Juan Barrera

Professor Mark Voortman

Week 11 Project Explanation

How Automated Planning (PDDL) Connects to My Track Meet Project

1. Context of My Project

My final project is a **lane assignment and progression system** for athletics competitions. It generates heats, assigns athletes to lanes using fairness rules (serpentine distribution and centrality), simulates results with noise, and progresses athletes round by round (heats → semifinals → finals). It enforces constraints such as:

- **Capacity**: maximum 8 athletes per heat.
- **Centrality**: fastest athletes in middle lanes.
- **Uniqueness**: each athlete appears once per round.
- **Progression rules**: winners auto-qualify, others advance by fastest times.

This system already integrates AI concepts from **agents, search, and CSPs**. However, automated planning (PDDL) adds another layer: **scheduling and coordination of heats across time slots and resources**.

2. Why Planning Is Relevant

Planning is about finding a sequence of actions that leads from an initial state to a goal state under constraints. In the context of my project:

- **Initial state**: athletes unassigned, heats unscheduled, resources free.
- **Actions**: check-in athletes, assign heats to slots, run heats.
- **Constraints**: track availability, officials availability, athletes must check in before running, no double-booking.
- **Goal state**: all heats scheduled and executed, resources respected, athletes properly checked in.

This complements my lane assignment system by ensuring that the heats it generates can actually be **scheduled and run feasibly** in real-world conditions.

3. Code Implementation Explained

The prototype Python code models a **simplified STRIPS-style planner**:

a. State Representation

- States are sets of fluents (facts), e.g.:
  - `At(A1, Home)` → athlete A1 is at home.
  - `HeatUnscheduled(H1)` → heat H1 has not yet been scheduled.
  - `TrackFree(T1, S1)` → track T1 is free in slot S1.
- The initial state includes all heats unscheduled, all athletes not checked in, and all slots/resources free.

b. Actions

Each action has **preconditions** (what must be true before it can be applied) and **effects** (what changes after execution):

- **CheckIn(athlete)**
  - Preconditions: athlete exists and is not checked in.
  - Effects: mark athlete as checked in.
- **AssignHeatToSlot(heat, slot)**
  - Preconditions: heat unscheduled, slot free, track and officials free.
  - Effects: heat assigned, slot reserved, resources reserved.
- **RunHeat(heat, slot)**
  - Preconditions: heat assigned to slot, slot reserved, all athletes checked in.
  - Effects: heat marked as run, resources freed after use.

c. Goal Test

The planner checks if **all heats have been run** at some slot. This ensures the meet is fully scheduled and executed.

d. Search Algorithm

- Uses **A*** search to explore sequences of actions.
- Heuristic: counts heats not yet run, athletes not checked in, and heats not yet assigned.
- Produces a valid plan (sequence of actions) that satisfies all constraints.

4. Example Plan Output

For a toy meet with 2 heats and 4 athletes, the planner might produce:

1. CheckIn(A1)
2. CheckIn(A2)
3. CheckIn(A3)
4. CheckIn(A4)
5. AssignHeatToSlot(H1, S1)
6. RunHeat(H1, S1)
7. AssignHeatToSlot(H2, S2)
8. RunHeat(H2, S2)

This plan ensures:

- All athletes are checked in before running.
- Each heat is assigned to a free slot.
- Track and officials are not double-booked.
- Both heats are completed successfully.

5. How It Complements My Project

- My **lane assignment system** ensures fairness and progression.
- The **planning system** ensures feasibility of scheduling and execution.
- Together, they form a **complete meet management solution**:
    - Fair lane assignments.
    - Valid progression rules.
    - Feasible scheduling under resource and time constraints.