**دانشگاه اصفهان**

**دانشکده مهندسی کامپیوتر**

# پروژه مبانی هوش مصنوعی

# گزارش کتبی فاز چهارم

**اعضای گروه:**

**فرناز موحدی**

**ارشیا شفیعی**

**رضا چراخ**

**بهمن ۱۴۰۳**

**فهرست مطالب**

**عنوان** **صفحه**

# فصل اول
# معرفی پایگاه دانش

## ۱-۱- مقدمه

فصل اول گزارش، به حل مسئله‌ای می‌پردازد که در آن عامل با دریافت یک grid به عنوان توصیف محیط، با استنتاج روی پایگاه دانش نوشته شده با منطق درجه اول، مجموعه‌ای از کنش‌ها برای حرکت در محیط را بدست می‌آورد. در نتیجه‌ی این کنش‌ها، عامل مسیر بهینه را به طوری که از همه اهداف (خوک‌ها) بگذرد، پیدا می‌کند.

## ۱-۲- محیط بازی

### ۱-۲-۱- تعریف شبکه

شبکه با استفاده از grid_size(8, 8) به‌صورت یک محیط ۸x۸ تعریف شده است. این شبکه شامل سلول‌هایی است که نشان‌دهنده فضاهای معتبر برای حرکت هستند (tile). موانع (سنگ‌ها) از حرکت معتبر حذف شده‌اند.

### ۱-۲-۱- عناصر محیط

- عامل (پرنده): موقعیت اولیه پرنده با دستور bird(BirdX, BirdY) تعریف شده است.
- اهداف (خوک‌ها): خوک‌ها در مختصات خاصی قرار دارند که با دستور pig(PigX, PigY) تعریف شده اند. پرنده باید از تمام خوک‌ها عبور کند تا برنده بازی شود.
- موانع (سنگ‌ها): سنگ‌ها که حرکت را مسدود می‌کنند، با دستور rock(X, Y) تعریف شده‌اند. حرکت به موقعیت‌های دارای سنگ نامعتبر است.
- سلول‌ها(Tiles) : موقعیت‌های معتبر برای حرکت با دستور tile(X, Y) تعریف شده‌اند.

### ۱-۲-۲- ویژگی‌های محیط

محیط قطعی و شناخته‌شده است، یعنی کنش‌ها نتایج کاملاً قابل پیش‌بینی دارند و پرنده از ابتدا از موقعیت تمام خوک‌ها، سنگ‌ها و سلول‌ها آگاه است.

## ۱-۳- پایگاه دانش

### ۱-۳-۱- قوانین مربوط به حرکت

قوانین حرکت در این پروژه با استفاده از دو قانون اصلی تنظیم شده‌اند. قانون adjacent/2 سلول‌های مجاور را مشخص می‌کند (سلول‌هایی که از سلول فعلی با یکی از کنش‌های بالا، پایین، چپ و راست قابل دسترسی هستند) این قانون تضمین می‌کند که حرکت‌ها در محدوده grid باقی بمانند.

قانون دوم، can_move/1، حرکت به سلول‌های معتبر را بررسی می‌کند و اطمینان می‌دهد که سلول انتخاب‌شده معتبر باشد به این معنا که حاوی سنگ نباشد.

برای ساده‌تر کردن تعریف اقدامات، حرکات به‌صورت عددی کدگذاری شده‌اند. برای مثال، عدد 0 به معنای حرکت به چپ، عدد 1 به معنای حرکت به راست، عدد 2 به معنای حرکت به بالا و عدد 3 به معنای حرکت به پایین است. قانون action_for_move/3 این حرکات را بین مختصات ترجمه می‌کند.

### ۱-۳-۲- قوانین مربوط مسیریابی

در بخش منطق مسیریابی، از الگوریتم جستجوی عرضی (BFS) برای پیدا کردن مسیر استفاده شده است. قانون bfs/4 وظیفه دارد مسیر بین موقعیت شروع و یک هدف را پیدا کند. در اینجا منظور از موقعیت شروع موقعیت عامل و منظور از هدف موقعیت خوک است. این قانون تمام مسیرهای ممکن از موقعیت فعلی را بررسی کرده و از قوانین adjacent/2 و can_move/1 برای اعتبارسنجی استفاده می‌کند. اگر موقعیت هدف پیدا شود، مسیر به‌صورت یک لیست از سلول‌ها بازگردانده می‌شود.

قانون path_to_actions/2 این مسیر را به یک لیست از کنش‌ها تبدیل می‌کند.

برای رسیدن به تمام اهداف، پرنده باید تمامی خوک‌ها را یکی پس از دیگری بازدید کند. قانون visit_all_pigs_graph/3 این فرآیند را مدیریت می‌کند. این قانون ابتدا نزدیک‌ترین خوک را با استفاده از قانون find_nearest_pig/4 پیدا می‌کند و سپس اقدامات لازم برای رسیدن به آن را محاسبه کرده و این روند را تا بازدید از تمام خوک‌ها ادامه می‌دهد.

برای انتخاب نزدیک‌ترین خوک، قانون find_nearest_pig/4 فاصله همه خوک‌ها را بررسی کرده و نزدیک‌ترین خوک را انتخاب می‌کند.

### ۱-۳-۲- قوانین مربوط مسیریابی

برای یکپارچه‌سازی این سیستم با پایتون، قانون solve_to_python/1 به‌عنوان نقطه ورود اصلی عمل می‌کند. این قانون موقعیت اولیه پرنده و لیست خوک‌ها را دریافت کرده و توالی کامل کنش‌های لازم برای بازدید از تمام خوک‌ها را محاسبه می‌کند.

## ۱-۴- مراحل اجرای برنامه

۱-۴-۱- اضافه کردن اطلاعات Grid به فایل Prolog

```python
env = FirstOrderAngry(template='test4')

with open("grid.pl", "a") as file:
    grid = env.grid

    grid_size = len(grid)

    # Write grid size fact
    file.write("% Define the grid size\n")
    file.write(f"grid_size({grid_size}, {grid_size}).\n\n")

    # Initialize fact lists
    bird_positions = []
    pig_positions = []
    rock_positions = []
    tile_positions = []

    # Iterate over the grid to categorize facts
    for row in range(grid_size):
        for col in range(grid_size):
            cell = grid[row][col]
            if cell == "B":
                bird_positions.append(f"bird({row}, {col}).\n")
            elif cell == "P":
                pig_positions.append(f"pig({row}, {col}).\n")
                tile_positions.append(f"tile({row}, {col}).\n")
            elif cell == "R":
                rock_positions.append(f"rock({row}, {col}).\n")
            elif cell == "T":
                tile_positions.append(f"tile({row}, {col}).\n")

    # Write bird facts
    file.write("% Bird position\n")
    file.writelines(bird_positions)
    file.write("\n")

    # Write pig facts
    file.write("% Pig positions\n")
    file.writelines(pig_positions)
    file.write("\n")

    # Write rock facts
    file.write("% Rock positions\n")
    file.writelines(rock_positions)
    file.write("\n")

    # Write tile facts
    file.write("% Tile positions (valid positions to move)\n")
    file.writelines(tile_positions)
    file.write("\n")
```

ابتدا سایز grid را در ادامه فایل پرولاگ اضافه می‌کنیم. سپس روی خانه‌های grid پیمایش می‌کنیم و مکان‌های پرنده، خوک‌ها، سنگ‌ها و خانه‌های خالی در لیست مربوط به خود اضافه می‌شوند و در آخر سر همگی به فایل پرولاگ اضافه می‌شوند.

۱-۴-۲- حل مسئله توسط پایگاه دانش

```
# Initialize Prolog and load your Prolog file
prolog = Prolog()
prolog.consult("grid.pl")

# Get the action list from Prolog
actions = list(prolog.query("solve_to_python(Actions)."))[0]["Actions"]

# Initialize your game
screen, clock = PygameInit.initialization()
FPS = 1

env.reset()

running = True
current_action_index = 0

while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            running = False

    # Select actions based on the extracted path
    if current_action_index < len(actions):
        action = actions[current_action_index]
        current_action_index += 1
    else:
        print("No more actions in the sequence.")
        break

    bird_pos, is_win = env.bird_step(action)
    env.render(screen)

    if is_win:
        print("Win")
        running = False

    pygame.display.flip()
    clock.tick(FPS)

pygame.quit()
```

ابتدا فایل پرولاگ آماده شده را خوانده و آماده استفاده می‌کنیم. سپس کوئری مربوط به حل مسئله را به پایگاه دانش می‌دهیم و در خروجی پایگاه دانش یک آرایه از عدد اکشن‌ها دریافت می‌کنیم. سپس در یک حلقه یکی یکی از اکشن‌های دریافت شده برای انتخاب حرکت بعدی پرنده استفاده می‌کنیم.

# فصل دوم
# نتایج

## ۲-۱- مثال

در ادامه دو نمونه اجرای برنامه در محیط simple و یک محیط test در قالب gif نشان داده شده است که هر دو مسیر بهینه را طی می‌کنند.


**۲-۱- اجرای بازی در محیط *simple***


**۲-۲- اجرای بازی در محیط *test***