

Corresponding Raw Content in Books: b

Chart 2.1 Characteristics of the principle of penetration
... ...

B) Line of the ball and opponent's goal line.

- Reduce the distance between the player in possession of the ball and the opponent's goal or goal line;
 - Unbalance opponent's defensive organization;
 - Directly attack the opponent or the goal;
 - Create advantageous attacking situations in numerical and spatial terms.
-

Player in possession of the ball.

- Carrying the ball through the available space (with or without defenders ahead).
 - Performing dribbles in search of numerical advantage in attacking situations or that enable the sequence of the play towards the opponent's goal line or goal.
 - Carrying the ball towards the opponent's goal line or goal.
 - Performing dribbles towards the opponent's goal line or goal searching for favourable conditions for a pass/assistance to a teammate to resume the play.
-



corresponding scenarios (rendered by simulator)

Imaginary State: \hat{s}

```
"active_player": 1, "active_player_role": "Forward", "game_mode": "Normal",
"ball_ownership": 1, "ball_ownership_player": 1, "ball_zone": [11, 7], "ball_direction": "east",
"player_0": {"team": "Left", "role": "Goalkeeper", "zone": [1, 7], "direction": "east"}, 
"player_1": {"team": "Left", "role": "Forward", "zone": [11, 7], "direction": "east"}, 
...
"player_21": {"team": "Right", "role": "Forward", "zone": [10, 4], "direction": "west"}}
```

Retrieved Knowledge Scope: K^S

```
{ 'preferred scope description': "This policy is best suited for making tactical decisions based on the opponent's system and style of play.", 'game_mode': 'Normal', 'active_player_role': "[Goalkeeper, Forward, Defender, Midfielder]", 'ball_ownership': "[Left, Right]", ... }
```

$$(1) \text{ retrieved by} \\ (\{K\}, \{K^S\}) = R_{\text{scope}}(\hat{s}, K^S)$$

Corresponding Aggregated Knowledge: K

```
'def assess_player_actions(player_observations, match_context,
ball_observations):'
    player_actions = {}
    for player, observations in player_observations.items():
        if player.role == 'defender':
            player_actions[player] = assess_defender_responses(observations,
match_context.get_opponent(player), ball_observations)
        elif player.role == 'midfielder':
            player_actions[player] = assess_midfielder_roles(observations,
match_context)
        elif player.role == 'forward':
            player_actions[player] = assess_forward_decisions(observations,
match_context, ball_observations.get_goal(), ball_observations)
        elif player.role == 'goalkeeper':
            ...
            ...
'def assess_forward_decisions(forward_observations, match_context,
goal, ball_observations):
    if is_in_shooting_range(goal) and
(ball_observations['is_controlled_by'] == forward_observations or not
'is_closely_marked()'):
        if has_clear_shot(goal) or opponent_actions == 'shoot':
            return 'take_shot'
        else:
            return 'pass_to_better_positioned_teammate' or
'dribble_to_improve_angle'
    else:
        return 'move_to_receive_pass' or 'create_space_for_teammates'"
```

$$(2) \text{ instantiated by } K_I = M_{\text{inst}}(\hat{s}, |\mathcal{M}|, \{K\})$$

Instantiated Knowledge: K_I

"analyze": "The current observation shows that the active player is a forward (player_1) from the Left team, who has possession of the ball in zone (11, 7). The ball is ... e defender (player_16) directly in front of him in zone (15, 7). Given this setup and the provided code-based knowledge, the optimal decision for the forward would be to either dribble or pass to a teammate in a better position. Player_2 (another forward) is in zone (12, 7), which could be a good option for a short pass. Additionally, player_10 (forward) is in zone (11, 9), which might also be a viable passing option.'],

```
'code': ['def football_manager_policy(observation):',
    def should_shoot(player, goal_info):
        distance_to_goal = calculate_distance(player['zone'], goal_info['zone'])
        if distance_to_goal < 10 and player['confidence'] > 50:
            return True
        return False
    def has_space(player, opponents):
        for opponent in opponents:
            if calculate_distance(player['zone'], opponent['zone']) < 10:
                return False
        return True
    def determine_action_for_forward(player, ball_position, teammates, opponents):
        if should_shoot(player, {'zone': [20, 6]}):
            return 12 # action_shot
        elif has_space(player, opponents):
            return 17 # action_dribble
        else:
            open_teammates = [teammate for teammate in teammates
if calculate_distance(player['zone'], teammate['zone']) < 5]
            if open_teammates:
                return 11 # action_short_pass
            else:
                return 0 # action_idle
        ...
        if active_player_role == 'Forward':
            return determine_action_for_forward(observation[f'player_{active_player}'],
'ball_position, teammates, opponents')
        else:
            return 0 # action_idle",]
```

$$(3) \text{ generated by } \hat{a} = M_{\pi}(\hat{s}, K_I)$$

Imaginary Action: \hat{a}

17 # action_dribble