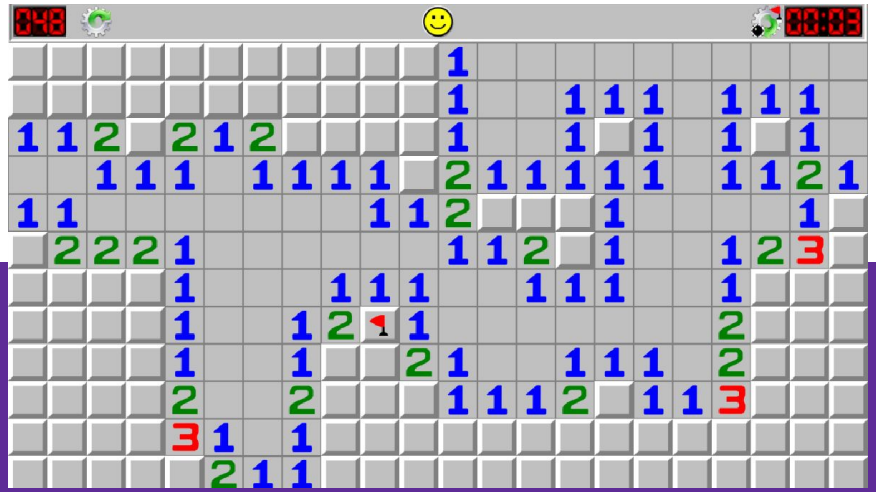
Buscaminas

IA - TP02

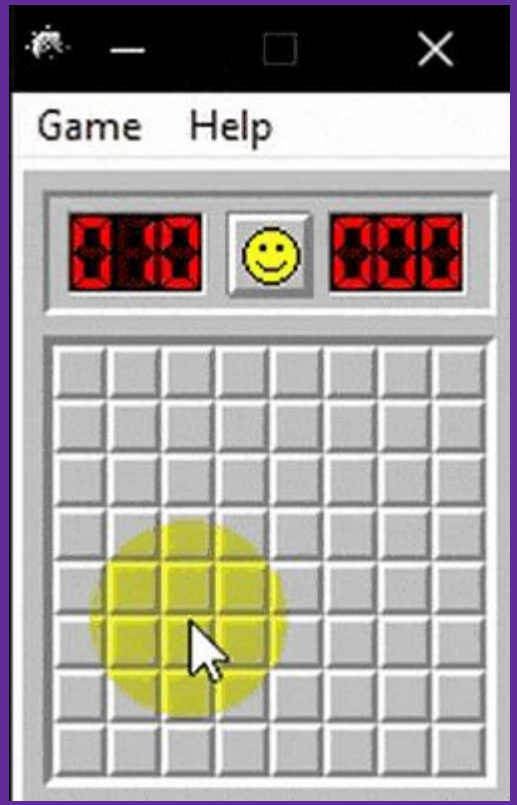


- Martín Cometta

¿De que trata el juego Buscaminas?

El Buscaminas es un juego de estrategia y lógica en el que debes evitar minas ocultas mientras descubres las casillas del tablero. Los números en las casillas te indican cuántas minas hay alrededor. El objetivo es despejar el tablero sin hacer estallar ninguna mina.

¿Como se juega?

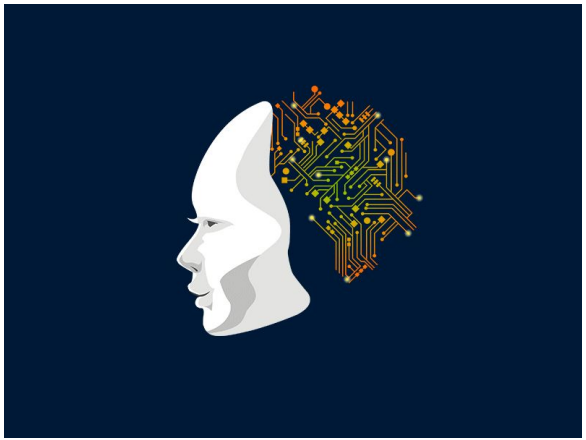


¿Como se podría implementar IA en este videojuego?

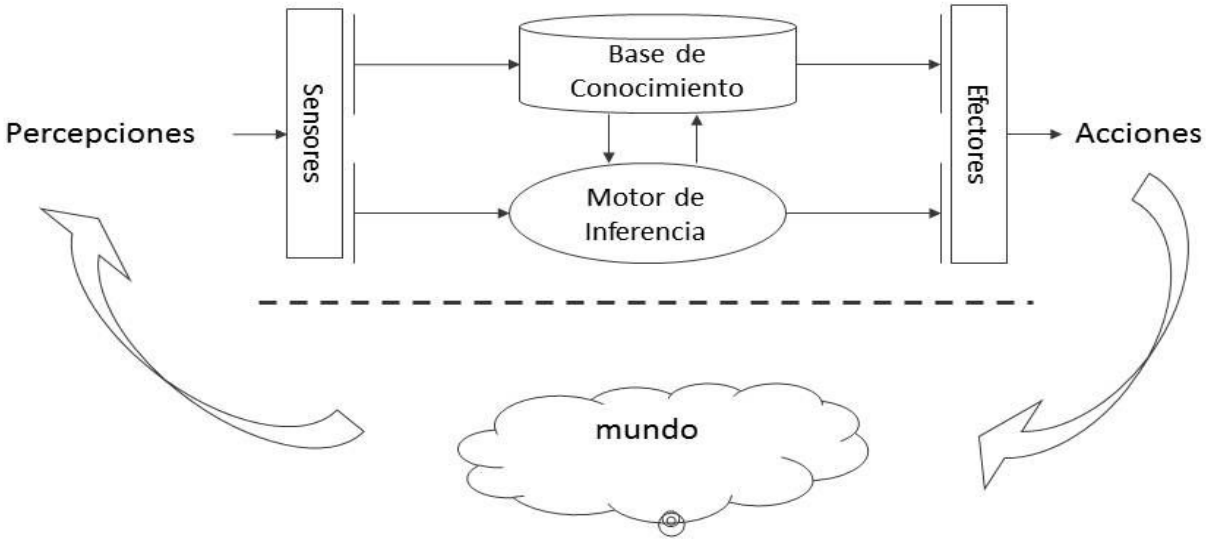
Lógica proposicional

Agentes basados en conocimiento → Toman decisiones en base a su conocimiento

Razonamiento en base a **SENTENCIAS**



Agentes Basado en Conocimiento



TELL y **ASK**

Sentencias

Afirmaciones o declaraciones que pueden ser verdaderas o falsas.

EJEMPLO:

1. Si el día está soleado, entonces hará calor.

¿Y con otra proposición? Ya que esa no nos afirma nada.

2. Hoy está soleado.

Teniendo estas dos proposiciones, podemos inferir:

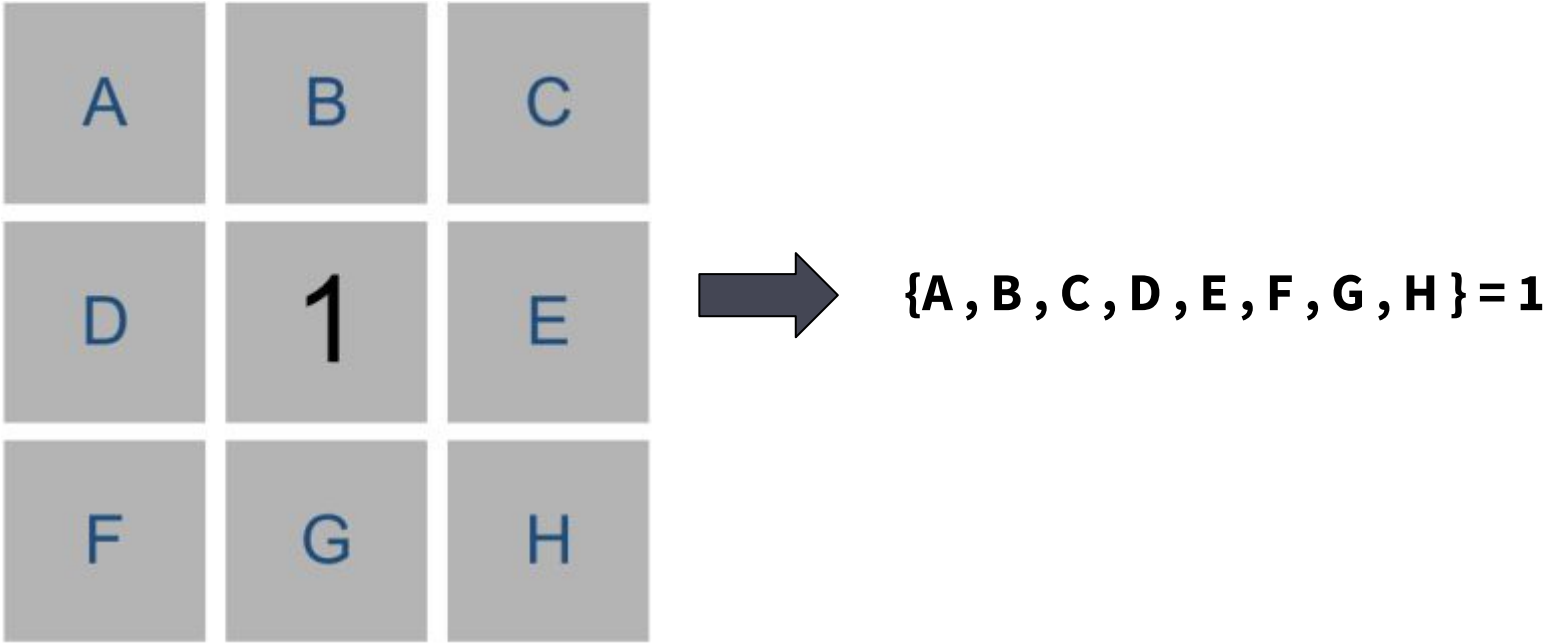
3. **Hoy hará calor.**

Sentencias Lógicas

en el juego Buscaminas

Sentencias lógicas

Manera en la que representaremos el conocimiento.



Clase “Sentence”

```python
class Sentence():
    """
    Logical statement about a Minesweeper game
    A sentence consists of a set of board cells,
    and a count of the number of those cells which are mines.
    """

    def __init__(self, cells, count):
        self.cells = set(cells)
        self.count = count
```

Análisis de celdas de “Sentence”

```python
def known_mines(self):
    """
    Returns the set of all cells in self.cells known to be mines.
    """
    # Si self.count = len(self.cell) es porque todas son minas.
    if self.count == len(self.cells):
        return self.cells.copy()
    return set()
```

```python
def known_safes(self):      You, 4 days ago • listo
    """
    Returns the set of all cells in self.cells known to be safe.
    """
    # Si es 0, es porque todas las celdas son seguras.
    if self.count == 0:
        return self.cells.copy()
    return set()
```

Marcado de celdas de “Sentence”

```python
def mark_mine(self, cell):
    """
    Updates internal knowledge representation given the fact that
    a cell is known to be a mine.
    """
    if cell in self.cells:
        # La celda es una mina, por lo que debe ser eliminada de la sentencia
        self.cells.remove(cell)
        # Se reduce el número de minas conocido en la sentencia
        self.count -= 1
```

```python
def mark_safe(self, cell):
    """
    Updates internal knowledge representation given the fact that
    a cell is known to be safe.
    """
    if cell in self.cells:
        # Celda segura, la elimino de la sentencia
        self.cells.remove(cell)
```

Inteligencia Artificial

en el juego Buscaminas

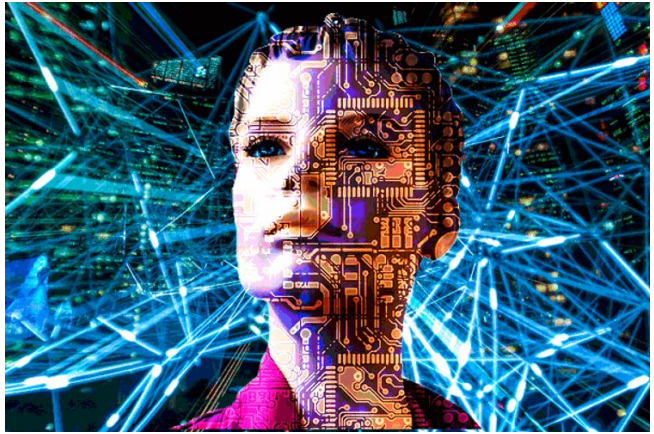
Clase “MinesweeperAI”

```python
class MinesweeperAI():
    """
    Minesweeper game player
    """

    def __init__(self, height=8, width=8):

        # Altura y ancho inicial.
        self.height = height
        self.width = width

        # Rastro de las celdas en las que he clickeado.
        self.moves_made = set()

        # Rastro de las celdas que son minas y las que son seguras.
        self.mines = set()
        self.safes = set()

        # Conocimiento
        self.knowledge = []
```

Marcado de celdas de “MinesweeperAI”

```python
def mark_mine(self, cell):
    """
    Marks a cell as a mine, and updates all knowledge
    to mark that cell as a mine as well.
    """
    self.mines.add(cell)
    for sentence in self.knowledge:
        sentence.mark_mine(cell)

def mark_safe(self, cell):
    """
    Marks a cell as safe, and updates all knowledge
    to mark that cell as safe as well.
    """
    self.safes.add(cell)
    for sentence in self.knowledge:
        sentence.mark_safe(cell)
```

Función **add_knowledge**, o agregar conocimiento, es donde podemos ver realmente la inferencia…



```python
def add_knowledge(self, cell, count):
    i, j = cell #posicion
    # 1) Marcar la celda como un movimiento realizado
    self.moves_made.add(cell)
    # 2) Marcar la celda como segura
    self.mark_safe(cell)
    # 3) Agregar una nueva sentencia a la base de conocimientos para indicar que count de los vecinos de cell son minas
    neighbors = []
    for x in range(max(0, i - 1), min(self.height, i + 2)):
        for y in range(max(0, j - 1), min(self.width, j + 2)):
            neighbor = (x, y)
            if neighbor != cell and neighbor not in self.safes:
                neighbors.append(neighbor)
    new_sentence = Sentence(neighbors, count)
    self.knowledge.append(new_sentence)
    # 4) y 5) Actualizar celdas como seguras o minas según la nueva información
    for sentence in self.knowledge:
        known_mines = sentence.known_mines()
        known_safes = sentence.known_safes()
        if known_mines:
            for mine in known_mines.copy():
                self.mark_mine(mine)
        if known_safes:
            for safe in known_safes.copy():
                self.mark_safe(safe)
    # 5) Agregar nuevas sentencias si se pueden inferir      You, 4 days ago • listo …
    new_knowledge = []
    for sentence1 in self.knowledge:
        for sentence2 in self.knowledge:
            if sentence1 != sentence2:
                if sentence1.cells.issubset(sentence2.cells):
                    new_cells = sentence2.cells - sentence1.cells
                    new_count = sentence2.count - sentence1.count
                    new_sentence = Sentence(new_cells, new_count)
                    if new_sentence not in self.knowledge and new_sentence not in new_knowledge:
                        new_knowledge.append(new_sentence)
    # Actualizar celdas vecinas de 'cell' si es posible deducir información
    for sentence in new_knowledge:
        for neighbor in sentence.cells.copy():
            if neighbor != cell:
                if neighbor not in self.mines and neighbor not in self.safes:
                    if sentence.count == 0:
                        self.mark_safe(neighbor)
                    elif sentence.count == len(sentence.cells):
                        self.mark_mine(neighbor)
    self.knowledge.extend(new_knowledge)
```

Movimientos de “MinesweeperAI”

```python
def make_safe_move(self):

    for i in range(self.height):
        for j in range(self.width):
            cell = (i, j)
            # Verificar si la celda es segura y no ha sido movida
            if cell not in self.moves_made and cell in self.safes:
                return cell
    return None  # No se encontró un movimiento seguro
```

```python
def make_random_move(self):

    possible_moves = []
    for i in range(self.height):
        for j in range(self.width):
            cell = (i, j)
            # Verificar si la celda no ha sido movida y no es una mina conocida
            if cell not in self.moves_made and cell not in self.mines:…

    if possible_moves:
        return random.choice(possible_moves)  # Aleatorio
    else:
        return None  # Si no hay aletorio
```

Muchas gracias!