~W32 and ~W34 and ~W43 and ~W23, ~S33
By Modus Ponens:
~W32 and ~W34 and ~W43 and ~W23

Rule: ~W32 and ~W34 and ~W43 and ~W23
By And Elimination:
~W32
~W34
~W43
~W23

Rule: B33 => P32 or P34 or P43 or P23, B33
By Modus Ponens:
P32 or P34 or P43 or P23

Rule: P32 or P34 or P43 or P23
~P32
~P43
~P23
By Resolution:
P34

Rule: ~W31 and ~W33 and ~W42 and ~W22
By And Elimination:
~W31

~W33
~W42
~W22

Rule: P31 or P33 or P42 or P22
~P31
~P33
~P42
By Resolution:
P22

Rule: P33 or P42 or P22
~P33
~P42
By Resolution:
P22

Rule: P33 or P22
~P33
By Resolution:
P22

Rule: ~W42 and ~W44 and ~W53 and ~W33
By And Elimination:
~W42
~W44
~W53
~W33

Rule: P42 or P44 or P53 or P33
~P42
~P33
By Resolution:
P44 or P53

Rule: P44 or P53 or P33
~P33
By Resolution:
P44 or P53

Rule: ~W22 and ~W24 and ~W33 and ~W13
By And Elimination:
~W22
~W24
~W33
~W13

Rule: P22 or P24 or P33 or P13
~P24

~P33
By Resolution:
P22 or P13

Safe Neighbours: ['32', '43', '23']
Unknown Neighbours: []
Unsafe Neighbours: ['P34']

Action: Right
(True, [3, 3], 'down')
Action: Forward
(True, [3, 2], [None, 'breeze', None, None, None])

Rule: ~S32 => ~W31 and ~W33 and ~W42 and ~W22, ~S32
By Modus Ponens:
~W31 and ~W33 and ~W42 and ~W22

Rule: ~W31 and ~W33 and ~W42 and ~W22
By And Elimination:
~W31
~W33
~W42
~W22

Rule: B32 => P31 or P33 or P42 or P22, B32
By Modus Ponens:
P31 or P33 or P42 or P22

Rule: P31 or P33 or P42 or P22
~P31
~P33
~P42
By Resolution:
P22

Rule: W32 or W41 or W21
~W32
~W21
By Resolution:
W41

Rule: W32 or W41
~W32
By Resolution:
W41

Rule: ~P32 and ~P41 and ~P21
By And Elimination:
~P32

~P41
~P21

Rule: W41 or W43 or W52 or W32
~W43
~W32
By Resolution:
W41 or W52

Rule: ~P41 and ~P43 and ~P52 and ~P32
By And Elimination:
~P41
~P43
~P52
~P32

Rule: ~W32 and ~W34 and ~W43 and ~W23
By And Elimination:
~W32
~W34
~W43
~W23

Rule: P32 or P34 or P43 or P23
~P32
~P43
~P23
By Resolution:
P34

Safe Neighbours: ['31', '33', '42']
Unknown Neighbours: []
Unsafe Neighbours: ['P22']

Action: Forward
(True, [3, 1], ['stench', None, None, None, None])

Rule: S31 => W32 or W41 or W21, S31
By Modus Ponens:
W32 or W41 or W21

Rule: W32 or W41 or W21
~W32
~W21
By Resolution:
W41

Rule: ~B31 => ~P32 and ~P41 and ~P21, ~B31
By Modus Ponens:

~P32 and ~P41 and ~P21

Rule: ~P32 and ~P41 and ~P21
By And Elimination:
~P32
~P41
~P21

Rule: ~W22 and ~W31 and ~W11
By And Elimination:
~W22
~W31
~W11

Rule: P22 or P31 or P11
~P31
~P11
By Resolution:
P22

Rule: P22 or P31
~P31
By Resolution:
P22

Rule: ~W31 and ~W33 and ~W42 and ~W22
By And Elimination:
~W31
~W33
~W42
~W22

Rule: P31 or P33 or P42 or P22
~P31
~P33
~P42
By Resolution:
P22

Safe Neighbours: ['32', '21']
Unknown Neighbours: []
Unsafe Neighbours: ['W41']

Action: Right
(True, [3, 1], 'left')
Action: Forward
(True, [2, 1], [None, 'breeze', None, None, None])

Rule: ~S21 => ~W22 and ~W31 and ~W11, ~S21

By Modus Ponens:
~W22 and ~W31 and ~W11

Rule: ~W22 and ~W31 and ~W11
By And Elimination:
~W22
~W31
~W11

Rule: B21 => P22 or P31 or P11, B21
By Modus Ponens:
P22 or P31 or P11

Rule: P22 or P31 or P11
~P31
~P11
By Resolution:
P22

Rule: ~W12 and ~W21
By And Elimination:
~W12
~W21

Rule: P12 or P21
~P21
By Resolution:
P12

Rule: W32 or W41 or W21
~W32
~W21
By Resolution:
W41

Rule: ~P32 and ~P41 and ~P21
By And Elimination:
~P32
~P41
~P21

Safe Neighbours: ['31', '11']
Unknown Neighbours: []
Unsafe Neighbours: ['P22']

Action: Forward
(True, [1, 1], [None, 'breeze', None, None, None])
Action: Climb
(True, [1, 1], 'Congrats! R2 has saved Luke! +1000 points! Your final score: 958')

Scenario W5:

Current Direction : right
Current Location: [1, 1]
Current Percepts: [None, None, None, None, None]

Rule: ~S11 => ~W12 and ~W21, ~S11
By Modus Ponens:
~W12 and ~W21

Rule: ~W12 and ~W21
By And Elimination:
~W12
~W21

Rule: ~B11 => ~P12 and ~P21, ~B11
By Modus Ponens:
~P12 and ~P21

Rule: ~P12 and ~P21
By And Elimination:
~P12
~P21

Safe Neighbours: ['12', '21']
Unknown Neighbours: []
Unsafe Neighbours: []

Action: Left
(True, [1, 1], 'up')
Action: Forward
(True, [1, 2], [None, 'breeze', None, None, None])

Rule: ~S12 => ~W11 and ~W13 and ~W22, ~S12
By Modus Ponens:
~W11 and ~W13 and ~W22

Rule: ~W11 and ~W13 and ~W22
By And Elimination:
~W11
~W13
~W22

Rule: B12 => P11 or P13 or P22, B12
By Modus Ponens:
P11 or P13 or P22

Rule: P11 or P13 or P22
~P11
By Resolution:
P13 or P22

Rule: ~W12 and ~W21
By And Elimination:
~W12
~W21

Rule: ~P12 and ~P21
By And Elimination:
~P12
~P21

Safe Neighbours: ['11']
Unknown Neighbours: ['13', '22']
Unsafe Neighbours: []

Action: Forward
(True, [1, 3], [None, None, None, None, None])

Rule: ~S13 => ~W12 and ~W14 and ~W23, ~S13
By Modus Ponens:
~W12 and ~W14 and ~W23

Rule: ~W12 and ~W14 and ~W23
By And Elimination:
~W12
~W14
~W23

Rule: ~B13 => ~P12 and ~P14 and ~P23, ~B13
By Modus Ponens:
~P12 and ~P14 and ~P23

Rule: ~P12 and ~P14 and ~P23
By And Elimination:
~P12
~P14
~P23

Rule: ~W11 and ~W13 and ~W22
By And Elimination:
~W11
~W13
~W22

Rule: P11 or P13 or P22

~P11
~P13
By Resolution:
P22

Rule: P13 or P22
~P13
By Resolution:
P22

Safe Neighbours: ['12', '14', '23']
Unknown Neighbours: []
Unsafe Neighbours: []

[1, 3]
up
Action: Forward
(True, [1, 4], [None, 'breeze', None, None, None])

Rule: ~S14 => ~W13 and ~W24, ~S14
By Modus Ponens:
~W13 and ~W24

Rule: ~W13 and ~W24
By And Elimination:
~W13
~W24

Rule: B14 => P13 or P24, B14
By Modus Ponens:
P13 or P24

Rule: P13 or P24
~P13
By Resolution:
P24

Rule: ~W12 and ~W14 and ~W23
By And Elimination:
~W12
~W14
~W23

Rule: ~P12 and ~P14 and ~P23
By And Elimination:
~P12
~P14
~P23

Safe Neighbours: ['13']
Unknown Neighbours: []
Unsafe Neighbours: ['P24']

Action: Right
(True, [1, 4], 'right')
Action: Right
(True, [1, 4], 'down')
Action: Forward
(True, [1, 3], [None, None, None, None, None])

Rule: ~S13 => ~W12 and ~W14 and ~W23, ~S13
By Modus Ponens:
~W12 and ~W14 and ~W23

Rule: ~W12 and ~W14 and ~W23
By And Elimination:
~W12
~W14
~W23

Rule: ~B13 => ~P12 and ~P14 and ~P23, ~B13
By Modus Ponens:
~P12 and ~P14 and ~P23

Rule: ~P12 and ~P14 and ~P23
By And Elimination:
~P12
~P14
~P23

Rule: ~W11 and ~W13 and ~W22
By And Elimination:
~W11
~W13
~W22

Rule: P11 or P13 or P22
~P11
~P13
By Resolution:
P22

Rule: P13 or P22
~P13
By Resolution:
P22

Rule: ~W13 and ~W24

By And Elimination:
~W13
~W24

Rule: P13 or P24
~P13
By Resolution:
P24

Safe Neighbours: ['12', '14', '23']
Unknown Neighbours: []
Unsafe Neighbours: []

Action: Left
(True, [1, 3], 'right')
Action: Forward
(True, [2, 3], ['stench', 'breeze', None, None, None])

Rule: S23 => W22 or W24 or W33 or W13, S23
By Modus Ponens:
W22 or W24 or W33 or W13

Rule: W22 or W24 or W33 or W13
~W22
~W24
~W13
By Resolution:
W33

Rule: B23 => P22 or P24 or P33 or P13, B23
By Modus Ponens:
P22 or P24 or P33 or P13

Rule: P22 or P24 or P33 or P13
~P13
By Resolution:
P22 or P24 or P33

Rule: ~W12 and ~W14 and ~W23
By And Elimination:
~W12
~W14
~W23

Rule: ~P12 and ~P14 and ~P23
By And Elimination:
~P12
~P14
~P23

Safe Neighbours: ['13']
Unknown Neighbours: []
Unsafe Neighbours: ['P22', 'P24', 'W33']

Action: Shoot
(True, [2, 3], 'Blaster bolt was shot')
Action: Forward
(True, [3, 3], [None, None, None, None, 'scream'])

Rule: ~S33 => ~W32 and ~W34 and ~W23, ~S33
By Modus Ponens:
~W32 and ~W34 and ~W23

Rule: ~W32 and ~W34 and ~W23
By And Elimination:
~W32
~W34
~W23

Rule: ~B33 => ~P32 and ~P34 and ~P23, ~B33
By Modus Ponens:
~P32 and ~P34 and ~P23

Rule: ~P32 and ~P34 and ~P23
By And Elimination:
~P32
~P34
~P23

Rule: W22 or W24 or W33 or W13
~W22
~W24
~W33
~W13
By Resolution:

Rule: P22 or P24 or P33 or P13
~P33
~P13
By Resolution:
P22 or P24

Rule: P22 or P24 or P33
~P33
By Resolution:
P22 or P24

Safe Neighbours: ['32', '34', '23']
Unknown Neighbours: []
Unsafe Neighbours: []

Action: Left
(True, [3, 3], 'up')
up
Action: Forward
(True, [3, 4], [None, 'breeze', 'gasp', None, 'scream'])

Rule: ~S34 => ~W33 and ~W24, ~S34
By Modus Ponens:
~W33 and ~W24

Rule: ~W33 and ~W24
By And Elimination:
~W33
~W24

Rule: B34 => P33 or P24, B34
By Modus Ponens:
P33 or P24

Rule: P33 or P24
~P33
By Resolution:
P24

Rule: ~W32 and ~W34 and ~W23
By And Elimination:
~W32
~W34
~W23

Rule: ~P32 and ~P34 and ~P23
By And Elimination:
~P32
~P34
~P23

Safe Neighbours: ['33']
Unknown Neighbours: []
Unsafe Neighbours: ['P24']

Action: Grab
(True, [3, 4], 'R2-D2 has picked up Luke')
Action: Right
(True, [3, 4], 'right')
Action: Right

(True, [3, 4], 'down')
Action: Forward
(True, [3, 3], [None, None, None, None, 'scream'])

Rule: ~S33 => ~W32 and ~W34 and ~W23, ~S33
By Modus Ponens:
~W32 and ~W34 and ~W23

Rule: ~W32 and ~W34 and ~W23
By And Elimination:
~W32
~W34
~W23

Rule: ~B33 => ~P32 and ~P34 and ~P23, ~B33
By Modus Ponens:
~P32 and ~P34 and ~P23

Rule: ~P32 and ~P34 and ~P23
By And Elimination:
~P32
~P34
~P23

Rule: W22 or W24 or W33 or W13
~W22
~W24
~W33
~W13
By Resolution:

Rule: P22 or P24 or P33 or P13
~P33
~P13
By Resolution:
P22 or P24

Rule: P22 or P24 or P33
~P33
By Resolution:
P22 or P24

Rule: ~W33 and ~W24
By And Elimination:
~W33
~W24

Rule: P33 or P24

~P33
By Resolution:
P24

Safe Neighbours: ['32', '34', '23']
Unknown Neighbours: []
Unsafe Neighbours: []

Action: Forward
(True, [3, 2], [None, 'breeze', None, None, 'scream'])

Rule: ~S32 => ~W31 and ~W33 and ~W22, ~S32
By Modus Ponens:
~W31 and ~W33 and ~W22

Rule: ~W31 and ~W33 and ~W22
By And Elimination:
~W31
~W33
~W22

Rule: B32 => P31 or P33 or P22, B32
By Modus Ponens:
P31 or P33 or P22

Rule: P31 or P33 or P22
~P33
By Resolution:
P31 or P22

Rule: ~W32 and ~W34 and ~W23
By And Elimination:
~W32
~W34
~W23

Rule: ~P32 and ~P34 and ~P23
By And Elimination:
~P32
~P34
~P23

Safe Neighbours: ['33']
Unknown Neighbours: ['31']
Unsafe Neighbours: ['P22']

Action: Forward
(True, [3, 1], [None, None, None, None, 'scream'])

Rule: ~S31 => ~W32 and ~W21, ~S31
By Modus Ponens:
~W32 and ~W21

Rule: ~W32 and ~W21
By And Elimination:
~W32
~W21

Rule: ~B31 => ~P32 and ~P21, ~B31
By Modus Ponens:
~P32 and ~P21

Rule: ~P32 and ~P21
By And Elimination:
~P32
~P21

Rule: ~W31 and ~W33 and ~W22
By And Elimination:
~W31
~W33
~W22

Rule: P31 or P33 or P22
~P31
~P33
By Resolution:
P22

Rule: P31 or P22
~P31
By Resolution:
P22

Safe Neighbours: ['32', '21']
Unknown Neighbours: []
Unsafe Neighbours: []

Action: Right
(True, [3, 1], 'left')
Action: Forward
(True, [2, 1], [None, 'breeze', None, None, 'scream'])

Rule: ~S21 => ~W22 and ~W31 and ~W11, ~S21
By Modus Ponens:
~W22 and ~W31 and ~W11

Rule: ~W22 and ~W31 and ~W11

By And Elimination:
~W22
~W31
~W11

Rule: B21 => P22 or P31 or P11, B21
By Modus Ponens:
P22 or P31 or P11

Rule: P22 or P31 or P11
~P31
~P11
By Resolution:
P22

Rule: ~W12 and ~W21
By And Elimination:
~W12
~W21

Rule: ~P12 and ~P21
By And Elimination:
~P12
~P21

Rule: ~W32 and ~W21
By And Elimination:
~W32
~W21

Rule: ~P32 and ~P21
By And Elimination:
~P32
~P21

Safe Neighbours: ['31', '11']
Unknown Neighbours: []
Unsafe Neighbours: ['P22']

Action: Forward
(True, [1, 1], [None, None, None, None, 'scream'])
Action: Climb
(True, [1, 1], 'Congrats! R2 has saved Luke! +1000 points! Your final score: 977')

Part 2: Analyze it

4. I rely on an extensive set of logical rules that are based on where I am and what I can see in my surroundings. This method ensures that I consider all the possible rules that apply to my current situation. So, I carefully analyze the available information at each step to deduce any relevant rules that can help me make decisions. However, there are times when I'm uncertain about the safety of nearby locations. In such cases, when I explore these uncharted areas, I might run into problems, need to backtrack, or encounter unexpected dangers. This occasionally leads to situations where I unintentionally enter hazardous zones, even though I had no way of foreseeing the risks due to a lack of information about those specific places.

5. Yes, there were shorter routes available, but my approach was quite instinctual when guiding the agent's movements. I aimed to minimize the number of times the agent changed direction, so I initially prioritized exploring the path straight ahead if it seemed safe or unexplored. However, this sometimes led to the need to retrace my steps when the forward path didn't pan out, forcing me to return to the point where I initially made the wrong choice and then alter the agent's direction to explore other options.

6. Yes, there are many steps where the agent has to navigate blindly. Mostly in the beginning, since many times at the first step, there may be a stench, so both its neighbors can have Wampa, there was no way to know where, so it was by basic rule that I was following that, first go in the direction you are facing, if that is not correct then come back and make the turn. So, because of this blind navigation, many times my agent crashed.

Part 3: Expand your thinking

7. In different scenarios, the agent could find itself in need of searching for specific items or people, all while facing barriers like raging fires or lurking threats. For instance, within a blazing building, multiple paths could be blocked by the flames, and the agent's primary mission is to locate and rescue trapped individuals. By relying on sensory information such as the intensity of light and density of smoke, alongside auditory signals like people's distress calls, the agent can smartly navigate through the dangerous environment. This allows it to optimize its pathfinding strategy, enabling a quicker and more effective rescue, while simultaneously increasing the chances of ensuring the safety of those in distress.

8. The logical agent outshines conventional search agents by virtue of its advanced capabilities in deducing valuable insights from what it observes in the environment. In

contrast, typical search agents stick to predefined rules and static information to determine their best route. What sets logical agents apart is their continuous evaluation of the current environment, taking into account both the incoming data (percepts) and the knowledge they've gathered up to that point. This analytical approach equips logical agents with heightened resilience and adaptability, enabling them to make well-informed decisions at every turn. Consequently, they excel in handling dynamic situations and effectively navigating through ever-changing surroundings.

9. We can combine a search algorithm like the UCS method to guide our agent's movements. Each action is associated with certain points, so the search agent can choose the action with the fewest points, making the path shorter. Additionally, we can maintain a list of visited and unsafe locations, which we've identified using a logical agent. This helps the search agent avoid revisiting places unnecessarily. To simplify things, we can use basic methods like BFS or DFS to explore safe or uncharted areas. This way, we optimize our agent's navigation efficiently.

Part 4:

10. I spent around 10-12 hours on this assignment.

11. First I think it was very difficult to decide how I wanted to store the rules. And how to use the stored rules easily and with efficiency for inference. I didn't want to go over all the rules and do a brute-force approach by comparing every rule with every other rule. So finally, in the end, I decided it to be a dictionary with Boolean Values and Rules as keys. It was also very difficult for me to understand, what I should pass as an argument to the inference rules. How to process further, and how to infer all the possible logic at every step.

12. I liked that I understood propositional logic very clearly after doing this assignment. But it would have been better if the instructions and expectations were more clear. If we were given some example of how you want the output, or how we are supposed to move the agent. I spent a lot of time thinking about how to move the agent dynamically and after I cleared with Professor, she told me that it is ok if we move it manually.