Veštačka inteligencija

Projekat – Dominacija (Domineering)

Faza II – Implementacija operatora promene stanja

Stefan Stojadinović 17975, Andrija Tošić 18015

Funkcija koja obezbeđuje odigravanje partije između dva igrača

- Unos početnih parametara igre
  `def input_board_dimensions() -> tuple[int, int]`
- Unos poteza i proveru da li je potez ispravan
  `def input_move(to_move: Turn) -> Move | None`
  `def is_valid_move(state: State, move: Move) -> bool`
- Unos novog poteza sve dok on nije ispravan
  `def input_valid_move(state: State) -> Move`
- Odigravanje novog poteza ako je ispravan i promenu trenutnog stanja igre (table)
  `def derive_state(state: State, move: Move) -> None | State`
- Prikaz novonastalog stanja igre (table) nakon odigravanja poteza
  `def print_state(state: State) -> None`
- Proveru kraja i određivanje pobednika u igri nakon odigravanja svakog poteza, odnosno promene stanja igre (table)
  `def is_game_over(state: State) -> bool`

```
def game_loop() -> None:
    n, m = input_board_dimensions()

    # initial_turn = input_move_order()
    # player1 = input_player_type()
    # player2 = input_player_type()

    state = create_initial_state(n, m)

    while not is_game_over(state):
        print_state(state)

        move = input_valid_move(state)

        new_state = derive_state(state, move)
        if new_state:
```

```
            state = new_state

    print_state(state)
```

Funkcija za operator promene stanja problema

- Određivanje svih mogućih poteza igrača na osnovu stanja problema

  ```
  def possible_moves(state: State) -> set[Move]
  ```

- Funkcija koja na osnovu zadatog poteza i zadatog stanje igre (table) formira novo stanje igre (table)

Iz skupova koji određuju stanje igre izbacuju se potezi koji su onemogućeni zadatim potezom (izbacuju se mogući potezi iz skupova mogućih poteza za vertikalnog i horizontalnog igrača, odigrani potez dodaje se u odgovarajući skup odigranih poteza).

Funkcija u zavisnosti od validnosti poteza u zadatoj poziciji vraća novo izvedeno stanje, ili `None` u slučaju da prosleđeni potez nije moguć.

```
def derive_state(state: State, move: Move) -> None | State
```

- Funkcija koja na osnovu zadatog igrača na potezu i zadatog stanje igre (table) formira sva moguća stanje igre (sve moguće table), korišćenjem funkcija iz prethodne stavke

```
def generate_possible_states(state: State) -> Iterable[State]
```